HITACHI
Inspire the Next

Cosminexus V11 アプリケーションサーバ アプリケーション開発ガイド

手引・操作書

3021-3-J20-50

前書き

■ 対象製品

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」の前書きの対象製品の説明を参照してください。

■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

■ 発行

2024 年 2 月 3021-3-J20-50

■ 著作権

変更内容

変更内容(3021-3-J20-50) uCosminexus Application Server 11-40，uCosminexus Client 11-40，uCosminexus Developer 11-40，uCosminexus Service Architect 11-40，uCosminexus Service Platform 11-40

追加・変更内容	変更個所
Developer をインストールするときの注意事項を追加した。	2.2.1(3)
JDK17 をインストールした環境についての注意事項を追加した。	2.3.2(5)，2.3.3，2.3.5
Eclipse のバージョンについての注意事項を追加した。	2.4.1(2)，付録 B.1(1)
JDK17 サポートに伴い，［JDK 準拠］の項目について説明を変更した。	2.5.2
JDK17 サポートに伴い，［構成］で Java のバージョン「17」を選択する場合のポイントを追加した。	4.4.1，4.4.2，4.4.3
Java プログラムのコンパイルについての注意事項を追加した。	付録 L.3

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルをお読みになる際の前提情報については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」のはじめにの説明を参照してください。

目次

第 2 編　Eclipse を使用した J2EE アプリケーションの開発

1 アプリケーション開発の概要

アプリケーションサーバで動作する J2EE アプリケーションを開発するには，Developer で開発する J2EE アプリケーションの形式や構成を理解しておく必要があります。

この章では，Developer を使用した J2EE アプリケーション開発の特長や開発の流れを説明します。また，Developer が提供する機能と開発環境のマシン構成についても説明します。また，そのほかのアプリケーションの開発の概要についても説明します。

1.1　Developer による J2EE アプリケーション開発の特長

アプリケーションサーバで動作する J2EE アプリケーションは，Developer を使用して開発できます。

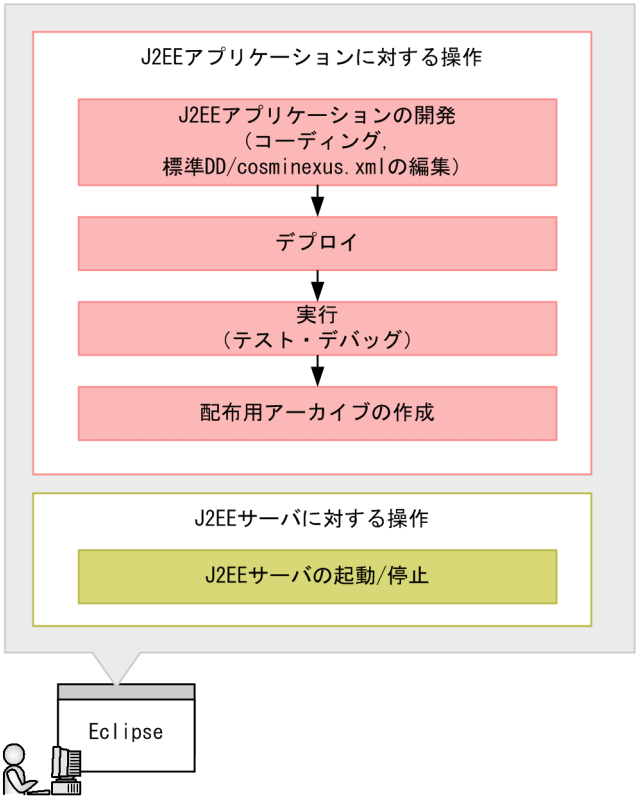
ここでは，Developer を使用した J2EE アプリケーション開発の特長を説明します。

1.1.1　Eclipse との連携によるスムーズな J2EE アプリケーション開発

Developer では，Eclipse Web Tools Platform（Eclipse）を使用した J2EE アプリケーション開発をサポートしています。

Eclipse を使用すると，J2EE アプリケーション開発の一連の作業をシームレスに実行できます。Eclipse を使用した場合に開発環境から実行できる作業を次の図に示します。

図 1-1　Eclipse 使用時に開発環境から実行できる作業



図に示すように，Eclipse を使用した開発環境では，J2EE アプリケーションの開発からデプロイ，テスト，デバッグ，配布用アーカイブの作成までの J2EE アプリケーション開発の一連の作業を実行できます。開発時に作成したプログラムは，すべて Eclipse のプロジェクトとして管理されます。

次に，Eclipse を使用した J2EE アプリケーション開発の特長を説明します。

- アプリケーションサーバ独自の定義情報の定義

アプリケーションサーバ独自の定義情報は，cosminexus.xml という属性ファイルに記述します。Eclipse では，cosminexus.xml の作成および編集をサポートしています。これによって，開発効率を向上できます。

- J2EE サーバへのデプロイ・デバッグ

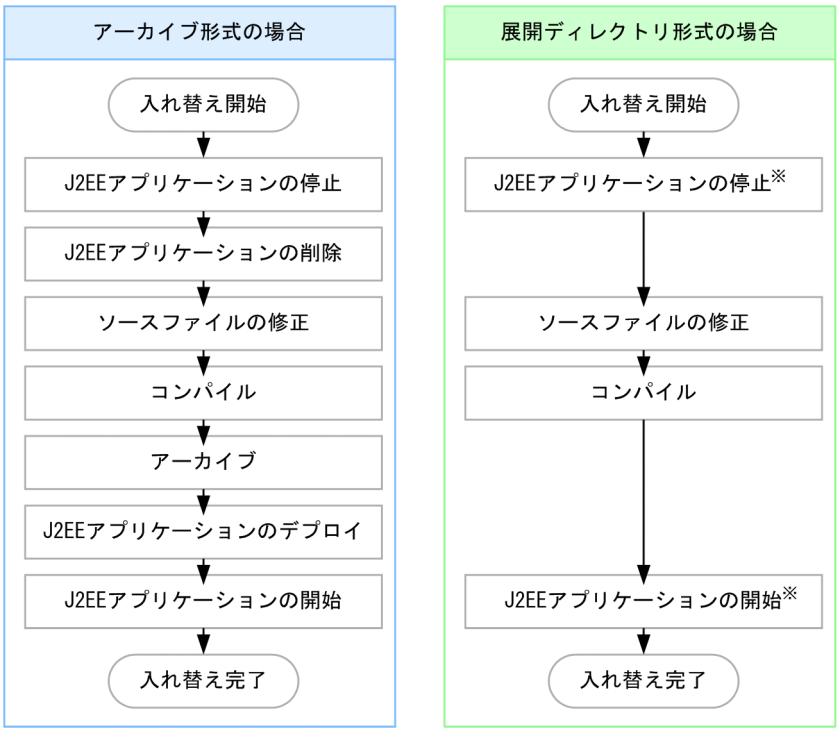
  Eclipse では，Eclipse のプロジェクトで開発した J2EE アプリケーションを J2EE サーバにデプロイできます。アーカイブの作成またはアプリケーションディレクトリへのコピーは，デプロイ時に実施されるため，ユーザ側でのコピーやアーカイブのインポート作業は不要になります。また，Eclipse で開発したソースは，デプロイ時に自動的にビルドされます。さらに，デプロイした J2EE アプリケーションは，Eclipse の機能を使用してデバッグできます。

1.1.2 展開ディレクトリ形式の利用による開発効率の向上

展開ディレクトリ形式とは，J2EE アプリケーションを構成するプログラムやファイルを，アーカイブしないでディレクトリの状態のままデプロイする形式です。

展開ディレクトリ形式の J2EE アプリケーションを利用すると，J2EE アプリケーションの入れ替えが容易になります。次の図に，アーカイブ形式と展開ディレクトリ形式の J2EE アプリケーションの入れ替え手順を示します。

図 1-2 J2EE アプリケーション入れ替え手順の比較（アーカイブ形式・展開ディレクトリ形式）



注※ リロード機能利用時は不要となります。

デプロイ済みの J2EE アプリケーションを入れ替える場合，アーカイブ形式の J2EE アプリケーションでは，J2EE アプリケーションを停止，削除してからソースファイルを修正します。また，修正したファイルを含めてアーカイブし直してから，再度デプロイする必要があります。

展開ディレクトリ形式の J2EE アプリケーションを入れ替える場合，J2EE アプリケーションの削除，アーカイブ，およびデプロイが不要となります。さらに**リロード機能**を使用すれば，J2EE アプリケーションを停止および開始しなくても，ソースファイルを自動で入れ替えることができます。

このように，展開ディレクトリ形式を利用すれば入れ替え手順が容易になるため，修正作業が多くなる J2EE アプリケーション開発の効率が向上します。

展開ディレクトリ形式の J2EE アプリケーションについては，「1.4.1　展開ディレクトリ形式の J2EE アプリケーション」を参照してください。

1.1.3　ユーザ拡張性能解析トレースの利用による J2EE アプリケーションの性能解析

ユーザ拡張性能解析トレースとは，アプリケーションのトレース対象の処理が実行された時点で出力される性能解析情報（トレース情報）を使用して，アプリケーションの処理性能を解析する機能です。

ユーザ拡張性能解析トレースを利用するには，**ユーザ拡張性能解析トレース設定ファイル**を作成する必要があります。Developer では，ユーザ拡張性能解析トレース設定ファイルの作成を支援します。作成したユーザ拡張性能解析トレース設定ファイルを基に J2EE アプリケーションのテスト・デバッグが実施でき，性能解析のための作業効率が向上します。

1.1.4　開発用データベースの標準提供

Developer では，開発環境で使用できるデータベース（組み込みデータベース）として，HiRDB/Single Server Version 10 を標準で提供しています。また，組み込みデータベースを操作するために，HiRDB SQL Executer を提供しています。HiRDB SQL Executer を使用すると，組み込みデータベースに対して対話形式で SQL を実行できます。これによって，開発環境でデータベース接続のテストも実施できるため，テスト・デバッグの作業効率が向上します。

なお，組み込みデータベースおよび HiRDB SQL Executer の用途は開発時の利用に限定されているため，本番環境で利用することはできません。

1.1.5　開発環境の簡易構築

Developer では，GUI を利用して開発環境およびデバッグ環境を構築できます。GUI を利用できるため，定義ファイルやコマンドを使用して構築する場合に比べて，短時間で環境を構築できるようになります。

GUI を利用して構築できる内容は次のとおりです。

- デバッグ環境の構築
- Eclipse のインストールおよびセットアップ

また，構築した環境の設定変更およびアンセットアップも GUI を利用して実行できます。

1.2　Developer で提供する機能

Developer では，J2EE アプリケーションを開発するための機能を提供しています。これらの機能を組み合わせて使うことで，効率の良い J2EE アプリケーション開発ができます。

ここでは，Developer が提供する次の機能について説明します。

- Eclipse を使用した J2EE アプリケーションの開発で使用する機能
- デバッグ環境と Eclipse 環境をセットアップする機能
- ユーザ拡張性能解析トレースを使用した J2EE アプリケーションのテストを支援する機能

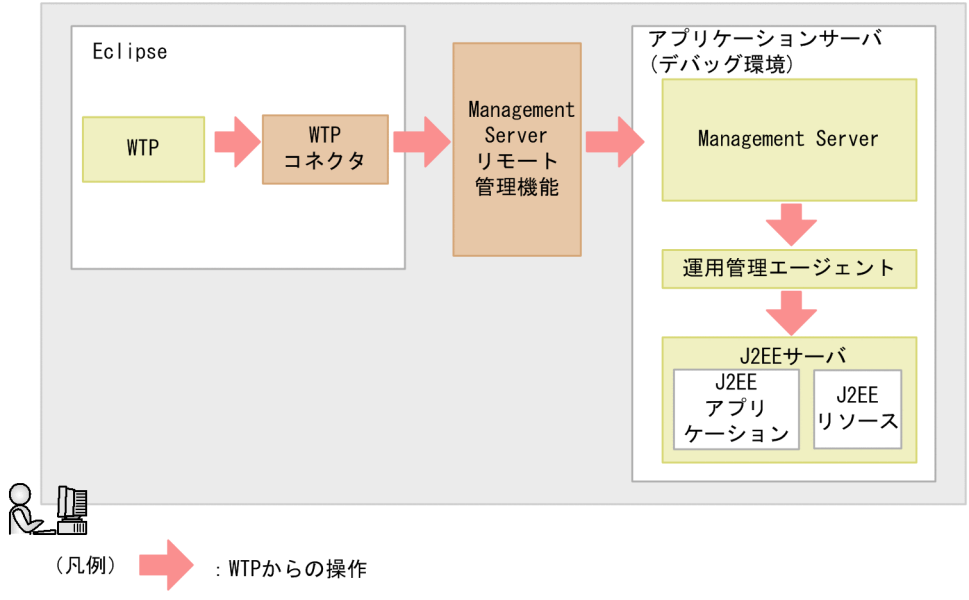
1.2.1　Eclipse を使用した J2EE アプリケーションの開発で使用する機能

Developer では，Eclipse の WTP（Eclipse Web Tools Platform）を使用して J2EE アプリケーションを開発するために次の機能を提供しています。

- WTP コネクタ
- Management Server リモート管理機能

それぞれの機能の関係を次の図に示します。

図 1-3　Eclipse を使用した J2EE アプリケーションの開発で使用する機能の関係



Developer では，WTP で開発した J2EE アプリケーションを WTP からデバッグ環境の J2EE サーバにデプロイしたり，J2EE サーバを起動したりできます。

WTP で開発した J2EE アプリケーションを操作するには，WTP コネクタおよび Management Server リモート管理機能を使用します。

WTP コネクタ

WTP コネクタは Eclipse のプラグインです。WTP コネクタは，WTP から J2EE サーバを起動したり，J2EE サーバに J2EE アプリケーションをデプロイしたりするときに使用します。

Management Server リモート管理機能

WTP コネクタから J2EE サーバにアクセスできるようにするためのライブラリです。

WTP からの操作は，Management Server リモート管理機能に要求として渡されます。Management Server リモート管理機能が受け取った要求は，さらに Management Server に渡され，J2EE サーバに対して要求された操作が実行されます。WTP から J2EE サーバを操作するためには，Management Server リモート管理機能の設定が必要になります。

WTP コネクタおよび Management Server リモート管理機能は，WTP からの要求を受け取り，デバッグ環境の Management Server に処理を渡します。Management Server は，運用管理エージェントを通して，J2EE アプリケーションをデプロイしたり，J2EE サーバを起動したりします。

1.2.2 デバッグ環境と Eclipse 環境をセットアップする機能

Developer では，環境構築のための機能として，次に示す二つの機能を提供しています。

- **開発環境インスタントセットアップ機能**

  J2EE アプリケーションをデバッグするための環境を GUI 上で設定し，構築できます。開発環境インスタントセットアップ機能では，次の環境をセットアップします。

  - J2EE サーバ
  - パフォーマンストレーサ
  - 組み込みデータベース
  - リソースアダプタ
  - Management Server

- **Eclipse セットアップ機能**

  Eclipse のインストールおよびセットアップを GUI 上で実施できます。

1.2.3 ユーザ拡張性能解析トレースを使用した J2EE アプリケーションのテスト支援機能

Developer では，ユーザ拡張性能解析トレースを利用して J2EE アプリケーションの性能解析を実施するために，次の機能を提供しています。

- **ユーザ拡張性能解析トレースのトレース取得ポイントの編集**

ユーザ拡張性能解析トレースが出力されるポイントをトレース取得ポイントといいます。Developerでは，Eclipse 上でソースコードを見ながら，トレース取得ポイントを設定したり，削除したりできます。

- **ユーザ拡張性能解析トレース設定ファイルのインポート・エクスポート**

  既存のユーザ拡張性能解析トレース設定ファイルのトレース取得ポイントを Eclipse に取り込んだり（インポート），Eclipse 上で設定したトレース取得ポイントをユーザ拡張性能解析トレース設定ファイルへ出力したり（エクスポート）できます。

1.3　開発環境のマシン構成

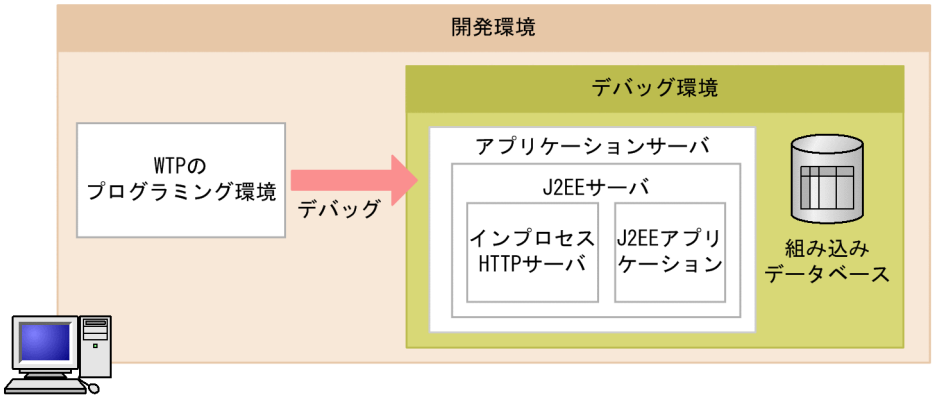
ここでは，J2EE アプリケーションを，1 台のマシンで開発・テストする場合の構成と，異なるマシンで開発・テストする場合の構成について説明します。

なお，このマニュアルでは 1 台のマシンで開発およびテストをする場合の手順を中心に説明します。

1.3.1　1 台のマシンで開発・テストする場合の構成

J2EE アプリケーションのプログラミング環境とデバッグ環境を 1 台のマシン上に構築する場合の構成は次のとおりです。

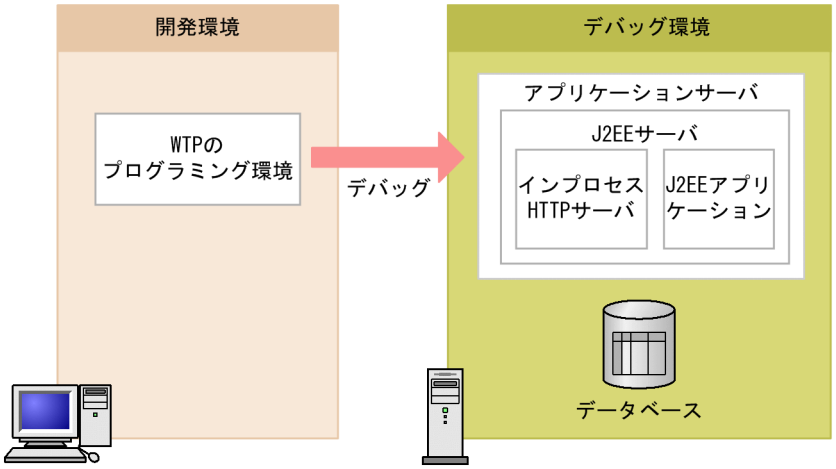
図 1-4　プログラミング環境とデバッグ環境を 1 台のマシンに構築する場合の構成



1.3.2　異なるマシンで開発・テストする場合の構成

J2EE アプリケーションを開発する環境とテストする環境を異なるマシンに構築する場合は，次のような構成になります。

図 1-5　プログラミング環境とデバッグ環境を異なるマシンに構築する場合の構成



なお，デバッグ環境には，アプリケーションサーバをインストールします。アプリケーションサーバの構成ソフトウェアについては，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」の「2.2　構成ソフトウェア」を参照してください。

注意事項

ホスト名が解決できないネットワーク環境で，プログラミング環境とデバッグ環境を異なるマシンに構築する場合は，次のどれかの方法で環境を構築してください。

- 簡易構築定義ファイルの<host-name>に IP アドレスを指定する
- 運用管理ポータルの「ホスト名」に IP アドレスを指定する
- hosts ファイルなどで，ホスト名が解決できるように設定する

ホスト名が解決できないネットワーク環境で J2EE サーバをデバッグしても，デバッグ接続できません。J2EE サーバがサスペンド状態になる場合があります。この場合は，次のどちらかの方法で対処してください。

1. J2EE サーバの起動監視時間（デフォルトは 10 分）を指定している場合（推奨設定）
   Eclipse 上では操作しないで，Manager が J2EE サーバを停止するのを待ってください。
2. J2EE サーバの起動監視時間（デフォルトは 10 分）を指定していない場合
   タスクマネージャから J2EE サーバプロセスを終了してください。タスクマネージャで，起動中の J2EE サーバプロセス（イメージ名：cjstartsv.exe）を終了してください。このとき，ほかの J2EE サーバプロセスを停止しないように注意してください。

1.4　Developer で開発する J2EE アプリケーションの形式

アプリケーションサーバで実行できる J2EE アプリケーションの形式には，展開ディレクトリ形式とアーカイブ形式があります。Eclipse を使用してアプリケーションを開発する場合は，デバッグ環境がローカルホストにあるか，またはリモートホストにあるかで，J2EE アプリケーションの形式が決まります。それぞれの場合の J2EE アプリケーションの形式を次に示します。

デバッグ環境がローカルホストにある場合

　展開ディレクトリ形式

デバッグ環境がリモートホストにある場合

　アーカイブ形式

ここでは，展開ディレクトリ形式およびアーカイブ形式それぞれの J2EE アプリケーションの概要について説明します。

参考

展開ディレクトリ形式を利用すると，修正したソースファイルを自動検知して更新するリロード機能を使用できます。リロード機能を使用すると，修正したソースファイルを，少ない手順で動的に入れ替えられるようになるため，作業効率が向上します。このマニュアルで説明する J2EE アプリケーションの開発手順では，展開ディレクトリ形式で開発することを前提としています。

1.4.1　展開ディレクトリ形式の J2EE アプリケーション

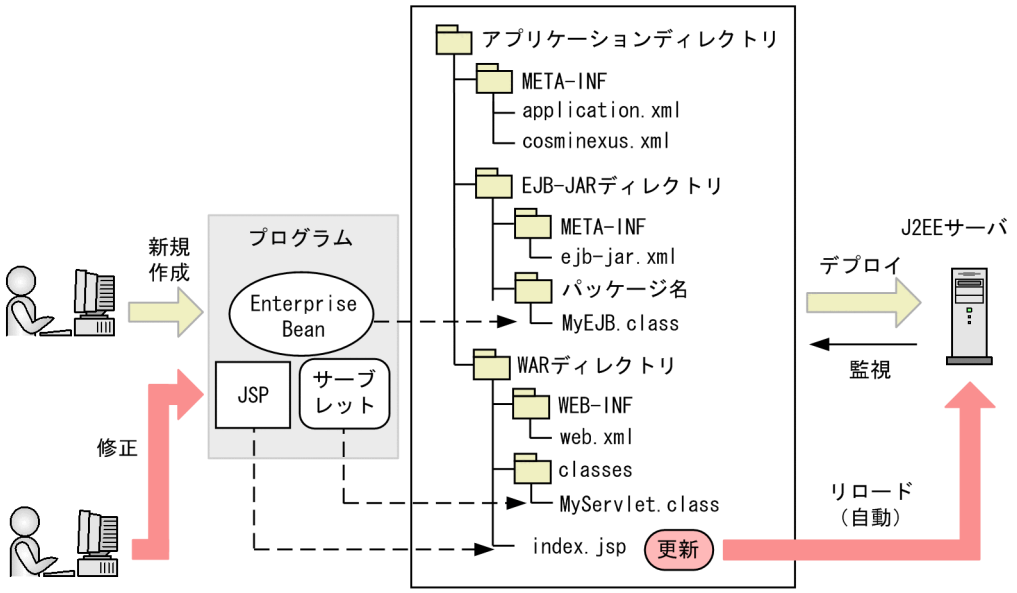
ここでは，展開ディレクトリ形式の J2EE アプリケーションの概要および構成を説明します。

(1)　展開ディレクトリ形式の J2EE アプリケーションの概要

展開ディレクトリ形式の J2EE アプリケーションとは，EJB やサーブレットなどの J2EE アプリケーションの実体を，J2EE サーバの外部に持つ J2EE アプリケーションです。

次の図に展開ディレクトリ形式の J2EE アプリケーションの概要を示します。

図 1-6　展開ディレクトリ形式の J2EE アプリケーションの概要



（凡例）

－－►　：アプリケーションディレクトリの展開先を示す。

展開ディレクトリ形式の J2EE アプリケーションを実行するためには，**アプリケーションディレクトリ**と呼ばれる，ある一定のルールに従ったディレクトリとファイルの構造に，作成したプログラムを含め，そのままデプロイします。

展開ディレクトリ形式の場合，デプロイした J2EE アプリケーションを J2EE サーバに監視させて，ファイルを更新したときなどに自動でリロードさせることができます（**リロード機能**）。このとき，J2EE アプリケーションの停止や削除，インポートなどの作業は不要です。

また，展開ディレクトリ形式の J2EE アプリケーションは，複数のクラスタで共有できます。J2EE アプリケーションの実体を J2EE サーバの外部に持ち，各クラスタから参照する形を取るため，プログラムを修正したときも，一回の更新作業ですべてのクラスタに反映できます。

なお，J2EE サーバにデプロイ済みの展開ディレクトリ形式の J2EE アプリケーションに対して，次に示す変更はできません。

- EJB-JAR および WAR の追加
- EJB-JAR および WAR の削除
- ライブラリ JAR※の追加
- ライブラリ JAR※の削除

注※

J2EE アプリケーションの DD（application.xml）の<module>タグ以下に定義されているファイル以外で，拡張子が小文字の JAR ファイル（.jar）です。J2EE アプリケーション内の J2EE コンポーネントから共通に使用できます。

これらの変更をする場合，一度 J2EE サーバから J2EE アプリケーションを削除してから，再度デプロイする必要があります。

(2) 展開ディレクトリ形式の J2EE アプリケーションの構成

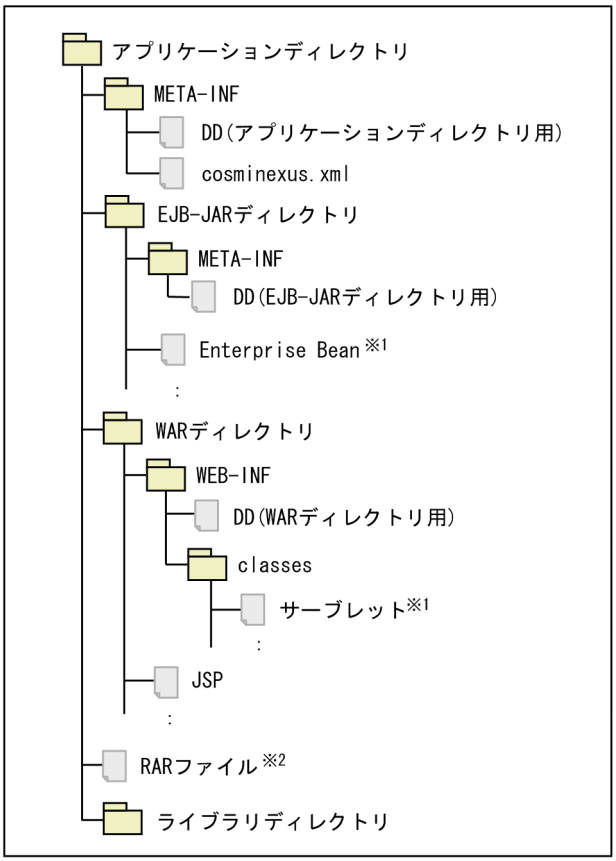
展開ディレクトリ形式では，アプリケーションディレクトリを作成し，アプリケーションディレクトリ以下に EJB-JAR ディレクトリと WAR ディレクトリを含めます。

ここでは，展開ディレクトリ形式でデプロイした場合の J2EE アプリケーションの構成を説明します。

なお，Eclipse を使用して J2EE アプリケーションを開発する場合は，Eclipse が自動的に展開ディレクトリ形式で作成するので，構成を意識する必要はありません。

アプリケーションディレクトリの構成例を次の図に示します。

図 1-7　アプリケーションディレクトリの構成例



注※1 コンパイル済みのクラスファイルを格納します。
注※2 リソースアダプタをJ2EEアプリケーションに含めて使用する場合だけ格納します。

アプリケーションディレクトリに含まれる EJB-JAR ディレクトリ，および WAR ディレクトリの構成を示します。

- EJB-JAR ディレクトリ

  EJB-JAR ディレクトリには，次のファイルなどが含まれます。

  - Enterprise Bean

- DD（EJB-JAR ディレクトリ用）

- WAR ディレクトリ

  WAR ディレクトリには，次のファイルなどが含まれます。

  - JSP
  - サーブレット
  - DD（WAR ディレクトリ用）

1.4.2　アーカイブ形式の J2EE アプリケーション

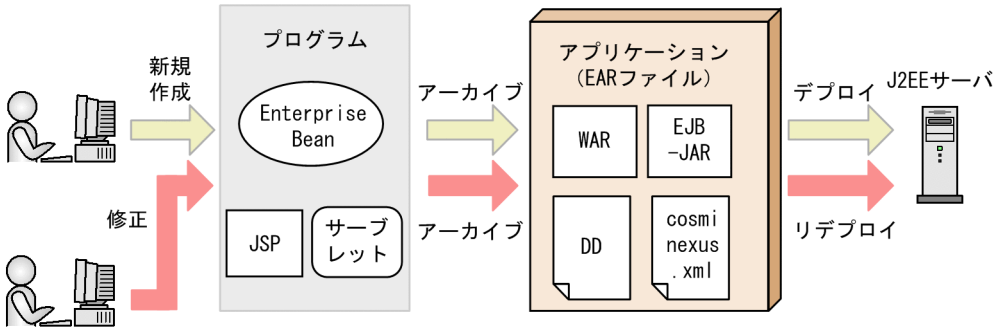
ここでは，アーカイブ形式の J2EE アプリケーションの概要および構成を説明します。

(1)　アーカイブ形式の J2EE アプリケーションの概要

アーカイブ形式の J2EE アプリケーションとは，EJB やサーブレットなどの J2EE アプリケーションの実体を J2EE サーバの作業ディレクトリに持つ J2EE アプリケーションです。

次の図にアーカイブ形式の J2EE アプリケーションの概要を示します。

図 1-8　アーカイブ形式の J2EE アプリケーションの概要



アーカイブ形式の J2EE アプリケーションを実行するには，作成したプログラムを EJB-JAR ファイルおよび WAR ファイルにアーカイブし，さらに EJB-JAR ファイルおよび WAR ファイルを EAR ファイルにアーカイブしてから，J2EE サーバにデプロイします。

プログラムを修正した場合は，アーカイブとデプロイをやり直して J2EE アプリケーションを入れ替えます。なお，リデプロイ機能を使用すると，J2EE アプリケーションが持つすべての属性情報を引き継いで，J2EE アプリケーションを入れ替えられます。

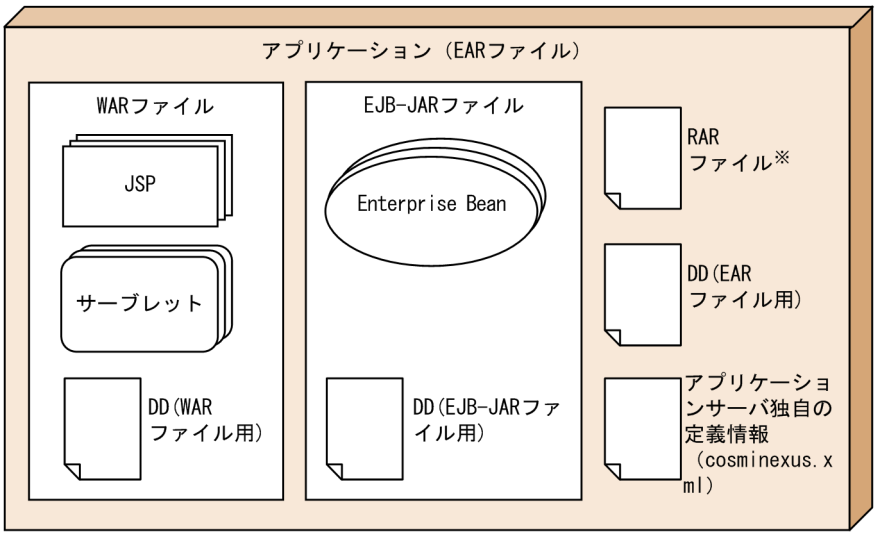
(2)　アーカイブ形式の J2EE アプリケーションの構成

アーカイブ形式では，J2EE アプリケーションを EAR ファイルとして作成し，EAR ファイルに WAR ファイルおよび EJB-JAR ファイルを含めます。

ここでは，アーカイブ形式でデプロイした場合の J2EE アプリケーションの構成を説明します。

EAR ファイルの構成例を次の図に示します。

図 1-9　EAR ファイルの構成例



注※　リソースアダプタをJ2EEアプリケーションに含めて使用する場合だけ格納します。

EAR ファイルに含まれる WAR ファイル，および EJB-JAR ファイルの構成を示します。

- WAR ファイル

  WAR ファイルには，次のファイルなどが含まれます。

  - JSP
  - サーブレット
  - DD（WAR ファイル用）

- EJB-JAR ファイル

  EJB-JAR ファイルには，次のファイルなどが含まれます。

  - Enterprise Bean
  - DD（EJB-JAR ファイル用）

  なお，Enterprise Bean は JavaBeans に実装することもできます。その場合，EJB-JAR ファイルは作成しません。

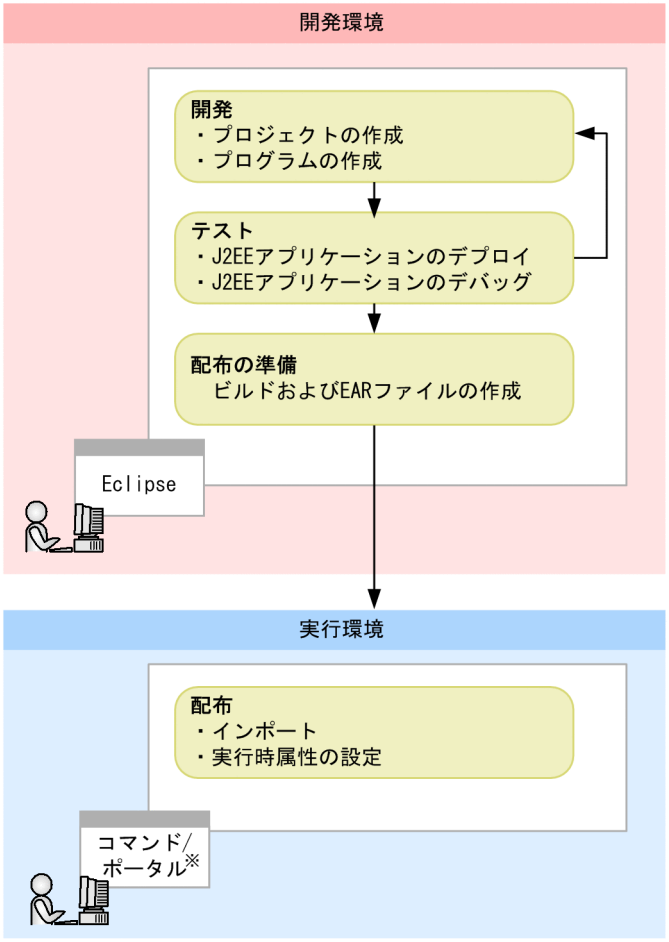
1.5　J2EE アプリケーション開発の流れ

Developer での J2EE アプリケーションの開発をスムーズに進めるために，J2EE アプリケーションの開発サイクルと，開発の手順を説明します。

1.5.1　J2EE アプリケーションの開発サイクル

J2EE アプリケーションの開発サイクルと，開発環境および実行環境で使用するツールを次の図に示します。

図 1-10　J2EE アプリケーションの開発サイクル



注※　サーバ管理コマンド，または運用管理ポータルを使用します。

開発環境では，Eclipse を使用して，J2EE アプリケーションの開発およびテストを実施します。J2EE アプリケーションのテストで問題があった場合は，J2EE アプリケーションを修正します。

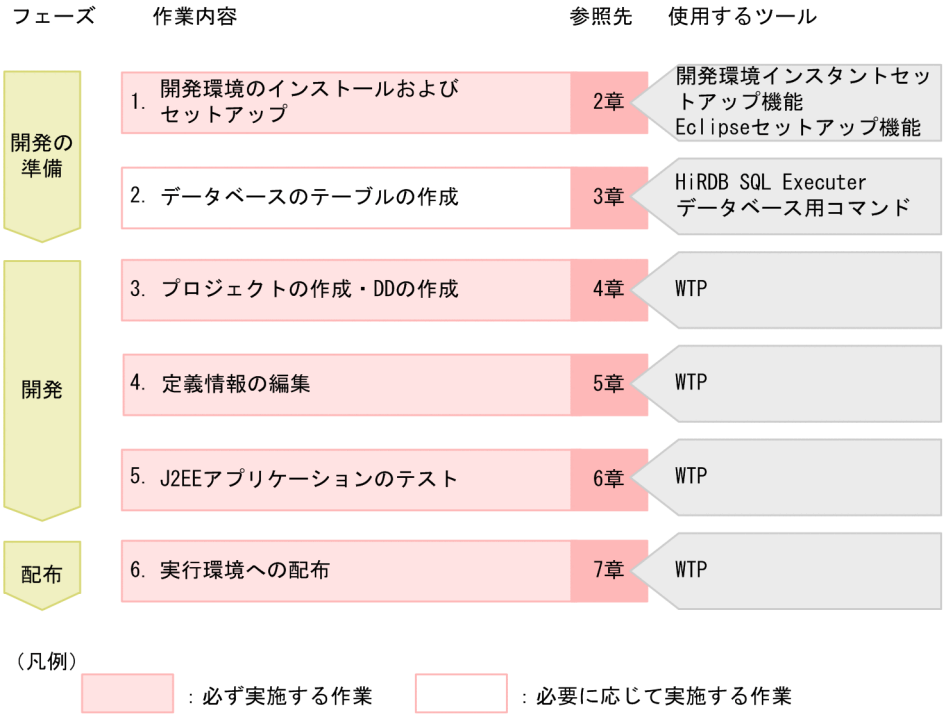
完成した J2EE アプリケーションは，サーバ管理コマンド，または運用管理ポータルを使用して実行環境に配布します。実行環境では，サーバ管理コマンドを使用して J2EE アプリケーションや J2EE リソースに実行時属性を設定します。

1.5.2　J2EE アプリケーションの開発手順

ここでは，J2EE アプリケーションの開発手順について説明します。J2EE アプリケーションの開発の流れ，参照先，および各工程で必要となるツールについて，次の図に示します。

このマニュアルで説明する J2EE アプリケーションは，テストで組み込みデータベースを利用するため，組み込みデータベースも構築します。

図 1-11　組み込みデータベースと EJB を利用した J2EE アプリケーション開発の流れ



各工程の概要を説明します。

1. 開発環境のインストールおよびセットアップ

J2EE アプリケーション開発に必要な環境を準備するために，次の作業を実施します。デバッグ環境のセットアップは，開発環境インスタントセットアップ機能および Eclipse セットアップ機能を使用して実行します。

- デバッグ環境のセットアップ
- Eclipse のインストールおよびセットアップ

これらインストールおよびセットアップを実行したあと，開発環境を設定する必要があります。開発環境のインストールおよびセットアップについては，「2.　インストールとセットアップ」を参照してください。

2. データベースのテーブルの作成

デバッグ環境で使用する組み込みデータベースのテーブルを作成します。組み込みデータベースのテーブルは，HiRDB SQL Executer を使用して作成します。組み込みデータベースのテーブルの作成については，「3.　デバッグ環境で使用するデータベースのテーブルの作成」を参照してください。

3. プロジェクトの作成・DD の作成

Eclipse のプロジェクトである，動的 Web プロジェクト，EJB プロジェクト，ユーティリティプロジェクト，エンタープライズアプリケーションプロジェクトを作成します。また，プロジェクトを作成するのと同時に，DD も作成します。各プロジェクトと DD の作成については，「4.　Eclipse を使用した J2EE アプリケーションの開発」を参照してください。

4. 定義情報の編集

作成する J2EE アプリケーションの内容に合わせて，定義情報を編集します。編集する定義情報には，DD の定義情報とアプリケーションサーバ独自の定義情報があります。定義情報の編集については，「5. 定義情報の編集」を参照してください。

5. J2EE アプリケーションのテスト

Eclipse の機能を使って，開発環境で J2EE アプリケーションのデプロイ，テスト，およびデバッグを実施します。J2EE アプリケーションのテストについては，「6.　J2EE アプリケーションのテスト」を参照してください。

6. 実行環境への配布

開発した J2EE アプリケーションを Eclipse でビルドして，EAR ファイルを生成します。生成した EAR ファイルは，サーバ管理コマンドまたは運用管理ポータルを利用して，実行環境へ配布します。J2EE アプリケーションの実行環境への配布については，「7.　実行環境への J2EE アプリケーションの配布」を参照してください。

1.6　そのほかのアプリケーションの開発

Developer では，J2EE アプリケーションのほかに Web サービスおよびバッチアプリケーションを開発できます。

それぞれの開発の特長を説明します。

Web サービスの開発

Developer では，Eclipse プラグインを使用して Web サービスを開発できます。Web サービスおよび Web サービスクライアントの開発方法については，「8.　Web サービスの開発」を参照してください。

バッチアプリケーションの開発

Developer では，Eclipse プラグインを使用してバッチアプリケーションを開発できます。バッチアプリケーションの開発方法については，「9.　バッチアプリケーションの開発」を参照してください。

1.7 Windows 使用時の注意事項

ここでは，Windows 環境で Developer を使用して J2EE アプリケーションを開発したり，テスト用のアプリケーションサーバを構築したりするときの注意事項について説明します。

1.7.1 管理者特権で実行する必要がある操作

管理者特権で実行する必要がある操作について説明します。なお，ここではシステムドライブを C ドライブとして説明します。

(1) プログラムのインストール

Developer や Eclipse などのプログラムを，C:¥Program Files 以下のディレクトリにインストールする場合，管理者特権でインストールする必要があります。インストール時に，［UAC］ダイアログが表示される場合は，［UAC］ダイアログで管理者アカウントのパスワードを入力してください。

(2) 開発環境インスタントセットアップ機能および Eclipse セットアップ機能の実行

開発環境インスタントセットアップ機能および Eclipse セットアップ機能は，必ず管理者特権で実行してください。管理者特権がない場合，開発環境インスタントセットアップ機能および Eclipse セットアップ機能を起動できません。

管理者特権で実行する場合は，次に示すスタートメニューのショートカットを右クリックして，［管理者として実行］を選択してください。

開発環境インスタントセットアップ機能

- ［Cosminexus］－［デバッグ環境セットアップ］
- ［Cosminexus］－［デバッグ環境アンセットアップ］
- ［Cosminexus］－［デバッグ環境設定変更］

Eclipse セットアップ機能

- ［Cosminexus］－［Eclipse セットアップ］
- ［Cosminexus］－［Eclipse アンセットアップ］

(3) アプリケーションサーバが提供するコマンドの使用

アプリケーションサーバが提供するコマンドは，管理者特権で実行する必要があります。アプリケーションサーバが提供するコマンドは，「管理者：コマンドプロンプト」で実行してください。

「管理者：コマンドプロンプト」は，OS が提供する機能を使用して起動してください。

(4) アプリケーションサーバが提供する定義ファイルの更新

アプリケーションサーバが提供する定義ファイルは，管理者特権で更新する必要があります。管理者特権のないユーザが定義ファイルを更新しても，C:¥Program Files 以下のディレクトリにある定義ファイルは更新されません。管理者特権のないユーザが更新したファイルは，次に示すディレクトリ以下に保存されます。

```
C:¥Users¥<ユーザ名>¥AppData¥Local¥VirtualStore
```

なお，アプリケーションサーバは管理者特権で起動されるため，管理者特権のないユーザが更新した定義ファイルの内容は無視されます。

(5) 制限されたフォルダに対する Eclipse からの操作

C:¥Program Files 以下のディレクトリまたは C:¥windows 以下のディレクトリに対する操作には管理者特権が必要です。これらの制限されたディレクトリに対して，管理者特権のないユーザがファイルをエクスポートしたり，Eclipse のワークスペースを保存したりすると，リダイレクション機能によって，操作対象のファイルはリダイレクトされ，次の場所に格納されます。

```
C:¥Users¥<ユーザ名>¥AppData¥Local¥VirtualStore¥Program Files
```

Eclipse 上からは，C:¥Program Files 以下に実ファイルがあるかのように，ファイルを読み込んだり書き込んだりできますが，エクスポートまたは保存したファイルの実体は上記の場所に格納されます。

1.7.2 JIS X0213:2004 に含まれる Unicode の補助文字を使用する場合の注意事項

JIS X0213:2004 の第三水準および第四水準の文字の一部には，Unicode の補助文字が含まれます。Unicode の補助文字とは，基本多言語面以外の文字（Unicode のコードポイントが U+10000～U+10FFFF の範囲の文字）のことです。UTF-16 エンコーディングでは，サロゲートペアとして表されます。

Unicode の補助文字を使用する場合の注意事項を次に示します。

(1) リクエストで使用する場合の注意事項

アプリケーションサーバに対して，クライアントから Unicode の補助文字を含むリクエストを送信した場合，Unicode の補助文字は，ログや PRF トレースに正しく出力されません。ただし，その場合も，Unicode の補助文字以外の文字は，ログや PRF トレースに正しく出力されます。

また，リクエストに Unicode の補助文字が含まれる場合も，J2EE アプリケーションは正しく動作します。

リクエストでの Unicode の補助文字の使用を制限したい場合には，J2EE アプリケーションでの対応などを検討してください。

(2) 環境構築／運用時の注意事項

アプリケーションサーバを構築，運用する場合，および J2EE アプリケーションやリソースアダプタをデプロイする場合に使用する定義に，Unicode の補助文字は使用できません。

Unicode の補助文字を使用できない定義の例を示します。

- EAR，WAR，JAR，EJB-JAR，サーブレット，JSP，クラス，メソッド，引数，または変数の名称
- DD 内の各種定義
- システムのインストール先の指定値
- 各種定義ファイルの設定値
- そのほか，Eclipse で扱うファイルおよびディレクトリ全般
  （Eclipse および Developer インストール先，デプロイディレクトリなど）

1.8　マニュアルの記載に関する注意事項

フォルダとパスの表記について

このマニュアルでは，IDE や Java コマンドでの表示に従い，フォルダ／ディレクトリの表現，およびパスの表記（「¥」または「/」）を変更している場合があります。ご使用の環境に合わせて，フォルダ／ディレクトリの表現，およびパスの表記を置き換えてお読みください。

Eclipse に関するサポートについて

Developer が提供する Eclipse のプラグイン機能以外の Eclipse に関するサポートはしていません。Eclipse の使用方法や Eclipse が表示するエラーの対処方法については，ユーザで調査，対処してください。

このマニュアルで記載している操作以外の操作について

このマニュアルで記載している操作以外の操作（Eclipse 上の操作など）をした場合，動作の保証はできません。

ランゲージパックの使用について

このマニュアルでは，Eclipse Babel Project が提供する BABEL 日本語ランゲージパックを適用した Eclipse メニュー表記を使用しています。使用するランゲージパックのバージョンによっては，マニュアル内の表記と異なる場合があります。

Windows のメニュー名の表記について

このマニュアルに記載している Windows のメニュー名の表記は，次の OS を前提としています。

- Windows 10

2 インストールとセットアップ

この章では，Eclipse を使用した J2EE アプリケーション開発で使用するプログラムのインストール，およびデバッグ環境と Eclipse のセットアップの手順について説明します。また，プログラムのアンインストール，およびデバッグ環境と Eclipse のアンセットアップの手順についても説明します。

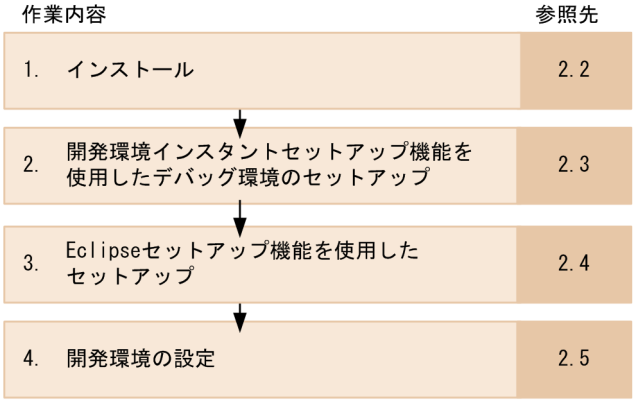
なお，この章では，開発環境インスタントセットアップ機能および Eclipse セットアップ機能を使用したセットアップ手順を中心に説明します。これらの機能を使用しない場合は，「付録 B　開発環境インスタントセットアップ機能および Eclipse セットアップ機能を使用しないデバッグ環境の構築手順」を参照してください。

2.1　インストールとセットアップの流れ

J2EE アプリケーション開発環境で使用するプログラムをインストールおよびセットアップします。

プログラムのインストールと開発環境インスタントセットアップ機能および Eclipse セットアップ機能を使用したセットアップの流れを次に示します。

図 2-1　インストールとセットアップの流れ



それぞれの作業の概要を説明します。

1. インストール

   インストーラを使用して，Developer をインストールします。また，セットアップに必要な環境変数の設定をします。詳細は，「2.2　インストール」を参照してください。

2. 開発環境インスタントセットアップ機能を使用したデバッグ環境のセットアップ

   デバッグ環境のセットアップをします。デバッグ環境のセットアップについては，「2.3　開発環境インスタントセットアップ機能を使用したデバッグ環境のセットアップ」を参照してください。

3. Eclipse セットアップ機能を使用したセットアップ

   Eclipse のセットアップをします。Eclipse のセットアップについては，「2.4　Eclipse セットアップ機能を使用したセットアップ」を参照してください。

4. 開発環境の設定

   Eclipse を使用して J2EE アプリケーションを開発するための設定をします。詳細は，「2.5　Eclipse の設定」を参照してください。

以降の節で，これらの手順について説明します。

また，この章ではセットアップした環境の設定変更とアンセットアップ，およびプログラムのアンインストールについても説明します。それぞれの手順については，次の個所を参照してください。

- Eclipse 使用時に必要となる設定を変更したい場合
  「2.6　Eclipse の設定変更（任意の作業）」
- デバッグ環境の設定を変更したい場合

「2.7　デバッグ環境の設定変更」

- デバッグ環境をアンセットアップしたい場合
  「2.8.1　デバッグ環境のアンセットアップ」
- Eclipse をアンセットアップしたい場合
  「2.8.2　Eclipse 環境のアンセットアップ」
- Developer をアンインストールしたい場合
  「2.9　Developer のアンインストール」

セットアップに関する注意事項

開発環境インスタントセットアップ機能を実行中に Eclipse セットアップ機能を実行（または Eclipse セットアップ機能を実行中に開発環境インスタントセットアップ機能を実行）すると，処理のステータスが未実行のまま終了することがあります。

この場合，先に実行していたセットアップの処理が完了してから，次のセットアップを再度実行してください。なお，再度実行する前に，環境をアンセットアップする必要はありません。

2.2 インストール

ここでは，Developer のインストールについて説明します。

2.2.1 Developer のインストール

Developer のインストールには，インストーラを使用します。

インストールするには，Administrator 権限または管理者特権が必要です。なお，Developer のインストール手順については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「2.2.2 Application Server を新規インストールする（Windows の場合)」を参照してください。

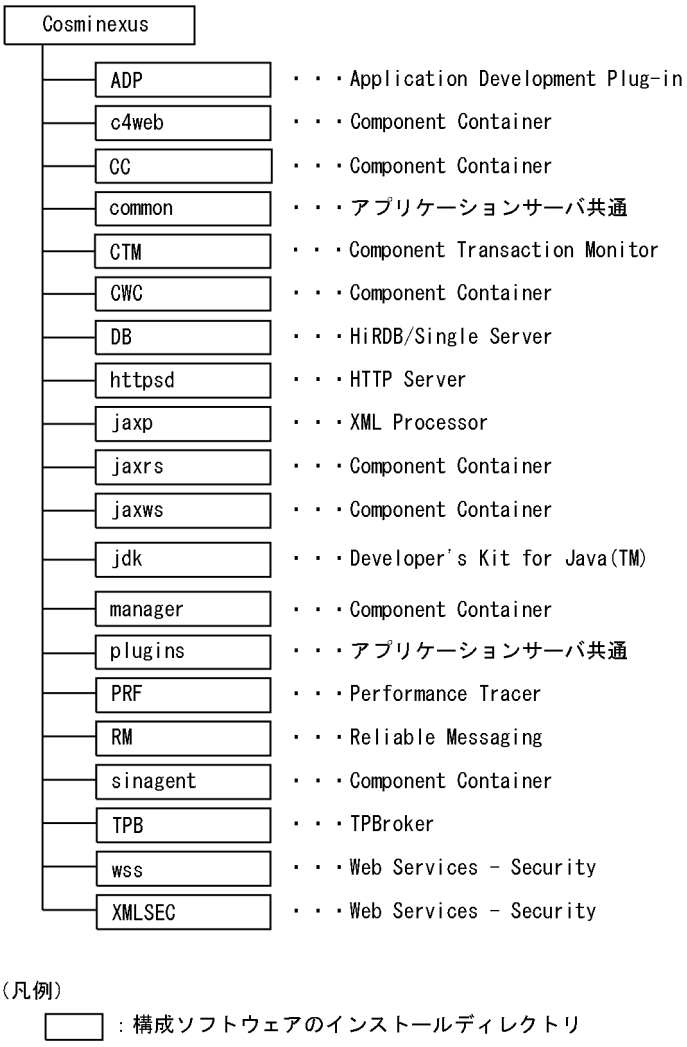
なお，インストール手順の製品名「Application Server」は，「Developer」と読み替えてください。

ここでは，インストール後のディレクトリ構成，環境変数の設定，および Developer を再インストールするときの注意事項について説明します。

(1) インストール後のディレクトリ構成

Developer を<システムドライブ>:¥Program Files¥Hitachi の下にインストールすると「Cosminexus」というディレクトリが作成されます。「Cosminexus」ディレクトリ以下の構成を次の図に示します。

図 2-2　Developer のディレクトリ構成



(2)　環境変数の設定

Developer では，次に示す環境変数をシステムの環境変数に設定する必要があります。

環境変数 TZ

JST-9 など

> **参考**
>
> 環境変数は，Windows の［コントロールパネル］の中の［システム］－［システムの詳細設定］を選択し，［詳細設定］タブの中の［環境変数］ボタンをクリックすることで設定します。環境変数が正しく設定されていない場合はエラーが発生します。

環境変数の設定が終わったら，OS を再起動してください。

(3) Developer をインストールするときの注意事項

ここでは，Developer をインストールするときの注意事項と必要な作業について説明します。

開発環境とデバッグ環境を異なるマシンに構築する場合，開発環境にインストールする JDK のバージョンは，デバッグ環境の JDK のバージョンに合わせてください。

JDK のバージョンを変更する場合，アプリケーションサーバをアンインストールしてから再インストールしてください。

(4) Developer を再インストールするときの注意事項

ここでは，Developer を再インストールするときの注意事項と必要な作業について説明します。

インストールを実行する前に，構成ソフトウェアのすべてのプロセスが実行中でないことを確認してください。実行中の場合は，構成ソフトウェアのすべてのプロセスを停止してから，インストールしてください。

開発環境インスタントセットアップ機能でデバッグ環境をセットアップしたあとで Developer を再インストールする場合は，開発環境インスタントセットアップ機能を使用して事前にデバッグ環境をアンセットアップしておいてください。また，Eclipse もアンセットアップしておいてください。事前にアンセットアップしなかった場合は，Developer の再インストール後に手動で Management Server，J2EE サーバ，パフォーマンストレーサ，および組み込みデータベースを削除してください。

2.3　開発環境インスタントセットアップ機能を使用したデバッグ環境のセットアップ

開発環境インスタントセットアップ機能を使用して，デバッグ環境をセットアップします。開発環境インスタントセットアップ機能では，Smart Composer 機能を使用してデバッグ環境を構築します。ここでは，開発環境インスタントセットアップ機能について，次の内容を説明します。

- 前提条件
- セットアップ実行時の注意事項
- セットアップする環境の設定内容
- セットアップ手順

> **参考**
>
> **開発環境インスタントセットアップ機能を使用しない場合**
>
> Developer では，Smart Composer 機能を直接使用してデバッグ環境を構築することもできます。また，論理 CTM の環境を使用するデバッグ環境を構築したい場合は，Smart Composer 機能を使用して構築してください。Smart Composer 機能については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「1.1.3　Smart Composer 機能とは」を参照してください。

2.3.1　開発環境インスタントセットアップ機能の前提条件

次の条件に当てはまる場合，開発環境インスタントセットアップ機能でデバッグ環境をセットアップできません。セットアップできない条件とその対策を次に示します。

表 2-1　開発環境インスタントセットアップ機能でセットアップできない条件とその対策

セットアップできない条件	対策
開発環境インスタントセットアップ機能以外の方法で Management Server をセットアップしている	Management Server をアンセットアップする
開発環境インスタントセットアップ機能以外の方法でホストやサーバを構築している	ホストおよびサーバを削除する
開発環境インスタントセットアップ機能以外の方法で組み込みデータベースをセットアップしている	組み込みデータベースをアンセットアップする
コマンド拡張機能が無効になっている	コマンド拡張機能を有効にする
次のサービスのスタートアップの種類が「無効」になっている • Management Server • Management Server - Administration Agent	該当するサービスのスタートアップの種類を「自動」または「手動」に設定する

セットアップできない条件	対策
• HiRDB/SingleServer_CS0	

2.3.2　開発環境インスタントセットアップ機能実行時の注意事項

開発環境インスタントセットアップ機能でデバッグ環境を構築する際の注意事項を説明します。

(1)　Windows セキュリティの重要な警告について

開発環境インスタントセットアップ機能の実行中に，［Windows セキュリティの重要な警告］ダイアログが表示された場合は，直ちに［ブロックを解除する］ボタンまたは［アクセスを許可する］ボタンをクリックする必要があります。

(2)　<Developer のインストールディレクトリ>¥ADP 以下のディレクトリおよびファイルについて

開発環境インスタントセットアップ機能の実行中に，<Developer のインストールディレクトリ>¥ADP 以下のディレクトリおよびファイルを削除または名称変更しないでください。

(3)　Management Server の管理ユーザの設定について

開発環境インスタントセットアップ機能でセットアップしたデバッグ環境を，ほかの PC（リモートホスト）から使用する場合は，Management Server の管理ユーザの設定で管理ユーザの認証を［認証あり］にしてください。［認証なし］にすると，デバッグ環境の使用はローカルホストだけに制限されます。

(4)　デバッグ環境を localhost 指定で構築した場合について

デバッグ環境を localhost 指定で構築すると，IP アドレスやホスト名を変更しても構築したデバッグ環境をそのまま利用できます。localhost 指定で構築する場合の注意事項を説明します。

(a)　ホスト名または IP アドレスの変更

PC（ローカルホスト）のホストコンピュータ名や IP アドレスを変更した場合，リモート管理機能との交信ができなくなります。リモート管理機能との接続情報をリセットするため，OS を再起動してください。

(b)　localhost 指定で構築した環境のカスタマイズ

localhost 指定で構築した環境に，運用管理ポータルなどでカスタマイズを加えた場合は，IP アドレスやホスト名を変更しないでください。

Eclipse のワークスペースや Developer で提供する以外のプラグインをコピーした場合の動作については，Developer ではサポートしていません。

(c)　localhost 固定設定

組み込みデータベースのホスト名を localhost とするため，localhost は必ずループバックアドレスに対応させておいてください。

Web システムのホスト名およびエージェントホスト名を localhost 固定で構築します。この場合，運用管理ポータル［論理サーバの環境設定］で J2EE サーバの設定読み込みを実施すると，デフォルト値で更新される項目があり，読み込み実施後に再設定する必要があります。詳細は，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」を参照してください。

(d)　標準のデバッグ環境への切り替え

標準のデバッグ環境への切り替えはできません。標準のデバッグ環境へ切り替える場合，デバッグ環境を再セットアップしてください。

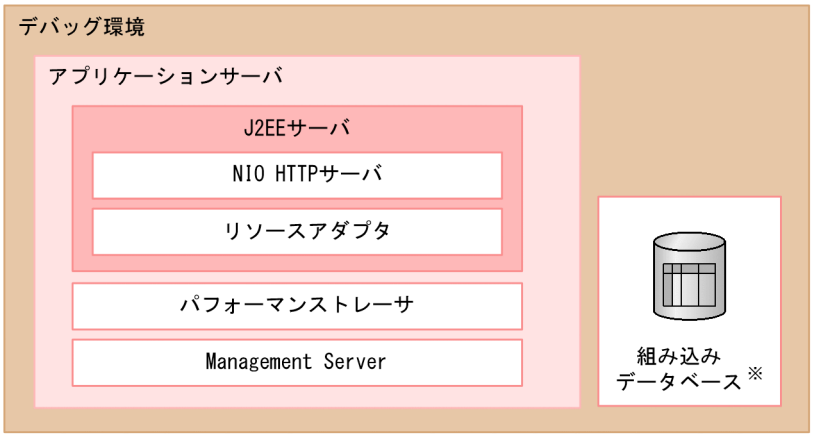
(5)　JDK17 をインストールした環境についての注意事項

JDK17 をインストールした環境では「V9 互換モード」は使用できません。そのため，カスタムセットアップの「J2EE サーバモード」の選択で，「V9 互換モード」を選択しないでください。JDK17 をインストールした環境で「V9 互換モード」を選択すると，構築中にエラーが発生し，セットアップに失敗します。

2.3.3　セットアップする環境の設定内容

開発環境インスタントセットアップ機能を使用すると，次の図に示すような環境をセットアップできます。

図 2-3　開発環境インスタントセットアップ機能でセットアップするシステム



注※
デバッグ環境の構築時に組み込みデータベースを構築するかどうかを指定できます。

開発環境インスタントセットアップ機能でセットアップできる環境は次のとおりです。

- J2EE サーバ※1
- パフォーマンストレーサ

- Management Server
- 組み込みデータベース※2

注※1

NIO HTTP サーバ，リソースアダプタを含みます。

注※2

組み込みデータベースとは，Developer が提供するデバッグ環境用のデータベースです。

組み込みデータベースは，実行環境で HiRDB を使用する J2EE アプリケーションをテストするときだけ使用できます。実行環境で Oracle などのデータベースを使用する場合は，デバッグ環境で組み込みデータベースを使用してテストできません。

また，組み込みデータベースの用途は開発時の利用に限定されているため，本番環境で利用することはできません。

組み込みデータベースへの接続で使用する JDBC ドライバーには，Developer で提供する JDBC ドライバーを使用します。なお，Developer で提供する JDBC ドライバーは Connector 1.5 に対応していません。

セットアップの実行方法は，標準セットアップとカスタムセットアップのどちらかを選択できます。標準セットアップを実行した場合は，表 2-2 に示す値で環境が構築されます。カスタムセットアップを実行した場合は，次の内容を任意の値に変更できます。

- J2EE サーバモード
- 組み込みデータベース構築ディレクトリ（設定ファイルおよび RD エリアのファイル用ディレクトリ）
- 組み込みデータベースのサイズ
- J2EE サーバのポート番号
- Management Server の管理ユーザの設定
- Management Server のポート番号
- 組み込みデータベースのポート番号

開発環境インスタントセットアップ機能がセットアップする環境の設定値を次の表に示します。構築したい環境に合わせて，標準セットアップまたはカスタムセットアップを選択してください。

表 2-2　開発環境インスタントセットアップ機能でセットアップする環境の設定値

環境	設定項目	標準セットアップで設定される値	カスタムセットアップでの変更
J2EE サーバ	ホスト名	InstantHost	×
	J2EE サーバ名（表示名）	InstantJ2EEServer	×
	J2EE サーバモード※9	推奨モード	○
	デバッグ接続のためのポート番号	3999※1	○

環境	設定項目	標準セットアップで設定される値	カスタムセットアップでの変更
	HTTP のポート番号	80	○
	内部通信用ポート番号	28008	○
	RMI レジストリのポート番号	23152	○
	ネーミングサービスのポート番号(インプロセスネーミングサービスのポート番号)	900	○
	構築できるサーバの数	1	×
	論理サーバ名	Smart Composer 機能が生成する論理サーバ名	×
	アクセスを許可するホストの定義	すべてのホストからのアクセスを許可する	×
	ライトトランザクション	有効	×
	JTA リカバリの固定ポート番号	20302	×
	リロード機能	リロード対象と設定されるファイルは次のとおりです。 • EJB-JAR • サーブレット • JSP	×
	リロードの更新検知のインターバル	1 秒	×
	セキュリティマネージャ	使用しない※2	×
	クラスパス	add.class.path=<Developer のインストールディレクトリ>¥DB¥CLIENT¥UTL¥<HiRDB Type4 JDBC Driver の JAR ファイル>	×
	JSP デバッグ	有効	×
	例外発生時のデフォルトエラーページへのスタックトレース出力	有効	×
Management Server	サーバ名	cosmi_m	×
	Eclipse コンソール出力の連携	連携する	×
	コンソールへの情報出力	出力する	×
	運用管理エージェントの起動	自動	×
	Management Server の起動	自動	×
	リモート管理機能の接続ポート番号	28099※3	○
	終了要求受信ポート番号	28005	○
	内部通信用ポート番号	28009	○

環境	設定項目	標準セットアップで設定される値	カスタムセットアップでの変更
	接続 HTTP ポート番号	28080	○
	運用管理エージェントのポート番号	20295	○
	管理ユーザの設定	認証なし	○
	管理ユーザ ID※4	−	○
	パスワード※4	−	○
	Management Server への接続を許可するホスト	localhost	×※5
組み込みデータベース	組み込みデータベース構築ディレクトリ（設定ファイルおよび RD エリアのファイル用ディレクトリ）	<Developer のインストールディレクトリ>¥ADP¥DB¥	×
	データベースサイズ	Small（500MB）	×
	認可識別子	USER1※6	×
	パスワード	＿※6	×
	ポート番号	22200	○
	開始・停止	ユーザの操作	×
ホスト	構築できるホストの数	1	×
	ホスト名	Smart Composer 機能が生成するホスト名	×
	表示名	InstantHost	×
パフォーマンストレーサ	構築できるサーバの数	1	×
	論理サーバ名	Smart Composer 機能が生成する論理サーバ名	×
	表示名	InstantPRF	×
リソースアダプタ	インポート	DBConnector_HiRDB_Type4_CP.rar をインポートする	×
	表示名※7	DB_Connector_for_Cosminexus_Driver	×
	ユーザ ID※7	USER1※8	×
	パスワード※7	＿※8	×
	description（コンフィグレーションプロパティ）※7	22200※8	×
	DBHostName（コンフィグレーションプロパティ）※7	localhost	×
	コネクション・プールの最小値※7	1	×

環境	設定項目	標準セットアップで設定される値	カスタムセットアップでの変更
	コネクション・プールの最大値※7	8	×
	デプロイ	デプロイする	×

（凡例）

○：カスタムセットアップで変更できます。

×：カスタムセットアップで変更できません。標準セットアップと同じ値が設定されます。

－：値は設定されません。空欄の状態です。

注※1

Eclipse のデバッグ構成で使用します。

開発環境インスタントセットアップ機能の標準セットアップでは，デバッグ接続のためのポート番号として 3999 を設定します。

注※2

J2EE サーバの起動オプションに「-nosecurity」を指定します。

注※3

Eclipse から Management Server に接続するために使用します。

注※4

開発環境インスタントセットアップ機能の設定変更以外での変更はしないでください。開発環境インスタントセットアップ機能の設定変更以外で変更すると，開発環境インスタントセットアップ機能の設定変更およびアンセットアップが実行できなくなります。

注※5

カスタムセットアップ時に Management Server 管理ユーザの設定を「認証あり」とした場合には「*」が設定され，すべてのホストからのアクセスが許可されます。

注※6

標準セットアップでも値を変更できます。

注※7

インポートしたリソースアダプタの Connector 属性として設定されます。

注※8

組み込みデータベースの設定値と同じ値が設定されます。

注※9

JDK17 をインストールした環境では「V9 互換モード」は使用できないため，選択しないでください。JDK17 をインストールした環境で「V9 互換モード」を選択した場合，セットアップは失敗します。詳細はマニュアル「アプリケーションサーバ 機能解説 互換編」の「2.2　インストール環境によって使用できるモード」，およびマニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「9.9.1　J2EE サーバの追加」を参照してください。

2.3.4　デバッグ環境の標準セットアップ

開発環境インスタントセットアップ機能の標準セットアップでは，決まった値でデバッグ環境をセットアップします。標準セットアップを実行するためには，次の条件を満たしている必要があります。

- 組み込みデータベース構築ディレクトリ（設定ファイルおよび RD エリアのファイル用ディレクトリ）

- <Developer のインストールディレクトリ>¥ADP¥DB があること。
- <Developer のインストールディレクトリ>のパスの長さが 50 バイト以内であること。
- <Developer のインストールディレクトリ>¥ADP¥DB 内にファイルがないこと。
- 次に示すディレクトリおよびファイルが<Developer のインストールディレクトリ>¥ADP¥DB の直下にないこと。
  ・area ディレクトリ，area ファイル
  ・bats ディレクトリ，bats ファイル
  ・conf ディレクトリ，conf ファイル
  ・ini ディレクトリ，ini ファイル
- ネットワークドライブを使用したパスを指定していないこと。

- **開発環境インスタントセットアップ機能で指定するポート番号**

  システムでほかに使用しているポート番号と重複していないこと。標準セットアップで使用するポート番号については，「2.3.3　セットアップする環境の設定内容」を参照してください。

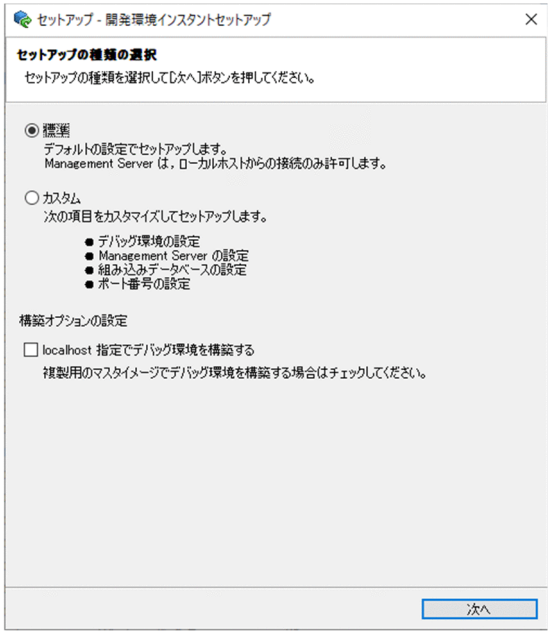
デバッグ環境の標準セットアップを実行する場合の手順を次に説明します。

1. **スタートメニューから，［Cosminexus］－［デバッグ環境セットアップ］※を選択します。**

   注※

   対象 OS のスタートメニューの表示仕様によっては，［Cosminexus］－［環境構築］－［デバッグ環境セットアップ］と表示される場合があります。

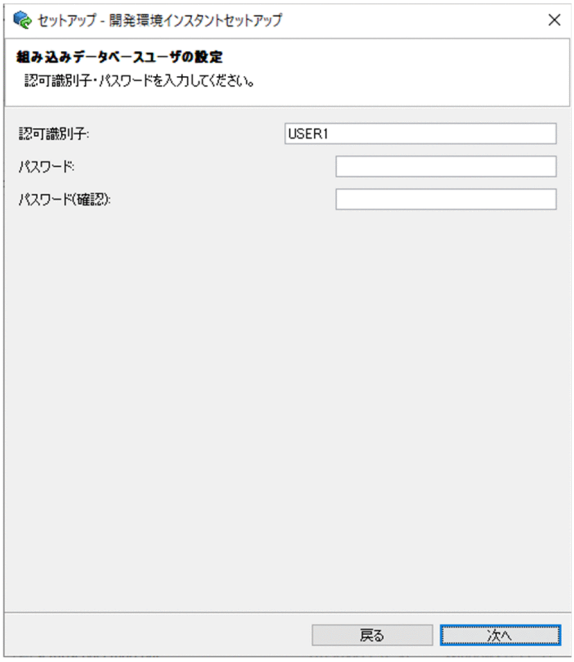
   開発環境インスタントセットアップ機能が起動して，［セットアップ - 開発環境インスタントセットアップ］ダイアログの［セットアップの種類の選択］ページが表示されます。



   「localhost 指定でデバッグ環境を構築する」チェックボックスをチェックした場合，複製用のマスターイメージとしてデバッグ環境を構築できます。必要に応じて選択してください。

2. [標準] を選択して，[次へ] ボタンをクリックします。

[組み込みデータベースユーザの設定] ページが表示されます。

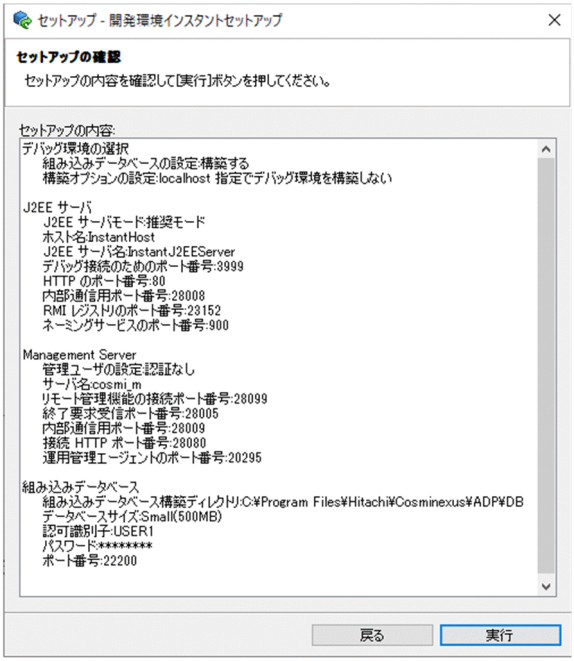


3. 次の項目を指定します。

項目名	指定値
認可識別子	組み込みデータベースを操作するときに使用する認可識別子を指定します。 8 バイト以内の半角英数字で指定してください。
パスワード	組み込みデータベースを操作するときに使用するパスワードを指定します。 半角英字で始まる 30 バイト以内の半角英数字で指定してください。
パスワード(確認)	[パスワード] に指定した値と同じ値を指定します。

4. [次へ] ボタンをクリックします。

[セットアップの確認] ページが表示されます。



[セットアップの内容]エリアにセットアップされる内容が表示されるので，設定値を確認してください。標準セットアップで構築される環境の設定内容の詳細は，「2.3.3　セットアップする環境の設定内容」を参照してください。

なお，組み込みデータベースユーザのパスワードについては，指定した値に関係なく 8 文字の「*（アスタリスク）」で表示されます。

5. [実行]ボタンをクリックします。

[進行状況]ページが表示されます。

セットアップが終了すると，[セットアップの完了]ページが表示されます。

6. [終了]ボタンをクリックします。

［セットアップ - 開発環境インスタントセットアップ］ダイアログが閉じます。

注意事項

処理が中断された場合の対処

処理が中断された場合は，アンセットアップを実行してから再度セットアップを実行してください。アンセットアップの手順については，「2.8.1　デバッグ環境のアンセットアップ」を参照してください。

エラーによって中断された場合は，コンソールに出力されたエラーメッセージを確認してエラーを取り除いてから，アンセットアップしてください。そのあと，再度セットアップを実行してください。

出力されたエラーメッセージについては，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」を参照して対処してください。

組み込みデータベースのセットアップ時にエラーが発生した場合は，マニュアル「HiRDB メッセージ」を参照してください。

セットアップした環境を変更する場合の注意

開発環境インスタントセットアップ機能で変更できない設定を変更したい場合は，セットアップが完了したあとで，運用管理ポータルを使用して変更してください。ただし，運用管理ポータルで設定変更した J2EE サーバに対して，再度開発環境インスタントセットアップ機能で設定を変更しようとすると，エラーが発生するおそれがあります。

運用管理ポータルについてはマニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」を参照してください。

参考

セットアップ後の各サーバの状態

開発環境インスタントセットアップ機能を使用してセットアップすると，各サーバは次の状態になります。

表 2-3　セットアップ後の各サーバの状態

サーバ名	状態
運用管理エージェント	○
Management Server	○
J2EE サーバ	×
パフォーマンストレーサ	○※
組み込みデータベース	○

（凡例）

○：開始　×：停止

注※

localhost 指定でデバッグ環境を構築した場合は，停止状態となります。

セットアップログの確認方法

開発環境インスタントセットアップ機能を実行したときのログは，デバッグ環境のセットアップログに出力されます。セットアップログの確認方法については「付録 E.4　開発環境インスタントセットアップ機能および Eclipse セットアップ機能実行時の情報の採取」を参照してください。

2.3.5　デバッグ環境のカスタムセットアップ

カスタムセットアップでは，次の内容を指定して，デバッグ環境を構築できます。

- J2EE サーバモード
- Management Server の管理ユーザの設定
- 組み込みデータベースの構築の有無
- 組み込みデータベース構築ディレクトリ（設定ファイルおよび RD エリアのファイル用ディレクトリ）
- 組み込みデータベースのサイズ
- J2EE サーバのポート番号
- Management Server のポート番号
- 組み込みデータベースのポート番号

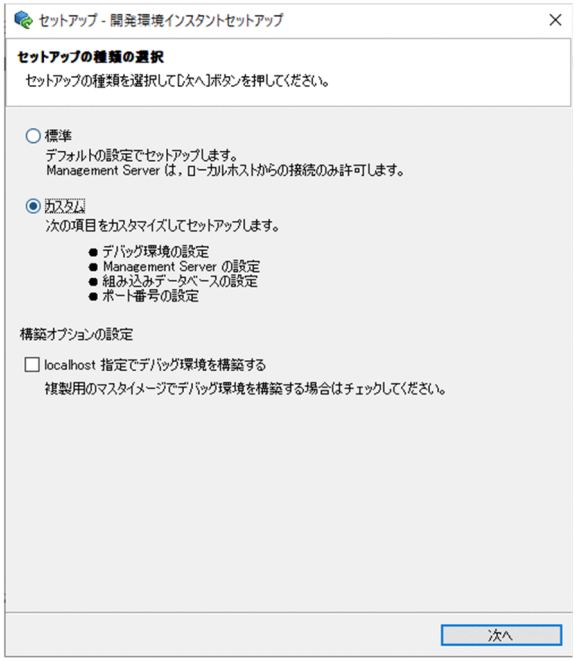
ここでは，カスタムセットアップを実行する場合の手順を説明します。

1. **スタートメニューから，［Cosminexus］－［デバッグ環境セットアップ］※を選択します。**

注※

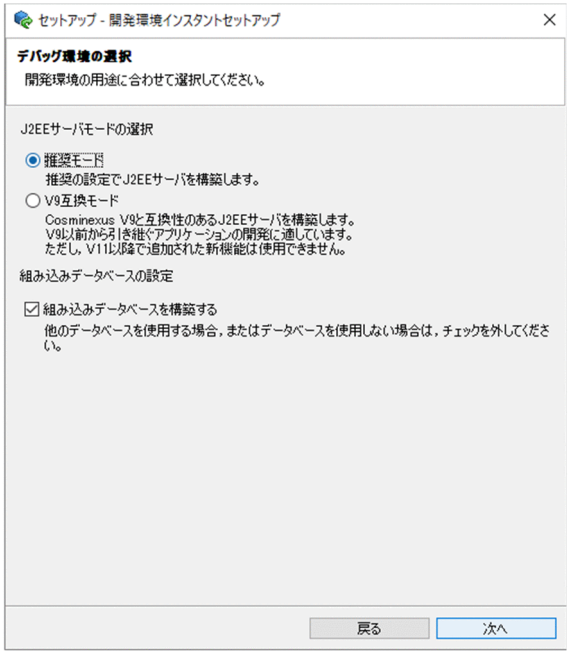
対象 OS のスタートメニューの表示仕様によっては，［Cosminexus］－［環境構築］－［デバッグ環境セットアップ］と表示される場合があります。

開発環境インスタントセットアップ機能が起動して，［セットアップ - 開発環境インスタントセットアップ］ダイアログの［セットアップの種類の選択］ページが表示されます。



「localhost 指定でデバッグ環境を構築する」チェックボックスをチェックした場合，複製用のマスターイメージとしてデバッグ環境を構築できます。必要に応じて選択してください。

2. [カスタム] を選択して，[次へ] ボタンをクリックします。
   [デバッグ環境の選択] ページが表示されます。



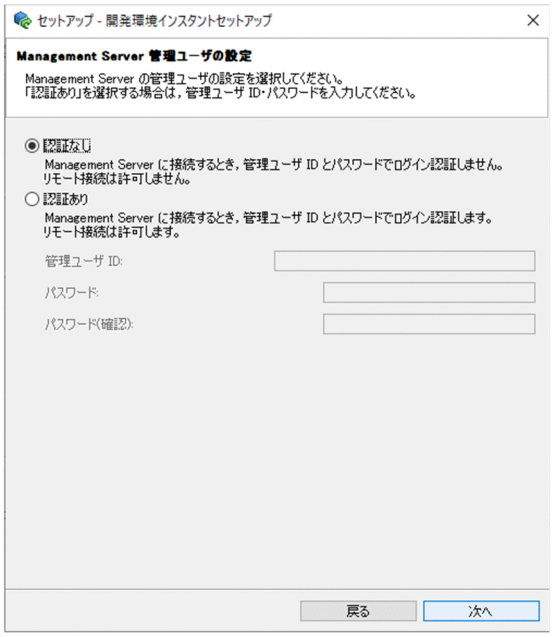
3. デバッグ環境を構築するマシンの環境に合わせて，構築するデバッグ環境を選択します。また，組み込みデータベースを構築するかどうかも指定します。

項目名		指定値
J2EE サーバモードの選択	推奨モード	推奨の設定で J2EE サーバを構築します。

項目名		指定値
	V9 互換モード	Cosminexus V9 と互換性のある J2EE サーバを構築します。 V9 以前から引き継ぐアプリケーションの開発に適しています。 ただし，V11 以降で追加された新機能は使用できません。 注意事項 JDK17 をインストールした環境では「V9 互換モード」は使用できないため，選択しないでください。JDK17 をインストールした環境で「V9 互換モード」を選択した場合，セットアップは失敗します。詳細はマニュアル「アプリケーションサーバ 機能解説 互換編」の「2.2　インストール環境によって使用できるモード」，およびマニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「9.9.1　J2EE サーバの追加」を参照してください。
組み込みデータベースの設定	組み込みデータベースを構築する	組み込みデータベースを構築するかどうかを指定します。 • 組み込みデータベースを構築する場合 チェックします。 • 組み込みデータベースを構築しない場合 チェックを外します。なお，チェックを外した場合は，手順 8.～手順 11.は省かれます。

4. [次へ] ボタンをクリックします。

[Management Server 管理ユーザの設定] ページが表示されます。



5. 次の項目を指定します。

項目名	指定値
認証なし	Management Server リモート管理機能で管理ユーザを設定しない場合に選択します。

項目名	指定値
認証あり	Management Server リモート管理機能で管理ユーザを設定する場合に選択します。
管理ユーザ ID	［認証あり］を選択した場合，Management Server リモート管理機能で使用する管理ユーザ ID を指定します。 8 バイト以内の半角英数字で指定してください。
パスワード	［認証あり］を選択した場合，Management Server リモート管理機能で使用するパスワードを指定します。 半角英字で始まる 30 バイト以内の半角英数字で指定してください。
パスワード(確認)	［認証あり］を選択した場合，［パスワード］に指定した値と同じ値を指定します。

6. ［次へ］ボタンをクリックします。

［組み込みデータベースのセットアップ］ページが表示されます。

手順 3.で［組み込みデータベースを構築する］のチェックを外した場合は，［ポート番号の設定］ページが表示されます。手順 12.に移ります。

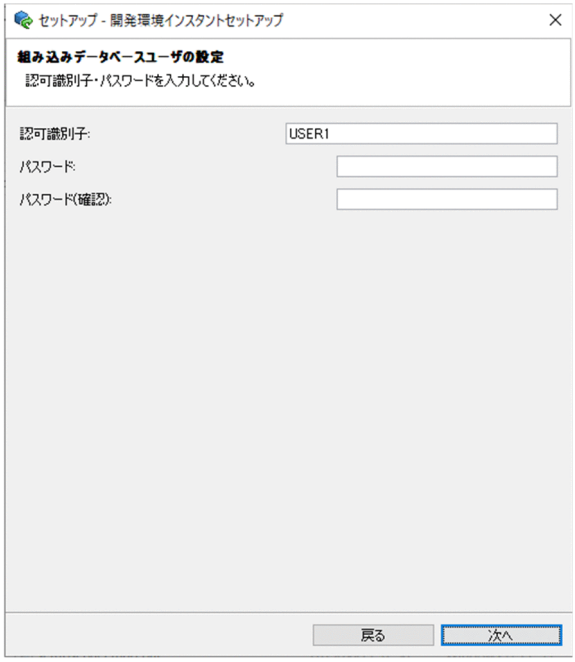
7. デバッグ環境で使用するデータベースの設定内容として，次の項目を指定および選択します。

項目名	指定値
組み込みデータベース構築ディレクトリ	組み込みデータベースの設定ファイルおよび RD エリアのファイルを格納するディレクトリを 80 バイト以内で指定します。 デフォルトから変更したい場合は，［参照］ボタンから格納するディレクトリを指定できます。 注意 次に示す条件のどれかに当てはまるディレクトリは指定できません。 • ディレクトリがない。

項目名	指定値
	• ディレクトリのパスが UNC で表記されている。 • ディレクトリ内に次の名前のディレクトリまたはファイルが存在する。 area，bats，conf，ini • ネットワークドライブを使用したパスを指定している。
データベースサイズ	構築したいデータベースのサイズを選択します。 • Small(500MB) • Medium(1GB) • Large(2GB)

8. [次へ] ボタンをクリックします。

[組み込みデータベースユーザの設定] ページが表示されます。



9. 次の項目を指定します。

項目名	指定値
認可識別子	組み込みデータベースを操作するときに使用する認可識別子を指定します。 8 バイト以内の半角英数字で指定してください。
パスワード	組み込みデータベースを操作するときに使用するパスワードを指定します。 半角英字で始まる 30 バイト以内の半角英数字で指定してください。
パスワード(確認)	[パスワード] に指定した値と同じ値を指定します。

10. [次へ] ボタンをクリックします。

[ポート番号の設定] ページが表示されます。

11. デバッグ環境で使用するポート番号を指定します。

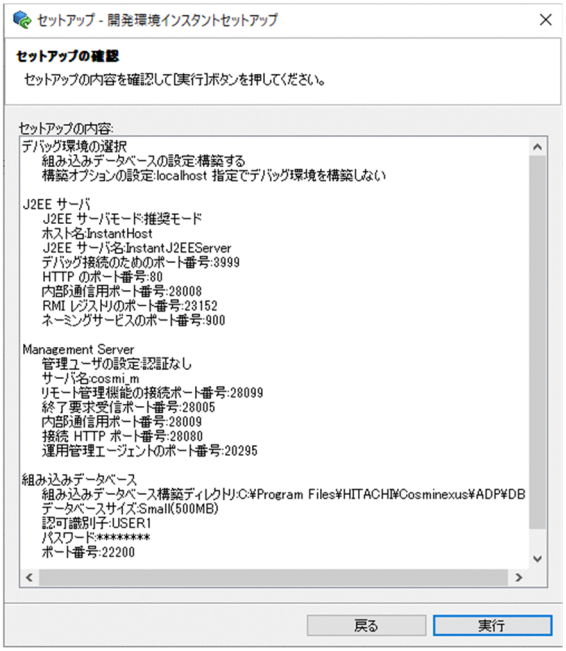
［∧］および［∨］ボタン，［↑］および［↓］キー，または直接数値を入力して指定してください。

項目名		指定値
J2EE サーバのポート番号	デバッグ接続のためのポート番号	J2EE サーバにデバッグ接続するために使用するポート番号を半角数字で指定します。 指定できる範囲は 0～65535 です。
	HTTP のポート番号	J2EE サーバが利用する HTTP のポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	内部通信用ポート番号	J2EE サーバが利用する内部通信用ポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	RMI レジストリのポート番号	J2EE サーバが利用する RMI レジストリのポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	ネーミングサービスのポート番号	J2EE サーバがネーミングサービスとして利用する CORBA ネーミングサービスのポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
Management Server のポート番号	リモート管理機能の接続ポート番号	Management Server リモート管理機能への外部接続ポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	終了要求受信ポート番号	Management Server 終了要求受信ポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。

項目名		指定値
	内部通信用ポート番号	Management Server 内部通信用ポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	接続 HTTP ポート番号	Management Server 接続 HTTP ポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	運用管理エージェントのポート番号	Management Server が使用する運用管理エージェントのポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
組み込みデータベースのポート番号	ポート番号	組み込みデータベースが使用するポート番号を半角数字で指定します。 指定できる範囲は 5001～65535 です。

12. [次へ] ボタンをクリックします。

[セットアップの確認] ページが表示されます。



[セットアップの内容] エリアにセットアップされる内容が表示されます。設定値を確認してください。カスタムセットアップで指定しなかった設定値については，「2.3.3　セットアップする環境の設定内容」を参照してください。

なお，Management Server 管理ユーザのパスワードおよび組み込みデータベースユーザのパスワードについては，指定した値に関係なく 8 文字の「*（アスタリスク）」で表示されます。

13. [実行] ボタンをクリックします。

[進行状況] ページが表示されます。

セットアップが終了すると，[セットアップの完了] ページが表示されます。

14. [終了] ボタンをクリックします。

［セットアップ - 開発環境インスタントセットアップ］ダイアログが閉じます。

処理が中断された場合の対処，セットアップ後の各サーバの状態，開発環境インスタントセットアップ機能では変更できない設定の変更方法については，「2.3.4　デバッグ環境の標準セットアップ」の注意事項および参考事項を参照してください。

2.4　Eclipse セットアップ機能を使用したセットアップ

ここでは，Eclipse のダウンロード，およびセットアップ機能を使用したセットアップについて説明します。

2.4.1　Eclipse セットアップ機能実行時の注意事項

Eclipse セットアップ機能で環境を構築する際の注意事項を説明します。

(1)　<Developer のインストールディレクトリ>¥ADP 以下のディレクトリおよびファイルについて

Eclipse セットアップ機能の実行中に，<Developer のインストールディレクトリ>¥ADP 以下のディレクトリおよびファイルを削除または名称変更しないでください。

(2)　サポートしている Eclipse のバージョンについて

サポートしている Eclipse については，リリースノートを参照してください。

なお，上記のバージョンの Eclipse であれば，ダウンロードサイトから入手した Eclipse でも使用できます。

注意事項

サポートする Eclipse のバージョンは，Developer でインストールした JDK のバージョンによって異なります。

(3)　Eclipse セットアップで使用するランゲージパックについて

Eclipse セットアップ機能では，ランゲージパックを適用した環境を構築できます。

ランゲージパックを適用するためには，添付品のランゲージパックのアーカイブファイルを，Eclipse のアーカイブファイルと同じフォルダに格納してください。添付品のランゲージパック以外のものを使用した際の動作は保証しません。

(4)　Eclipse のバージョン変更について

Eclipse のバージョンを変更する場合は，Eclipse の configuration フォルダを一度削除してから，Eclipse セットアップ機能を使ってバージョンを変更してください。

configuration フォルダを次に示します。

C:¥Users¥（全ユーザ分）¥ADP

(5) Eclipse のインストール先について

Eclipse セットアップでの Eclipse インストール先として，次に示すフォルダは未サポートです。

- 一般権限のアカウントが参照・変更できないフォルダとそのサブフォルダ
- 特定のアカウントしか参照・変更できないフォルダとそのサブフォルダ
- ネットワークドライブ上のフォルダ

2.4.2 Eclipse のダウンロード

ここでは，Eclipse のダウンロードについて説明します。

> **参考**
>
> アーカイブファイルは，Developer に同梱されている添付品のアーカイブファイルを使用することもできます。これを使用する場合は，ダウンロードサイトからの入手は不要です。

1. Eclipse のアーカイブファイルを入手します。
   Developer に同梱されている添付品または Eclipse.org のダウンロードサイトから，サポートしているバージョンを入手してください。

2. ダウンロードした Eclipse のアーカイブファイルを，Eclipse セットアップ機能で使用するため次のフォルダへ格納します。
   <Developer のインストールディレクトリ>¥ADP¥Archives

2.4.3 Eclipse 環境のセットアップ

Eclipse セットアップ機能を使用して，Eclipse 環境をセットアップします。

Eclipse セットアップ機能では，Eclipse のインストール，および Developer が提供する Eclipse プラグインの組み込みを実施します。また，必要に応じてランゲージパックを適用した環境も構築できます。

なお，Eclipse セットアップ機能以外の方法でインストールした Eclipse には，Eclipse セットアップ機能を使用して，Eclipse プラグインを組み込むことはできません。

Eclipse セットアップ機能を使用した場合，デフォルトの Eclipse のインストールディレクトリは次のとおりです。

```
<Developerのインストールディレクトリ>¥ADP¥IDE
```

Eclipse セットアップ機能を使用してセットアップする手順を説明します。

注意事項

- Babel ランゲージパックを適用する場合，製品に添付されている Babel ランゲージパックを使用してください。ダウンロードサイトから入手した Babel ランゲージパックはサポート対象外です。
- Eclipse セットアップ機能を実行する場合は，次の手順で実行します。

  1.スタートメニューのショートカットを右クリックして［ファイルの場所を開く］を選択します。

  2.エクスプローラーで，実行する機能のショートカットを右クリックして，［管理者として実行］を選択します。

1. スタートメニューから，［Cosminexus］－［Eclipse セットアップ］を選択します。

   Eclipse セットアップ機能が起動して，Eclipse のアーカイブファイルをダウンロードしたかどうかを確認するダイアログが表示されます。

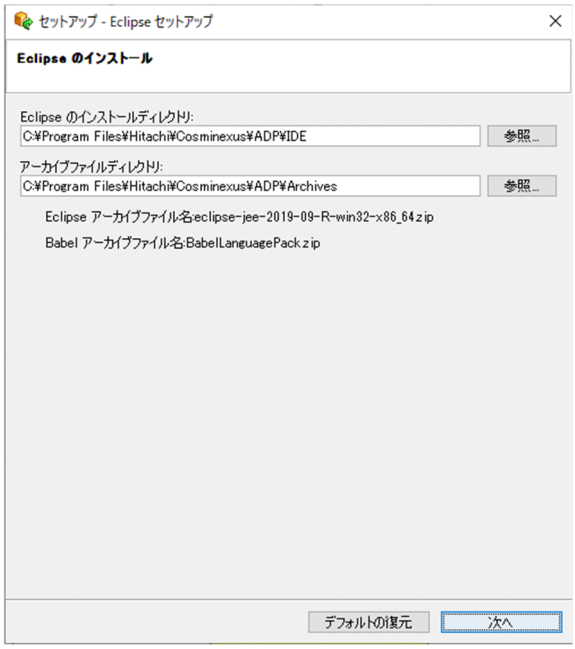
   次回からこのダイアログを表示させない場合は，［次回からこのダイアログを表示しない］チェックボックスにチェックします。

注意事項

手順 1.に示すスタートメニューは，対象 OS のスタートメニューの表示仕様によっては，［Cosminexus］－［環境構築］－［Eclipse セットアップ］と表示される場合があります。

2. ［了解］ボタンをクリックします。

   ［Eclipse のインストール］ページが表示されます。



ここでは，Eclipse のアーカイブファイルをアーカイブファイル格納用ディレクトリ（<Developerのインストールディレクトリ>¥ADP¥Archives）に格納した場合の画面を示しています。

Eclipse のアーカイブファイルをアーカイブファイル格納用ディレクトリに格納していない場合は，［次へ］ボタンが活性化されません。手順 3.で Eclipse のアーカイブファイルを指定してください。

3. Eclipse のインストールディレクトリおよびアーカイブファイルを指定します。

項目名	指定値
Eclipse のインストールディレクトリ	Eclipse のインストールディレクトリを 50 文字以内で指定します。 デフォルトの設定から変更したい場合は，［参照］ボタンから指定してください。 注意 次に示す条件のどちらかに当てはまるディレクトリは指定できません。 • ディレクトリがない。 • ディレクトリのパスが UNC で表記されている。
アーカイブファイルディレクトリ	アーカイブファイルディレクトリの［参照］ボタンから Eclipse のアーカイブファイルを指定します。

4. ［次へ］ボタンをクリックします。

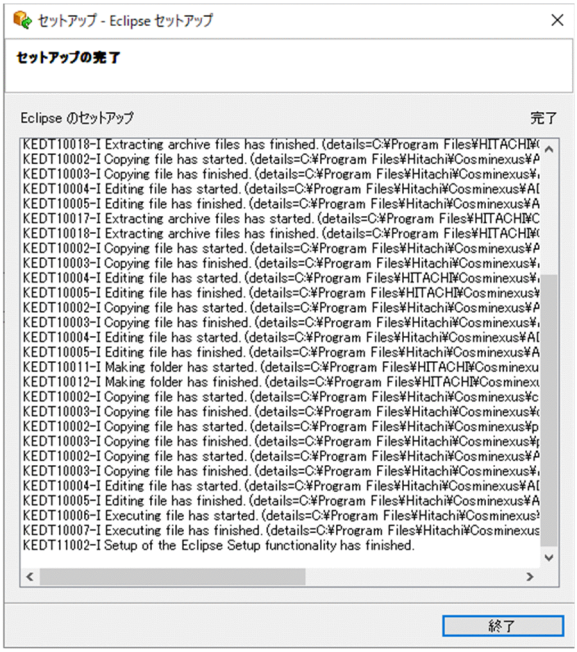
［セットアップの確認］ページが表示されます。

[セットアップの内容] エリアに表示された内容を確認してください。

5. [実行] ボタンをクリックします。

[進行状況] ページが表示されます。

セットアップが終了すると，[セットアップの完了] ページが表示されます。



6. [終了] ボタンをクリックします。

[セットアップ - Eclipse セットアップ] ダイアログが閉じます。Eclipse がセットアップされ，デスクトップ上に Eclipse のショートカットが生成されます。

参考

処理が中断された場合の対処

［中止］ボタンをクリックするなどによって，処理を中断した場合は，アンセットアップを実行してから再度セットアップを実行してください。アンセットアップの手順については，「2.8.2　Eclipse 環境のアンセットアップ」を参照してください。

エラーによって中断された場合は，コンソールに出力されたエラーメッセージを確認してエラーを取り除いてから，アンセットアップしてください。そのあと，再度セットアップを実行してください。

出力されたエラーメッセージについては，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」を参照して対処してください。

Eclipse セットアップ機能の起動オプションの使用

手順 1.のダイアログで設定した［次回からこのダイアログを表示しない］の設定を取り消して起動したい場合は，起動オプション-resetdialog を指定して，Eclipse セットアップ機能を起動します。

次のように起動オプションを指定して，実行してください。

```
<Developerのインストールディレクトリ>¥ADP¥EclipseSetup¥bin¥EclipseSetup.bat -resetdialog
```

セットアップログの確認方法

Eclipse セットアップ機能を実行したときのログは，Eclipse のセットアップログに出力されます。セットアップログの確認方法については「付録 E.4　開発環境インスタントセットアップ機能および Eclipse セットアップ機能実行時の情報の採取」を参照してください。

セットアップした環境の設定変更について

［セットアップ - Eclipse セットアップ］ダイアログで構築した環境は，eclipse.ini ファイルを編集することで設定変更できます。

Eclipse の設定情報は configuration フォルダの所有者アカウントで変更できます。Eclipse の設定情報を 09-00 以前のようにローカルマシン内の全アカウントで共有したい場合は，eclipse.ini の次の記述を行ごと削除してください。

```
-Dosgi.configuration.area=@user.home/ADP/eclipse/configuration
```

2.5　Eclipse の設定

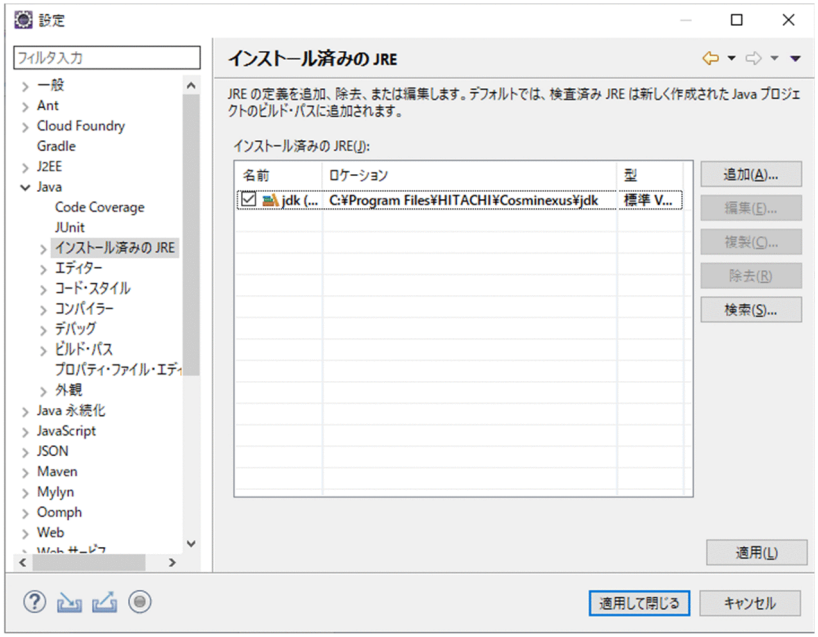
ここでは，J2EE アプリケーションを開発するために必要な，Eclipse 上での JDK の設定を確認する手順を示します。また，J2EE アプリケーションが持つローカル変数情報を出力する場合の設定手順を示します。

2.5.1　JDK の確認

Eclipse 上で開発するときに使用する JDK が Developer で提供されている JDK かどうかを確認します。確認手順を次に示します。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます。

2. 左ペインで［Java］－［インストール済みの JRE］を選択します。
   ［インストール済みの JRE］ページが表示されます。



3. ［インストール済みの JRE］が Developer で提供されている JDK かどうかを確認します。
   ［ロケーション］に次のパスが表示されているかどうかを確認します。
   `<Developerのインストールディレクトリ>¥jdk`

   - パスが表示されていない場合
     ［追加］ボタンをクリックして，上記のパスを指定します。パスの指定後，［名前］のチェックボックスにチェックします。
   - パスが表示されている場合
     ［名前］のチェックボックスにチェックしているかどうかを確認します。チェックしていない場合はチェックしてください。

特に，複数の JDK がインストールされている場合，<Developer のインストールディレクトリ>¥jdk にチェックしていないことがあります。チェックしていない場合は，チェックしてください。

4. [OK] ボタンをクリックします。
   設定が保存されます。

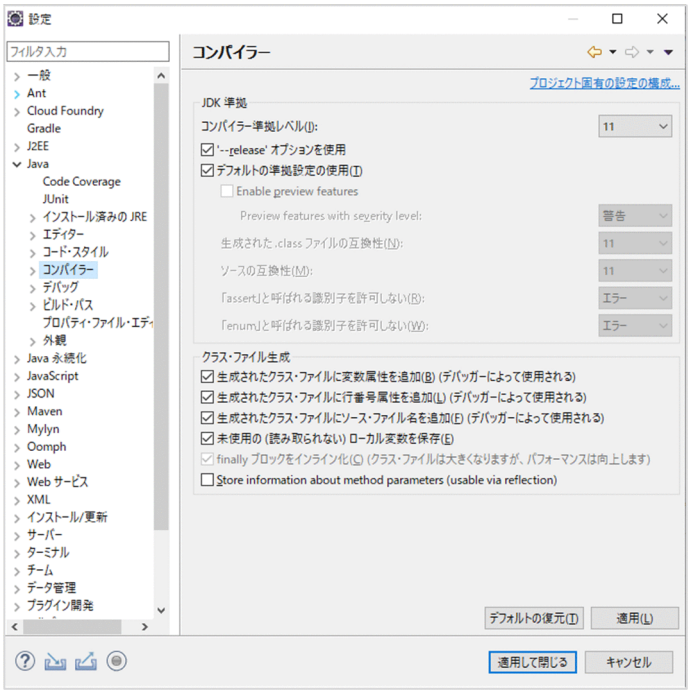
2.5.2 ローカル変数情報の出力の設定

Eclipse のコンパイラの設定によって，例外発生時に J2EE アプリケーションが持つローカル変数の情報をスタックトレース上に出力できます。ローカル変数情報の出力については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「5.10　JavaVM スタックトレース情報」を参照してください。

ローカル変数情報を出力する場合のコンパイラの設定手順を次に示します。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます。

2. 左ペインで［Java］－［コンパイラー］を選択します。
   ［コンパイラー］ページが表示されます。



3. 次の項目を指定します。

項目名		指定値
JDK 準拠	コンパイラー準拠レベル	任意の値を選択します。 デフォルトでは「11」が選択されています。JDK17 をインストールした環境で「17」を選択すると，アプリケーションを JDK17 でコンパイルし，使用できます。JDK11 をインストールした環境で「12」以降

項目名		指定値
		を選択すると，プロジェクトの環境エラーが発生するため選択しないようにしてください。
クラス・ファイル生成		[生成されたクラス・ファイルに変数属性を追加(デバッガーによって使用される)]をチェックします。 なお，それ以外の項目については，出力したい内容に合わせてチェックしてください。

必要に応じて，次の項目を指定してください。

項目名		指定値
JDK 準拠	デフォルトの準拠設定の使用	コンパイラに使用する設定を指定します。 • チェックする [コンパイラー準拠レベル]で指定したレベルに沿った設定が適用されます。 • チェックしない 次の項目を手動で指定します。 [生成された .class ファイルの互換性] [ソースの互換性] [「assert」と呼ばれる識別子を許可しない] [「enum」と呼ばれる識別子を許可しない]

4. [適用]ボタンまたは[OK]ボタンをクリックします。

設定が保存されます。

プロジェクト単位で設定を変更する場合は，Eclipse のメニューの[プロジェクト]－[プロパティ]－[Java コンパイラー]を選択することで，同様の設定ができます。

2.6　Eclipse の設定変更（任意の作業）

次に示す項目は，J2EE アプリケーション開発時に使用される設定です。

- プロキシの設定※
- ソケット操作のブロックのタイムアウト設定
- コンソールの設定

注※

インターネット接続時にプロキシを使用している場合は，使用しているブラウザのプロキシの設定に必ず合わせてください。

プロキシの設定が誤っている場合，Management Server のリモート管理機能への接続などで失敗する場合があります。

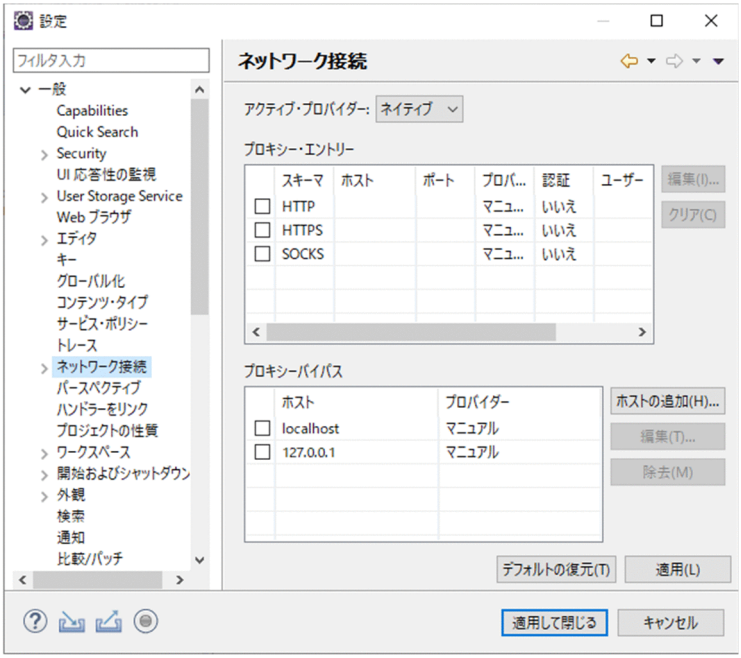
これらの設定は，デフォルト値が設定されていますが，使用したい設定に変更できます。

ここでは，これらの設定を変更する手順を説明します。

2.6.1　Eclipse のプロキシの設定

インターネット接続時にプロキシを使用している環境では，Eclipse でプロキシを設定します。プロキシの設定は，使用しているブラウザのプロキシの設定に合わせてください。プロキシの設定手順を次に示します。

1. **Eclipse のメニューから［ウィンドウ］－［設定］を選択します。**
   ［設定］ダイアログが表示されます。
2. **左ペインのツリービューで［一般］－［ネットワーク接続］を選択します。**
   ［ネットワーク接続］ページが表示されます。



3. [アクティブ・プロバイダー] を選択します。

   Windows の [インターネットオプション] でプロキシの設定がされている場合は，[アクティブ・プロバイダー] として [ネイティブ] が選択されているため，[OK] ボタンをクリックします。

   [インターネットオプション] でのプロキシ設定値とは異なる設定を使用する場合などは，[アクティブ・プロバイダー] として [手操作] を選択し，手順 4.以降の手順に従って値を設定してください。

4. [プロキシー・エントリー] で [HTTP] を選択し，[編集] ボタンをチェックします。

   [プロキシー・エントリーの編集] 画面が表示されます。

5. [ホスト]，[ポート]，および [認証情報] を入力して [OK] ボタンをクリックします。

   [ネットワーク接続] ページに戻ります。

6. [適用] ボタンまたは [OK] ボタンをクリックします。

   設定が保存されます。

2.6.2 ソケット操作のブロックのタイムアウト設定変更

Management Server リモート管理機能と接続するときに使用するソケット操作のブロックのタイムアウト設定を変更したい場合は，[接続] ページで設定内容を変更します。

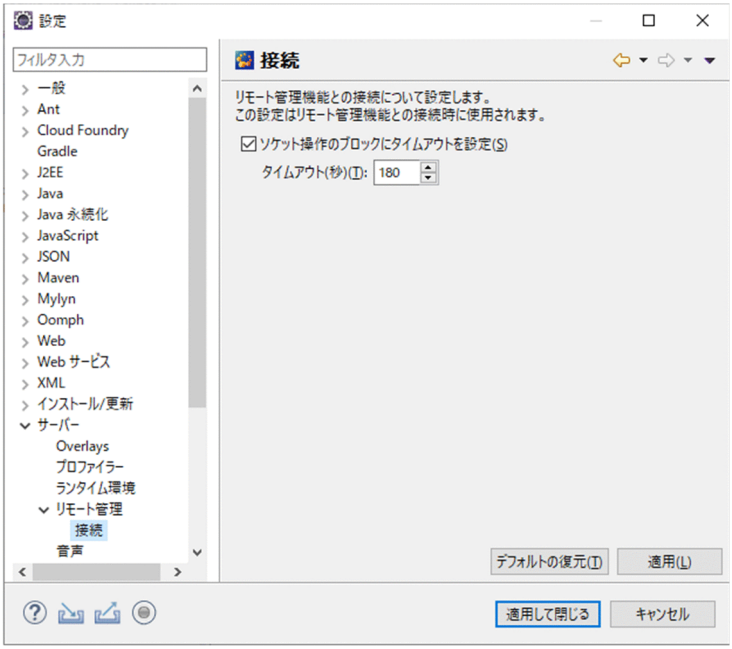
なお，Management Server リモート管理機能にログインしている場合は設定できません。Management Server リモート管理機能からログアウトして，設定してください。設定手順を次に示します。

1. Eclipse のメニューから，[ウィンドウ] − [設定] を選択します。

   [設定] ダイアログが表示されます。

2. 左ペインのツリービューで［サーバー］－［リモート管理］－［接続］を選択します。

［接続］ページが表示されます。



3. 使用したい環境に合わせて，次の項目を変更します。

項目名	指定値
ソケット操作のブロックにタイムアウトを設定	ソケット操作のブロックにタイムアウトを設定するかどうかを指定します。 • チェックする タイムアウトを設定します。 • チェックしない タイムアウトを設定しません。
タイムアウト(秒)	ソケット操作のブロックでタイムアウトする時間を秒で指定します。 指定できる値の範囲は，180～99999 秒です。

注

ソケット操作のブロックにタイムアウトを設定している場合で，Management Server リモート管理機能を使用した処理を実行しているときに，「Read time out」のメッセージが表示されることがあります。このメッセージが表示されたあとも，Management Server リモート管理機能の処理が続行されていることがあります。

4. ［適用］ボタンまたは［OK］ボタンをクリックします。

設定が保存されます。

注

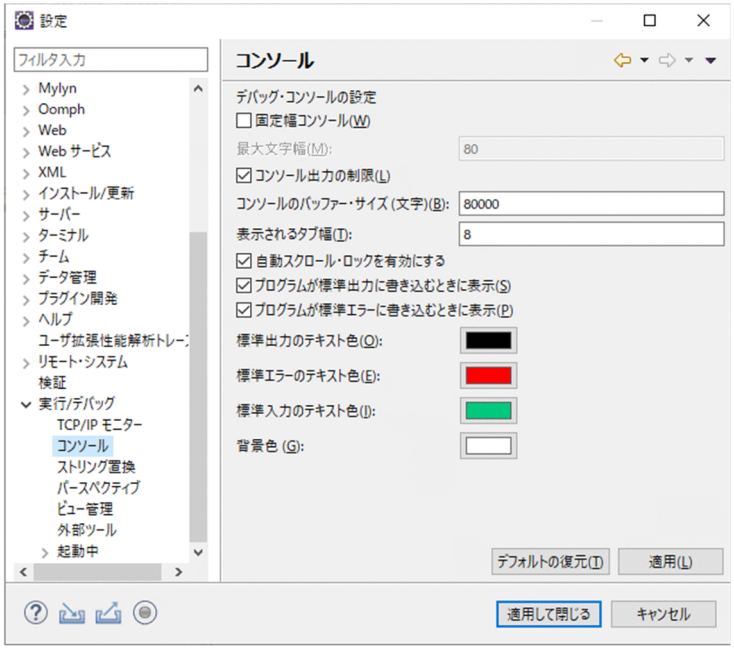
タイムアウトの値を変更した場合，変更した値を保存しないで Management Server リモート管理機能にログインすると，変更した値は破棄され，変更前の値で動作します。

2.6.3　コンソールの設定変更

J2EE サーバのプロセスの標準出力や標準エラー出力は，Eclipse が提供するコンソールに表示されます。コンソールの出力形式の設定を変更したい場合は，Eclipse の［設定］ダイアログで変更できます。Eclipse のコンソールの設定を変更する手順を次に示します。

1. Eclipse のメニューから，［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます。

2. 左ペインのツリービューで［実行/デバッグ］－［コンソール］を選択します。
   ［コンソール］ページが表示されます。



3. 必要に応じて，次の項目を変更してください。

項目名	指定値
固定幅コンソール	コンソールビューで表示する 1 行の文字数を固定するかどうかを指定します。 • チェックする 1 行の文字数を固定します。 • チェックしない 1 行の文字数を固定しません。 1 行の文字数を固定する場合，［最大文字幅］も指定します。 なお，1 行の文字数が［最大文字幅］に指定した数を超える場合は，改行して表示されます。
最大文字幅	コンソールビューに表示する 1 行の文字数を指定します。80～1000 の範囲で指定します。 ［固定幅コンソール］をチェックしている場合は必ず指定します。
コンソール出力の制限	コンソールの出力を制限するかどうかを指定します。 • チェックする

項目名	指定値
	コンソールの出力を制限します。 • チェックしない コンソールの出力を制限しません。 なお，コンソールの出力を制限している場合に，出力された行までの文字数が［コンソールのバッファー・サイズ(文字)］で指定されているバッファサイズを超えると，先頭から超えた分のメッセージが消去されます。消去されたメッセージは次のファイルで確認できます。 <Eclipseのワークスペースディレクトリ>¥.metadata¥.log
コンソールのバッファー・サイズ(文字)	コンソールのバッファサイズ（文字数）を，1000～1000000 の範囲で指定します。表示したいサイズに調節してください。 ［コンソール出力の制限］をチェックしている場合に必ず指定します。
表示されるタブ幅	コンソールビューで表示するタブ幅を文字数で指定します。1～100 の範囲で指定します。
プログラムが標準出力に書き込むときに表示	J2EE サーバや J2EE アプリケーションが標準出力にメッセージを出力したときに，コンソールビューを前面に表示するかどうかを指定します。 • チェックする コンソールビューを前面に表示します。 • チェックしない コンソールビューを前面に表示しません。
プログラムが標準エラーに書き込むときに表示	J2EE サーバや J2EE アプリケーションが標準エラー出力にメッセージを出力したときに，コンソールビューを前面に表示するかどうかを指定します。 • チェックする コンソールビューを前面に表示します。 • チェックしない コンソールビューを前面に表示しません。
標準出力のテキスト色	標準出力のテキストの色を指定します。
標準エラーのテキスト色	標準エラー出力のテキストの色を指定します。
標準入力のテキスト色	標準入力のテキストの色を指定します。
背景色	コンソールの背景色を指定します。

4. ［適用］ボタンまたは［OK］ボタンをクリックします。

設定が保存されます。

2.7　デバッグ環境の設定変更

開発環境インスタントセットアップ機能で構築したデバッグ環境の設定内容は，開発環境インスタントセットアップ機能を使用して変更できます。

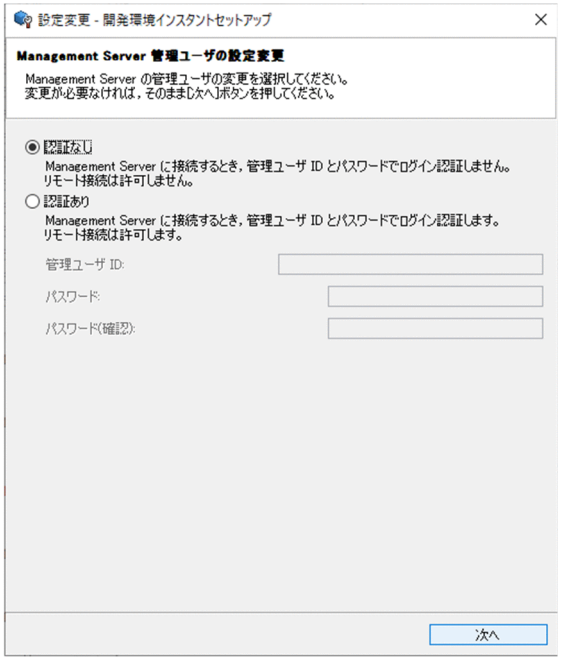
開発環境インスタントセットアップ機能でデバッグ環境の設定内容を変更する手順を次に説明します。

1. スタートメニューから，［Cosminexus］－［デバッグ環境設定変更］※を選択します。

   注※

   対象 OS のスタートメニューの表示仕様によっては，［Cosminexus］－［環境構築］－［デバッグ環境設定変更］と表示される場合があります。

   開発環境インスタントセットアップ機能が起動して，［設定変更 - 開発環境インスタントセットアップ］ダイアログの［Management Server 管理ユーザの設定変更］ページが表示されます。



2. 管理ユーザの設定を変更して，［次へ］ボタンをクリックします。

   管理ユーザの設定変更が不要な場合は，［次へ］ボタンをクリックして次ページへ進みます。

   管理ユーザの設定内容については，「2.3.5　デバッグ環境のカスタムセットアップ」を参照してください。

   ［ポート番号の設定変更］ページが表示されます。

ポート番号の設定変更が不要な場合は，手順 4.へ移ります。

3. J2EE サーバまたは Management Server のポート番号を変更します。

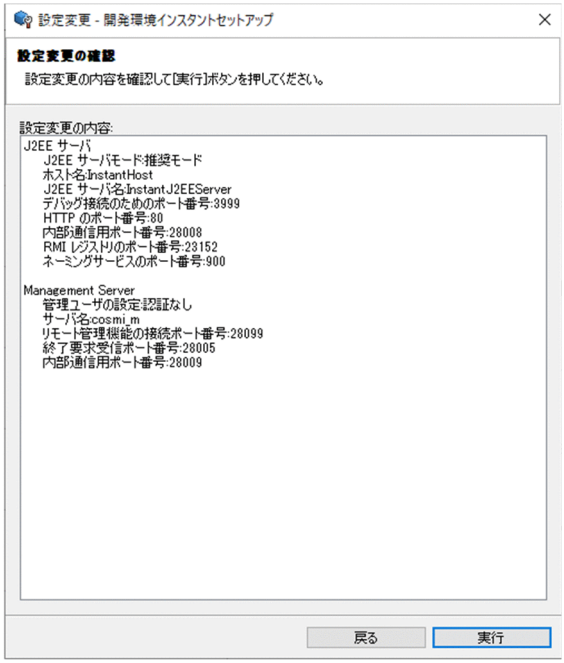
［∧］および［∨］ボタン，［↑］および［↓］キー，または直接数値を入力して指定してください。

項目名		指定値
J2EE サーバのポート番号	デバッグ接続のためのポート番号	J2EE サーバにデバッグ接続するために使用するポート番号を半角数字で指定します。 指定できる範囲は 0～65535 です。
	HTTP のポート番号	J2EE サーバが利用する HTTP のポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	内部通信用ポート番号	J2EE サーバが利用する内部通信用ポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	RMI レジストリのポート番号	J2EE サーバが利用する RMI レジストリのポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	ネーミングサービスのポート番号	J2EE サーバがネーミングサービスとして利用する CORBA ネーミングサービスのポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
Management Server のポート番号	リモート管理機能の接続ポート番号	Management Server リモート管理機能への外部接続ポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。
	終了要求受信ポート番号	Management Server 終了要求受信ポート番号を半角数字で指定します。

項目名		指定値
		指定できる範囲は 1～65535 です。
	内部通信用ポート番号	Management Server 内部通信用ポート番号を半角数字で指定します。 指定できる範囲は 1～65535 です。

4. [次へ] ボタンをクリックします。

   [設定変更の確認] ページが表示されます。



5. [設定変更の内容] エリアに表示された設定値を確認します。

6. [実行] ボタンをクリックします。

   [進行状況] ページが表示されます。

   設定変更が終了すると，[設定変更の完了] ページが表示されます。

7. [終了] ボタンをクリックします。

[設定変更 - 開発環境インスタントセットアップ] ダイアログが閉じます。

注意事項

エラーが発生した場合の対処

エラーが発生した場合はの対処については，「2.3.4　デバッグ環境の標準セットアップ」の注意事項にある処理が中断された場合の対処の説明を参照してください。

開発環境インスタントセットアップ機能以外から環境を変更する場合

開発環境インスタントセットアップ機能で構築した J2EE サーバの設定は，運用管理ポータルでも変更できます。

ただし，運用管理ポータルで設定変更した J2EE サーバに対して，再度開発環境インスタントセットアップ機能で設定を変更しようとすると，エラーが発生するおそれがあります。

[設定変更 - 開発環境インスタントセットアップ] ダイアログ以外で環境を変更したあとに，[設定変更 - 開発環境インスタントセットアップ] ダイアログで設定を変更すると，[設定変更 - 開発環境インスタントセットアップ] ダイアログ以外で変更した設定が，デバッグ環境から削除されます。[設定変更 - 開発環境インスタントセットアップ] ダイアログ以外で変更した設定が，デバッグ環境から削除された場合，削除された設定を再度設定してください。

参考

設定変更後の各サーバの状態

開発環境インスタントセットアップ機能を使用して設定を変更すると，各サーバは次の状態になります。

表 2-4　設定変更後の各サーバの状態

サーバ名	状態
運用管理エージェント	○
Management Server	○
J2EE サーバ	×※
パフォーマンストレーサ	△
組み込みデータベース	△

（凡例）

○：開始　×：停止　△：設定変更前と同じ状態

注※

J2EE サーバが開始されたときに，変更した内容が有効になります。

セットアップログの確認方法

開発環境インスタントセットアップ機能を実行したときのログは，デバッグ環境のセットアップログで確認できます。セットアップログの確認方法は「付録 E.4　開発環境インスタントセットアップ機能および Eclipse セットアップ機能実行時の情報の採取」を参照してください。

2.8　アンセットアップ

ここでは，デバッグ環境と Eclipse 環境をアンセットアップする手順を説明します。

開発環境インスタントセットアップ機能および Eclipse セットアップ機能で構築した環境をアンセットアップしたい場合は，必ず開発環境インスタントセットアップ機能および Eclipse セットアップ機能のアンセットアップを実行してください。

開発環境インスタントセットアップ機能および Eclipse セットアップ機能のアンセットアップを実行すると，環境がすべて削除されます。組み込みデータベース構築ディレクトリの下に手動で追加したファイルやディレクトリも削除されます。Eclipse のワークスペースやプロジェクトなどのデータは，このディレクトリ以外のディレクトリに作成してください。

2.8.1　デバッグ環境のアンセットアップ

開発環境インスタントセットアップ機能で構築したデバッグ環境をアンセットアップする手順を次に説明します。

なお，開発環境インスタントセットアップ機能で構築した環境をアンセットアップする前に，Developer をアンインストールおよび再インストールした場合は，「(2)　手動でアンセットアップする場合」を参照してください。

注意事項

開発環境インスタントセットアップ機能で構築したデバッグ環境に手動で論理サーバを追加した場合

開発環境インスタントセットアップ機能で構築したデバッグ環境に手動で論理サーバを追加した場合は，アンセットアップの前に手動で追加した論理サーバを削除してください。

削除しないでアンセットアップすると，手動で追加した論理サーバが残るため，アンセットアップ後に開発環境インスタントセットアップ機能のセットアップを実行する際，エラーが発生するおそれがあります。

エラーが発生した場合は，手動で追加した論理サーバ，およびホスト（InstantHost）を削除して，開発環境インスタントセットアップ機能のアンセットアップを実行してから，再度セットアップを実行してください。

(1)　開発環境インスタントセットアップ機能を使用する場合

開発環境インスタントセットアップ機能を使用したアンセットアップの手順を次に示します。

1. スタートメニューから，［Cosminexus］－［デバッグ環境アンセットアップ］※を選択します。

注※

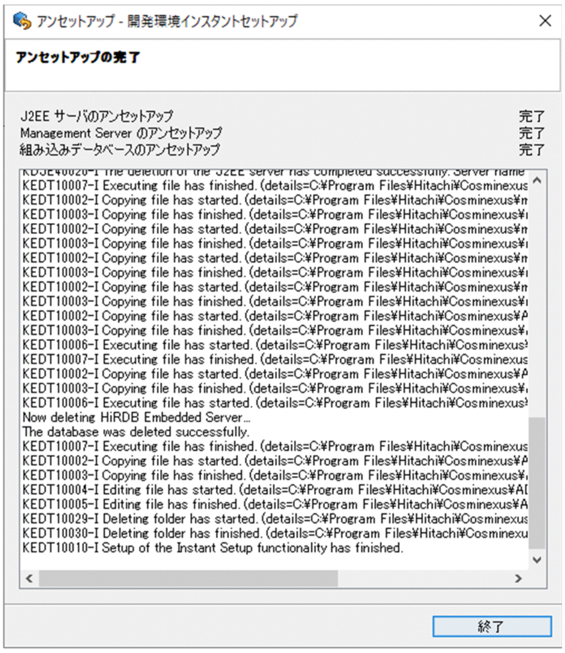
対象 OS のスタートメニューの表示仕様によっては，［Cosminexus］－［環境構築］－［デバッグ環境アンセットアップ］と表示される場合があります。

2. 開発環境インスタントセットアップ機能が起動して，［アンセットアップ - 開発環境インスタントセットアップ］ダイアログの［アンセットアップの確認］ページが表示されます。

3. ［アンセットアップの内容］エリアに表示された内容を確認して，［実行］ボタンをクリックします。

［進行状況］ページが表示されます。

アンセットアップが終了すると，［アンセットアップの完了］ページが表示されます。



4. ［終了］ボタンをクリックします。

［アンセットアップ - 開発環境インスタントセットアップ］ダイアログが閉じます。

参考

アンセットアップの処理内容の確認方法

アンセットアップの処理内容は，セットアップログで確認できます。セットアップログの確認方法は，「付録 E.4　開発環境インスタントセットアップ機能および Eclipse セットアップ機能実行時の情報の採取」を参照してください。

アンセットアップ後の各サーバの状態

開発環境インスタントセットアップ機能を使用してアンセットアップすると，各サーバは次の状態になります。

表 2-5　アンセットアップ後の各サーバの状態

サーバ名	状態
運用管理エージェント	×
Management Server	×
J2EE サーバ	－
パフォーマンストレーサ	－
組み込みデータベース	－

（凡例）

×：停止　－：削除されたため，該当しない。

(2)　手動でアンセットアップする場合

開発環境インスタントセットアップ機能で構築した環境をアンセットアップする前に，Developer をアンインストールおよび再インストールした場合は，次の手順で構築した環境を手動でアンセットアップしてください。

1. J2EE サーバおよびパフォーマンストレーサをアンセットアップします。

次に示す Smart Composer 機能のコマンドおよびサーバ管理コマンド（CUI）を実行して，J2EE サーバおよびパフォーマンストレーサをアンセットアップしてください。

- Web システムの停止（cmx_stop_target コマンド）

```
<Developerのインストールディレクトリ>¥manager¥bin¥cmx_stop_target -m localhost:<開発環境インスタントセットアップ機能で指定した接続HTTPポート番号※> -u <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのID> -p <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのパスワード> -mode ALL -s InstantWebSystem
```

注※

標準セットアップの場合は，28080 です。

- J2EE サーバのアンセットアップ（cjsetup コマンド）

```
<Developerのインストールディレクトリ>¥CC¥server¥bin¥cjsetup -d cmx_InstantWebSystem_unit1_J2EE_01
```

- Web システムの削除（cmx_delete_system コマンド）

```
<Developerのインストールディレクトリ>¥manager¥bin¥cmx_delete_system -m localhost:<開発環境インスタントセットアップ機能で指定した接続HTTPポート番号※> -u <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのID> -p <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのパスワード> -s InstantWebSystem
```

注※

標準セットアップの場合は，28080 です。

なお，コマンドの詳細はマニュアル「アプリケーションサーバ リファレンス コマンド編」を参照してください。

2. Management Server をアンセットアップします。

次に示すファイルをコピーし，cjsetup コマンド（J2EE サーバの削除）を実行して，Management Server をアンセットアップしてください。

- コピー元のファイル
  - <Developerのインストールディレクトリ>¥manager¥config¥templates¥adminagent.properties
  - <Developerのインストールディレクトリ>¥manager¥config¥templates¥mserver.properties
  - <Developerのインストールディレクトリ>¥manager¥config¥templates¥mserver.xml
  - <Developerのインストールディレクトリ>¥manager¥config¥templates¥mserver.cfg
- コピー先のディレクトリ

  <Developerのインストールディレクトリ>¥manager¥config
- 実行するコマンド（cjsetup コマンド）

```
<Developerのインストールディレクトリ>¥CC¥server¥bin¥cjsetup -d cosmi_m
```

3. 組み込みデータベースをアンセットアップします。

組み込みデータベースのアンセットアップについては，HiRDB のマニュアルを参照してください。また，次に示すディレクトリを手動で削除してください。

- <組み込みデータベース構築ディレクトリ>¥area
- <組み込みデータベース構築ディレクトリ>¥bats
- <組み込みデータベース構築ディレクトリ>¥conf
- <組み込みデータベース構築ディレクトリ>¥ini

2.8.2　Eclipse 環境のアンセットアップ

Eclipse セットアップ機能で構築した環境は，Eclipse セットアップ機能でアンセットアップします。

Eclipse セットアップ機能を使用したアンセットアップを実施した場合に削除される項目を次に示します。

表 2-6　Eclipse セットアップ機能のアンセットアップで削除される項目

項目	削除の有無
Eclipse Platform	削除される※1
Eclipse の configuration フォルダ (ユーザごとのフォルダ)	削除されない※2
Eclipse の workspace	削除されない
Eclipse のショートカット	削除される

注※1

ユーザが作成したファイルも含め，対象となるフォルダ内にあるすべてのデータが削除されます。

注※2

configuration フォルダおよび workspace は，Eclipse が生成します。このため，Developer が提供する Eclipse セットアップ機能のアンセットアップを実施しても削除されません。削除したい場合は手動で削除してください。手動での削除は，「(2)　手動でアンセットアップする場合」を参照してください。

Eclipse セットアップ機能を使用したアンセットアップの手順を次に示します。

なお，Eclipse セットアップ機能で構築した環境をアンセットアップする前に，Developer をアンインストールおよび再インストールした場合は，「(2)　手動でアンセットアップする場合」を参照してください。

注意事項

- アンセットアップ実行前に，Eclipse を終了してください。Eclipse 起動中にアンセットアップを実行すると，Eclipse のインストールディレクトリが削除されません。なお，アンセットアップ実行後に Eclipse のインストールディレクトリが残っていた場合は，<Eclipse のインストールディレクトリ>¥eclipse 以下のディレクトリおよびファイルを手動で削除してください。
- Eclipse アンセットアップ機能を実行する場合は，以下の手順で実行します。

  1.スタートメニューのショートカットを右クリックして［ファイルの場所を開く］を選択します。

  2.エクスプローラーで，実行する機能のショートカットを右クリックして，［管理者として実行］を選択します。

(1)　Eclipse セットアップ機能を使用する場合

1. スタートメニューから，［Cosminexus］－［Eclipse アンセットアップ］を選択します。

Eclipse セットアップ機能が起動して，［アンセットアップ - Eclipse セットアップ］ダイアログの［アンセットアップの確認］ページが表示されます。

注意事項

手順 1.に示すスタートメニューは，対象 OS のスタートメニューの表示仕様によっては，［Cosminexus］－［環境構築］－［Eclipse アンセットアップ］と表示される場合があります。

2. ［アンセットアップの内容］エリアに表示された内容を確認して，［実行］ボタンをクリックします。

［進行状況］ページが表示されます。

アンセットアップが終了すると，［アンセットアップの完了］ページが表示されます。

3. [終了] ボタンをクリックします。

[アンセットアップ - Eclipse セットアップ] ダイアログが閉じます。

参考

アンセットアップの処理内容の確認方法

アンセットアップの処理内容は，セットアップログで確認できます。セットアップログの確認方法は，「付録 E.4　開発環境インスタントセットアップ機能および Eclipse セットアップ機能実行時の情報の採取」を参照してください。

注意事項

アンセットアップを実行すると，セットアップ実行時にデスクトップに追加した，eclipse.exe へのショートカットが削除されますが，画面には表示されたままで，削除されていないように見える場合があります。この場合には，デスクトップを最新の情報に更新すると，ショートカットの削除が画面に反映されます。

(2)　手動でアンセットアップする場合

Eclipse セットアップ機能で構築した環境をアンセットアップする前に，Developer をアンインストールおよび再インストールした場合は，次の手順で，構築した環境を手動でアンセットアップしてください。

1. Eclipse のショートカットを削除します。

次に示すファイルを管理者が削除します。

C:¥Users¥Public¥Desktop¥Eclipse.lnk

2. Eclipse の configuration フォルダを削除します。

フォルダの場所については，「2.4.1　Eclipse セットアップ機能実行時の注意事項」の Eclipse のバージョン変更に関する説明を参照してください。

3. Eclipse のフォルダを削除します。

デフォルトのフォルダの場所を次に示します。

<Developer のインストールディレクトリ>¥ADP¥IDE¥eclipse

2.9 Developer のアンインストール

Developer をアンインストールします。アンインストールには，Administrator 権限または管理者特権が必要です。また，Developer をアンインストールする前に，必ずデバッグ環境と Eclipse をアンセットアップしてください。

Developer のアンインストール方法については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「3.3.2　Application Server をアンインストールする（Windows の場合）」を参照してください。

なお，アンインストール手順の製品名「Application Server」は，「Developer」と読み替えてください。

3 デバッグ環境で使用するデータベースのテーブルの作成

組み込みデータベースのテーブルは，組み込みデータベースを操作して作成します。この章では，デバッグ環境で使用するデータベースのテーブルを HiRDB SQL Executer を使用して作成する方法について説明します。

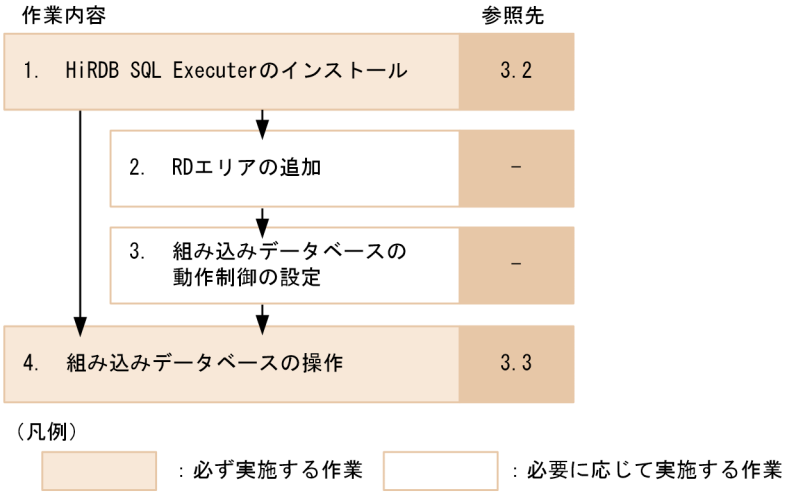
3.1　組み込みデータベースのテーブルの作成の流れ

Developer では，組み込みデータベースを提供しています。組み込みデータベースは，開発した J2EE アプリケーションのテストおよびデバッグで使用します。

組み込みデータベースは，Developer が提供する開発環境インスタントセットアップ機能によって構築されます。

組み込みデータベースのテーブルについては，HiRDB SQL Executer を使用して作成します。組み込みデータベースのテーブルを作成する流れを次の図に示します。

図 3-1　組み込みデータベースのテーブルを作成する流れ



注意事項

- Developer で提供している組み込みデータベースでは，データベースに障害が発生した際に障害を回復する機能を提供していません。このため，Developer で提供している組み込みデータベースに，復旧できないと問題となるデータを格納しないようにしてください。
- テスト・デバッグの際に開発データベースの内容を確認するため，Developer には HiRDB SQL Executer が付属しています。HiRDB SQL Executer は，Developer で開発したアプリケーションのテスト・デバッグ以外の目的では使用できません。

それぞれの作業の概要を説明します。

1. HiRDB SQL Executer のインストール

   組み込みデータベースを操作するために，HiRDB SQL Executer をインストールします。詳細は，「3.2　HiRDB SQL Executer のインストール」を参照してください。

2. RD エリアの追加

   使用する組み込みデータベースの内容に合わせて，データベースに RD エリアを追加します。詳細は，HiRDB のマニュアルを参照してください。

3. 組み込みデータベースの動作制御の設定

使用する組み込みデータベースの内容に合わせて，組み込みデータベースの動作制御について設定します。詳細は，HiRDB のマニュアルを参照してください。

4. 組み込みデータベースの操作

HiRDB SQL Executer を使用して，構築した組み込みデータベースを操作して，テーブルの作成または参照をします。詳細は，「3.3　組み込みデータベースの操作」を参照してください。

以降の節では，この流れに沿って組み込みデータベースの設定の手順を説明します。

なお，開発環境インスタントセットアップ機能で構築した組み込みデータベースは，開発環境インスタントセットアップ機能からアンセットアップできます。

3.2 HiRDB SQL Executer のインストール

組み込みデータベースの操作には，HiRDB SQL Executer を使用します。HiRDB SQL Executer は，手動でインストールする必要があります。インストールに必要な準備とインストール方法について説明します。

インストールの準備

- 異なるバージョンおよびリビジョンの HiRDB SQL Executer がインストールされている場合，新規に HiRDB SQL Executer をインストールできません。インストール済みの HiRDB SQL Executer をアンインストールしてから，新規に HiRDB SQL Executer をインストールしてください。
- HiRDB SQL Executer をインストールすると，GUI 版 HiRDB SQL Executer および簡易 GUI 版 HiRDB SQL Executer がインストールされます。GUI 版 HiRDB SQL Executer を使用するためには，あらかじめ「Microsoft .NET Framework」の 2.0 以上および「Microsoft .NET Framework 日本語 Language Pack」の 2.0 以上をインストールしておく必要があります。

インストール方法

1. Developer のインストール CD-ROM を，CD-ROM ドライブにセットします。[日立総合インストーラ］ダイアログに，「選択されたソフトウェアをインストールします。」と表示されます。[日立総合インストーラ］ダイアログが表示されない場合，エクスプローラを使用して，CD-ROM ディレクトリの「HCD_INST.EXE」をダブルクリックしてください。
2. HiRDB SQL Executer Version 9 を選択した状態で，［インストール実行］ボタンをクリックします。[インストール処理開始の確認］－［日立総合インストーラ］ダイアログに，「インストールを開始します。よろしいですか？」と表示されますので，［OK］ボタンをクリックします。
3. HiRDB SQL Executer Version 9 のインストーラが起動しますので，インストーラのガイドに従ってインストールします。
4. インストール後，システム環境変数 PATH に次のディレクトリを追加してください。

   <Developerのインストールディレクトリ>¥DB¥CLIENT¥UTL

システム環境変数 PATH は，Windows の［コントロールパネル］の中の［システム］－［システムの詳細設定］を選択し，［詳細設定］タブの中の［環境変数］ボタンをクリックして設定します。

システム環境変数 PATH に設定済みの定義内容がある場合は，セミコロン（;）で区切って設定してください。システム環境変数 PATH が正しく設定されていない場合は，HiRDB SQL Executer の起動時にエラーが発生します。

3.3 組み込みデータベースの操作

ここでは，組み込みデータベースのテーブルを作成または参照する方法について説明します。なお，HiRDB SQL Executer を使用して組み込みデータベースを操作する場合は，次の条件を満たしている必要があります。開発環境インスタントセットアップ機能で構築した環境は次の条件を満たしているので，HiRDB SQL Executer で操作できます。

- 組み込みデータベースが構築されていること。
- ユーザ定義が完了していること。
- 組み込みデータベースに RD エリアが追加されていること。
- データを参照または更新する場合は，参照または更新の対象となるテーブルが組み込みデータベースに作成されていること。

なお，組み込みデータベースを操作するには，組み込みデータベースに接続する必要があります。

組み込みデータベースの操作方法については，HiRDB のマニュアルを参照してください。

3.4 DTP プラグインを使って HiRDB を操作する

HiRDB（データベースサーバ）に接続して SQL を実行するには，DTP プラグインを使用して JDBC ドライバーを定義してからデータベースサーバに接続するための接続プロファイルを作成します。そのあと，Eclipse から SQL を実行します。

前提条件

- デフォルトのデバッグ環境がセットアップされている
- ドメイン管理サーバが起動している
- Eclipse が起動している
- アプリケーション開発環境用のデータベースサーバが起動している

想定ユーザー

- アプリケーション開発者

操作手順

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます。
2. Eclipse の［設定］ダイアログの左ペインで，［データ管理］－［接続］－［ドライバー定義］を選択します。
3. ［ドライバー定義］ページの［追加］ボタンをクリックします。
   ［新規ドライバー定義］ダイアログが表示されます。
4. ［新規ドライバー定義］ダイアログの［利用できるドライバーテンプレート］リストで，HiRDB Type4 JDBC Driver 10 を選択します。
5. ［JAR 一覧］タブを選択し，ドライバーファイル一覧で pdjdbc4.jar を選択して，［JAR/Zip を編集］ボタンをクリックします。
   ［ファイルの選択］ダイアログが表示されます。
6. ［ファイルの選択］ダイアログで，<Developerのインストールディレクトリ>¥DB¥CLIENT¥UTL¥pdjdbc4.jar を指定して，［開く］ボタンをクリックします。
7. ［新規ドライバー定義］ダイアログで，［OK］ボタンをクリックします。
   ［ドライバー定義］ページのリストに，HiRDB Type4 JDBC Driver 10 が表示されます。
8. ［設定］ダイアログで，［OK］ボタンをクリックします。
   ［設定］ダイアログが閉じて，JDBC ドライバーの定義が保存されます。

9. Eclipse の［データ・ソース・エクスプローラー］ビューで，データベース接続を選択して，右クリックして［新規］を選択します。

［新規接続プロファイル］ダイアログが表示されます。

10. ［接続プロファイル］ページの接続プロファイルの種類で HiRDB を選択し，名前を入力して，［次へ］ボタンをクリックします。

［ドライバーおよび接続の詳細の設定］ページが表示されます。

11. ［ドライバーおよび接続の詳細の設定］ページで次の項目を指定します。

- ドライバー

  手順 7 で作成した HiRDB Type4 JDBC Driver 10 の定義を指定します。
- データベース名

  接続するデータベース名を指定します。
- ホスト名

  接続する HiRDB のインストール先のコンピュータ名を指定します。
- ポート番号

  HiRDB で使用する TCP/IP ポート番号を指定します。
- 認可識別子

  HiRDB を操作するときに使用する認可識別子を指定します。
- パスワード

  HiRDB を操作するときに使用するパスワードを指定します。
- パスワードの保管

  入力したパスワードを保管するかどうかを指定します。 チェックした場合，次回以降，パスワードの入力をスキップできます。
- 接続 URL

  HiRDB に接続するための URL が表示されます。

ポイント

ユーザプロパティタブで JDBC_IF=OFF を指定すると，ドライバーのトレース情報が取得できなくなります。

12. ［終了］ボタンをクリックします。

HiRDB に接続するための接続プロファイルが作成されます。［ウィザードの完了時に接続］チェックボックスにチェックした場合は，自動的に HiRDB に接続します。

13. ［データ・ソース・エクスプローラー］ビューで作成した接続プロファイルに接続して，データベースの内容を確認します。

ポイント

データベースオブジェクトの変更は，表示内容に自動で反映されません。データベースを更新した場合，手動で表示内容を更新してください。

14. SQL ファイルを選択して右クリックして［SQL ファイルを実行］を選択し，SQL を実行します。

複合文を実行する場合には，右クリックして［SQL ファイルを一つの命令として実行］を選択します。複合文については，HIRDB のマニュアルを参照してください。実行時には，手順 10～12 で作成した接続プロファイルを指定します。DTP プラグインで使用できる HiRDB の SQL データ型を次に示します。

- INTEGER
- SMALLINT
- DECIMAL，NUMERIC
- FLOAT，DOUBLE PRECISION
- SMALLFLT，REAL
- CHAR
- VARCHAR
- NCHAR
- NVARCHAR
- MCHAR
- MVARCHAR
- DATE
- TIME
- TIMESTAMP
- BLOB
- BINARY

4 Eclipse を使用した J2EE アプリケーションの開発

この章では，Eclipse を使用した J2EE アプリケーションの開発について説明します。

アプリケーションの開発手順として，リモート管理機能の設定，およびサーバランタイムの作成について説明します。また，Eclipse プロジェクトの作成，およびリソースアダプタのインポートについても説明します。

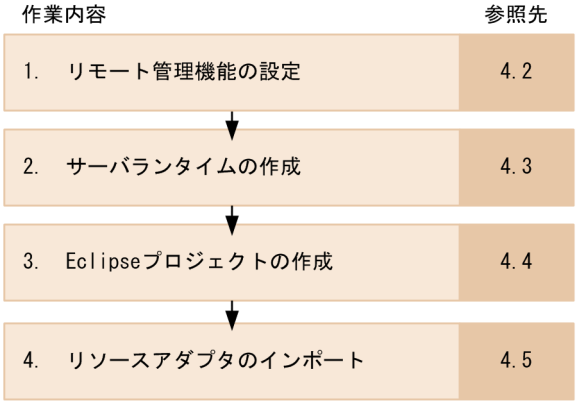
4.1　Eclipse を使用した J2EE アプリケーションの開発

Eclipse を使用した J2EE アプリケーションの開発の流れと Eclipse 操作時の注意事項について説明します。

4.1.1　Eclipse を使用した J2EE アプリケーションの開発の流れ

Eclipse を使用した J2EE アプリケーションの開発の流れを，次の図に示します。

図 4-1　Eclipse を使用した J2EE アプリケーションの開発の流れ



それぞれの作業の概要を次に示します。

1. リモート管理機能の設定

   WTP コネクタを操作するには，リモート管理機能を設定する必要があります。詳細は，「4.2　リモート管理機能の設定」を参照してください。

2. サーバランタイムの作成

   Eclipse で使用する Cosminexus J2EE サーバランタイムを登録します。詳細は，「4.3　サーバランタイムの作成」を参照してください。

3. Eclipse プロジェクトの作成

   J2EE アプリケーションを開発するためのプロジェクトを作成します。詳細は，「4.4　Eclipse プロジェクトの作成」を参照してください。

4. リソースアダプタのインポート

   Eclipse を使った開発では，Developer の提供するリソースアダプタをエンタープライズアプリケーションプロジェクトにインポートできます。詳細は，「4.5　リソースアダプタのインポート」を参照してください。

Eclipse で J2EE アプリケーションを開発するためには，［J2EE］パースペクティブを使用します。以降，J2EE アプリケーション開発の手順を［J2EE］パースペクティブを使用して説明します。

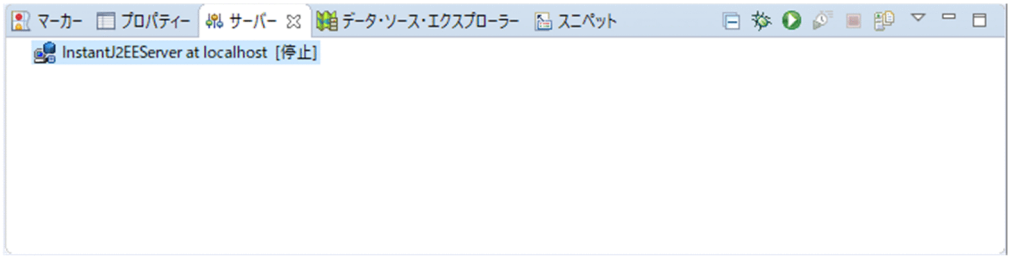
なお，以降の節ではこの流れに沿って Eclipse を使用した J2EE アプリケーションの開発について説明します。

4.1.2 Eclipse 操作時の注意事項

Eclipse を操作する場合は，次の点に注意してください。

- Eclipse のワークスペースの下のフォルダやファイルに対して，エクスプローラなどで直接操作しないでください。Eclipse や WTP コネクタが誤動作する場合があります。
- WTP コネクタと J2EE サーバの状態が不整合になるなどの問題が発生した場合は，サーバ管理コマンドまたは運用管理ポータルを使用して問題を解決してください。例えば，J2EE アプリケーションが存在する状態で，［サーバー］ビューからサーバを削除した場合，サーバ管理コマンドまたは運用管理ポータルを使用して，J2EE アプリケーションを削除する必要があります。［サーバー］ビューの表示例を次に示します。

  図 4–2 ［サーバー］ビューに［InstantJ2EEServer at localhost］が表示されている例



- WTP コネクタ機能を使用する際に［Windows セキュリティの重要な警告］ダイアログが表示された場合は，直ちに［ブロックを解除する］ボタンまたは［アクセスを許可する］ボタンをクリックする必要があります。

4.2　リモート管理機能の設定

Eclipse でアプリケーションを開発するためには，リモート管理機能を設定する必要があります。J2EE サーバの起動，停止などの操作は，リモート管理機能を使用します。

リモート管理機能へのログイン，ログアウトや，リモート管理機能を使用するための接続ホストの新規追加，編集，および削除について説明します。

4.2.1　リモート管理機能へのログインおよびログアウト

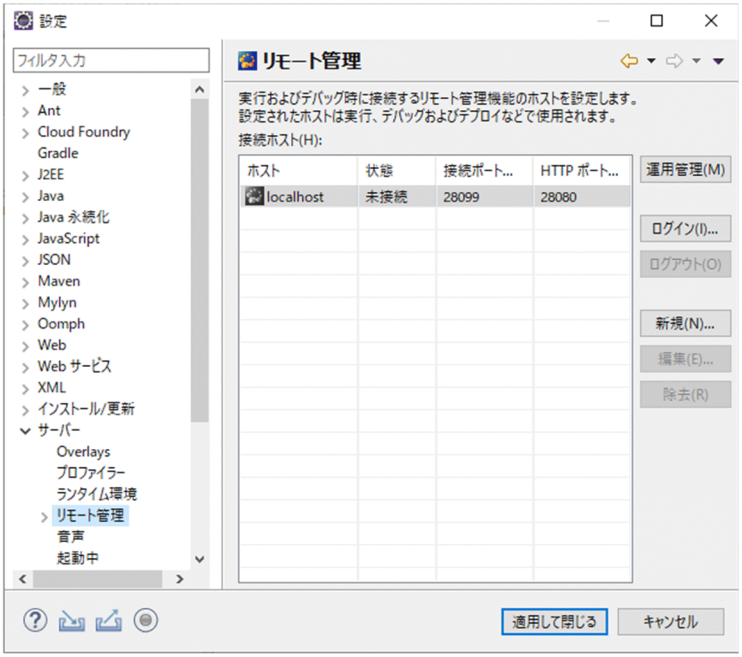
リモート管理機能へのログインおよびログアウトの操作方法を説明します。

> **参考**
>
> ソケット操作のブロックのタイムアウト設定をデフォルトの値から変更したい場合は，Management Server リモート管理機能にログインする前に変更してください。変更手順については，「2.6.2　ソケット操作のブロックのタイムアウト設定変更」を参照してください。

(1)　リモート管理機能へのログイン

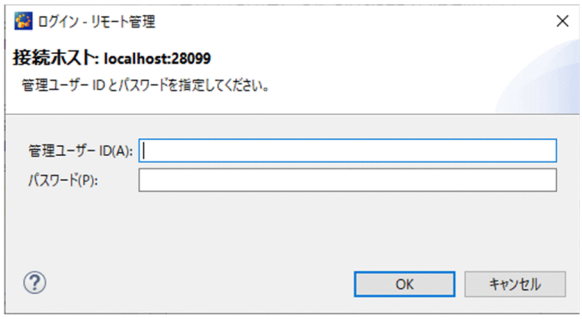
リモート管理機能へのログイン手順を次に示します。

1. Eclipse のメニューから，［ウィンドウ］－［設定］を選択します。
2. 左ペインのツリービューで［サーバー］－［リモート管理］を選択します。
   ［リモート管理］ページが表示されます。



3. 接続する Management Server のホストを選択し，［ログイン］ボタンをクリックします。

Management Server の管理ユーザの設定で管理ユーザの認証を［認証あり］に設定している場合は，［ログイン - リモート管理］ダイアログが表示されます。



Management Server の管理ユーザの設定で管理ユーザの認証を［認証なし］に設定している場合は，［ログイン - リモート管理］ダイアログは表示されないため，手順 4.の作業は不要です。手順 5.へ進んでください。

4. 次の項目を指定します。

項目名	指定値
管理ユーザー ID	Management Server にログインするときの管理ユーザ ID を指定します。 開発環境インスタントセットアップ機能でのセットアップ時に指定した Management Server の管理ユーザ ID を指定します。開発環境インスタントセットアップ機能で構築していない場合は，Management Server 設定時に使用した値を指定します。
パスワード	Management Server にログインするときの管理ユーザパスワードを指定します。 開発環境インスタントセットアップ機能でのセットアップ時に指定した Management Server のパスワードを指定します。開発環境インスタントセットアップ機能で構築していない場合は，Management Server 設定時に使用した値を指定します。

5. ［OK］ボタンをクリックします。

リモート管理機能に接続します。

(2) リモート管理機能からのログアウト

リモート管理機能からのログアウト手順を次に示します。

1. Eclipse のメニューから，［ウィンドウ］－［設定］を選択します。
2. 左ペインのツリービューで［サーバー］－［リモート管理］を選択します。

［リモート管理］ページが表示されます。

3. 接続済みのホストを選択します。
4. ［ログアウト］ボタンをクリックします。

［ログアウト - リモート管理］ダイアログが表示されます。

5. ［はい］ボタンをクリックします。

リモート管理機能からログアウトします。

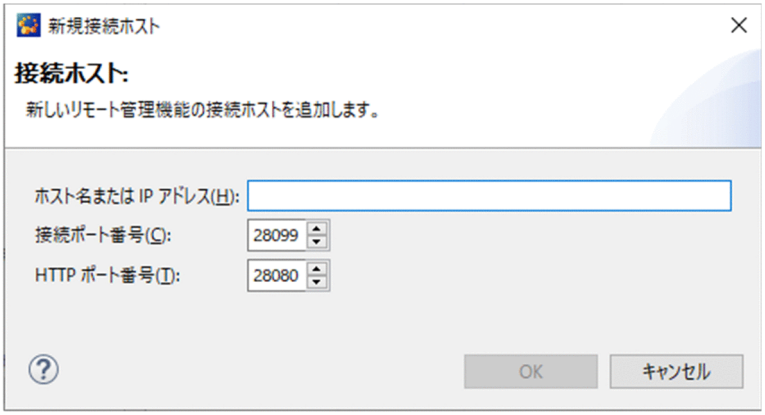
4.2.2　接続ホストの新規追加

デフォルトの接続ホスト（localhost）以外を使用する場合は，次の手順で接続ホストを追加してください。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます。

2. ［設定］ダイアログの左ペインで［サーバー］－［リモート管理］を選択します。
   ［リモート管理］ページが表示されます。

3. ［新規］ボタンをクリックします。
   ［新規接続ホスト］ダイアログが表示されます。



4. 次の項目を指定します。

項目名	指定値
ホスト名または IP アドレス	リモート管理機能の接続ホストのホスト名または IP アドレスを指定します。
接続ポート番号	リモート管理機能に接続するポート番号を 1～65535 の値で指定します。
HTTP ポート番号	運用管理ポータルに接続する HTTP ポート番号を 1～65535 の値で指定します。

5. ［OK］ボタンをクリックします。
   ［リモート管理］ページに追加したホストが表示されます。

4.2.3　接続ホストの編集

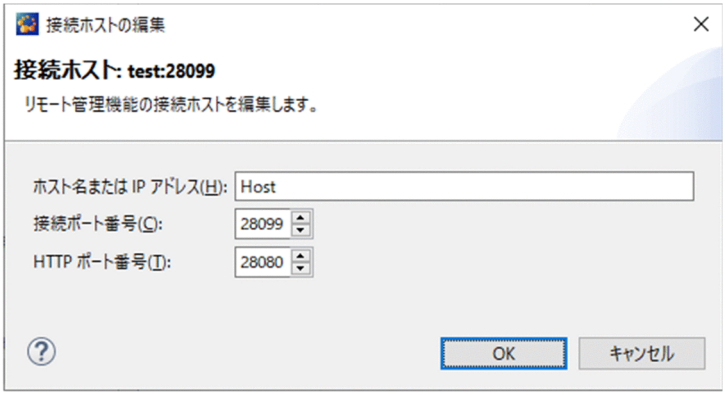
登録済みの接続ホスト（localhost 以外）の設定を変更する場合は，次の手順で接続ホストを編集してください。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。

［設定］ダイアログが表示されます。

2. ［設定］ダイアログの左ペインで［サーバー］－［リモート管理］を選択します。
［リモート管理］ページが表示されます。

3. 編集したい接続ホストを選択し，［編集］ボタンをクリックします。
［接続ホストの編集］ダイアログが表示されます。



4. 次の項目を編集します。

項目名	指定値
ホスト名または IP アドレス	リモート管理機能の接続ホストのホスト名または IP アドレスを指定します。
接続ポート番号	リモート管理機能に接続するポート番号を 1～65535 の値で指定します。
HTTP ポート番号	運用管理ポータルに接続する HTTP ポート番号を 1～65535 の値で指定します。

5. ［OK］ボタンをクリックします。
［リモート管理］ページに編集した内容が反映されます。

4.2.4　接続ホストの削除

登録済みの接続ホスト（localhost 以外）を削除する場合は，次の手順で接続ホストを削除してください。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
［設定］ダイアログが表示されます。

2. ［設定］ダイアログの左ペインで［サーバー］－［リモート管理］を選択します。
［リモート管理］ページが表示されます。

3. 削除したい接続ホストを選択し，［除去］を選択します。
［接続ホストの除去］ダイアログが表示されます。

4. ［はい］ボタンをクリックします。
接続ホストが削除されます。

4.2.5　リモート管理機能設定時の注意事項

リモート管理機能を設定する場合は，次の点に注意してください。

- WTP コネクタから J2EE サーバを起動した場合，運用管理ポータルで管理する設定情報で J2EE サーバの設定が上書きされます。そのため，J2EE サーバの設定変更は，usrconf.cfg や usrconf.properties を直接編集しないで，運用管理ポータルを操作して編集してください。
- WTP コネクタから J2EE サーバを操作する場合，リモート管理機能の接続ホストへログインしてください。
- [サーバー] ビューに J2EE サーバが表示されている状態で，該当する J2EE サーバに対応する接続ホストは削除しないでください。

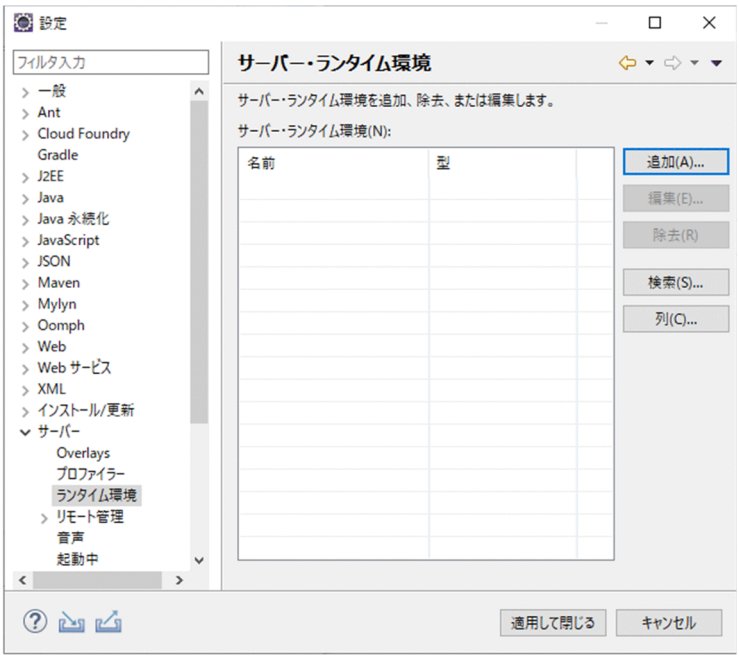
4.3　サーバランタイムの作成

Eclipse で J2EE サーバを操作するためには，J2EE サーバのインストール先などの環境情報を登録する必要があります。この環境情報をサーバランタイムといいます。サーバランタイムを登録することで，Eclipse で J2EE サーバを追加できるようになります。

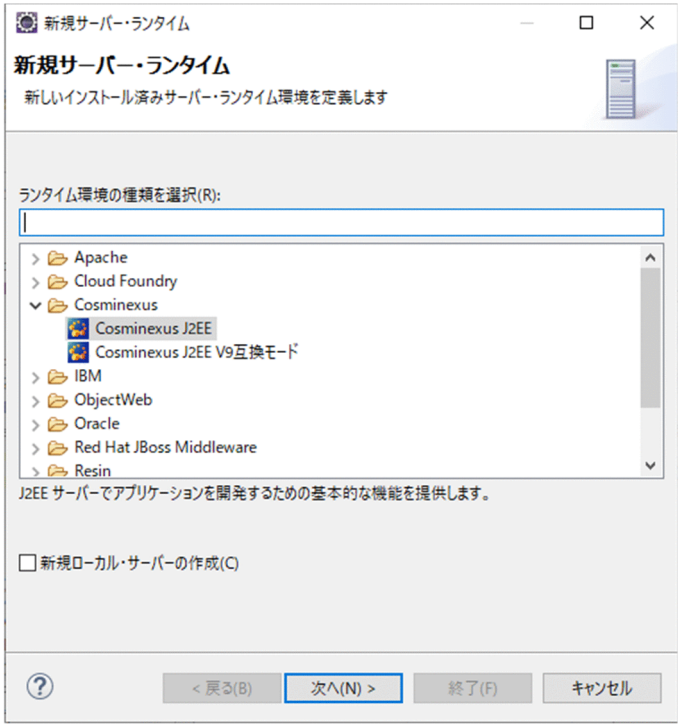
Cosminexus J2EE サーバランタイムを登録する手順を次に示します。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます。

2. ［設定］ダイアログの左ペインで［サーバー］－［ランタイム環境］を選択します。
   ［サーバー・ランタイム環境］ページが表示されます。



3. ［追加］ボタンをクリックします。
   ［新規サーバー・ランタイム］ダイアログが表示されます。



サーバランタイムは，J2EE サーバの J2EE サーバモードに対応した 2 種類があります。サーバランタイムの種類について，次の表に示します。また，この手順では，推奨モードのサーバランタイムを選択しています。

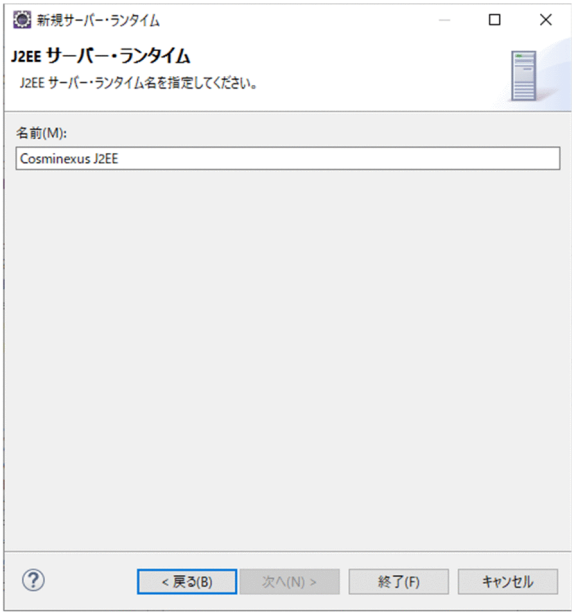
項番	J2EE サーバモード	サーバランタイム	説明
1	推奨モード	Cosminexus J2EE	推奨の設定で J2EE サーバを構築する場合に選択します。
2	V9 互換モード	Cosminexus J2EE V9 互換モード	Cosminexus V9 と互換性のある J2EE サーバを構築したい場合に選択します。Cosminexus V9 以前のバージョンから引き継ぐアプリケーションの開発に適しています。ただし，Cosminexus V11 以降で追加された機能は使用できません。

4. 次の項目を指定します。

項目名	指定値
ランタイム環境の種類を選択	J2EE サーバのランタイム・タイプを選択します。 ここでは，［Cosminexus］－［Cosminexus J2EE］を選択します。

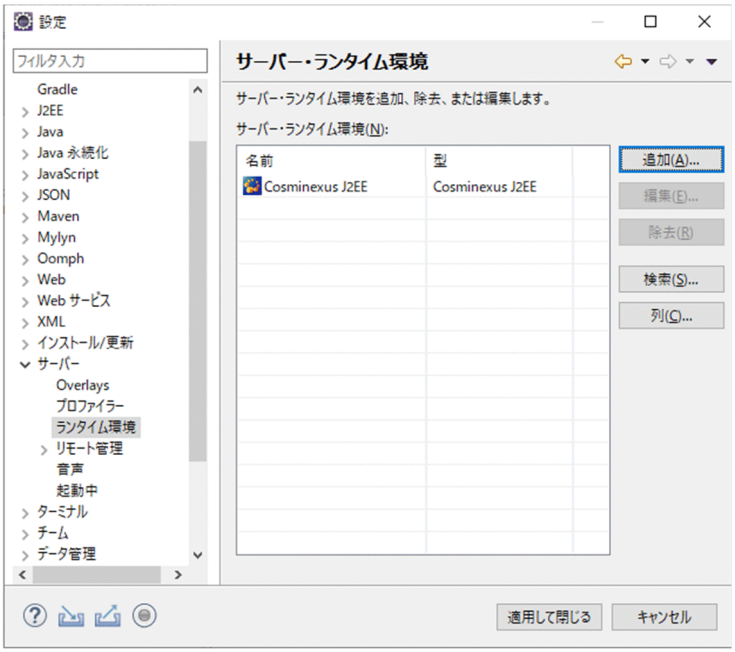
5. ［次へ］ボタンをクリックします。

［J2EE サーバー・ランタイム］ページが表示されます。ランタイム名には，Cosminexus J2EE がデフォルトで表示されます。



6. [終了] ボタンをクリックします。

[サーバー・ランタイム環境] ページの [サーバー・ランタイム環境] に，指定した Cosminexus J2EE サーバランタイムが登録されます。



7. [OK] ボタンをクリックします。

4.3.1 サーバランタイム作成時の注意事項

サーバランタイムを作成する場合は，次の点に注意してください。

- ［インストール済みの JRE］ページで Developer で提供されている JDK が登録されていない場合，Cosminexus J2EE サーバランタイムを作成するときに自動的に登録されます。ただし，次の手順でサーバランタイムを登録すると，Developer で提供されている JDK が自動登録されません。

  1. ［設定］ダイアログの左ペインで［Java］－［インストール済みの JRE］を選択して［インストール済みの JRE］ページを表示します。
  2. ［設定］ダイアログの左ペインで［サーバー］－［ランタイム環境］を選択して［サーバー・ランタイム環境］ページを表示します。
  3. J2EE サーバランタイムを登録します。

  この手順で Cosminexus J2EE サーバランタイムを登録した場合は，登録した Cosminexus J2EE サーバランタイムをいったん削除してください。そのあと，［設定］ダイアログを開き，［インストール済みの JRE］ページを表示しないで J2EE サーバランタイムを登録してください。

  また，Developer をアンインストールした場合，自動的に登録された Developer で提供されている JDK は残ります。この場合，必要に応じて Developer で提供されている JDK を削除してください。
- ［設定］ダイアログの［サーバー・ランタイム環境］ページで，［検索］ボタンをクリックしても Cosminexus J2EE サーバランタイムは検索されません。
- Cosminexus J2EE サーバランタイムをサーバまたはプロジェクトで使用している場合，［編集］ボタン，または［除去］ボタンを選択しないでください。

4.4　Eclipse プロジェクトの作成

Eclipse を使用して，アプリケーションの開発に必要な次のプロジェクトを作成できます。

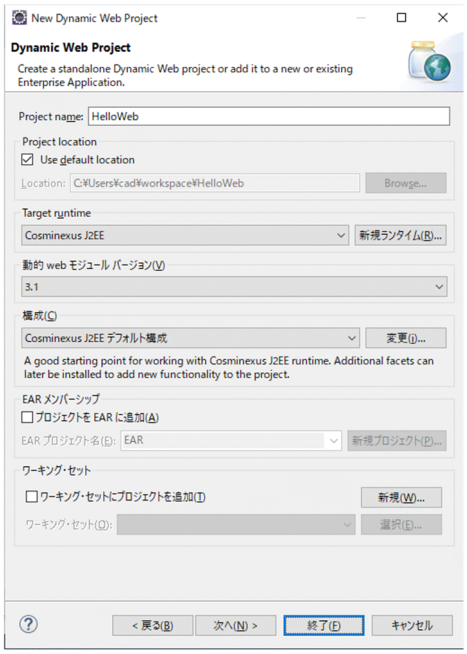
- 動的 Web プロジェクト
- EJB プロジェクト
- ユーティリティプロジェクト
- エンタープライズアプリケーションプロジェクト

4.4.1　動的 Web プロジェクトの作成

動的 Web プロジェクトを作成する手順を次に示します。

1. Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。
   ［新規］ダイアログが表示されます。

2. ［新規］ダイアログで，［Web］－［動的 Web プロジェクト］を選択し，［次へ］ボタンをクリックします。
   ［New Dynamic Web Project］ダイアログが表示されます。



3. ［Dynamic Web Project］ページで，次の項目を指定します。

項目名	説明
Project name※1	作成する動的 Web プロジェクトのプロジェクト名を指定します。

項目名	説明
Target runtime	「Cosminexus J2EE」または「Cosminexus J2EE V9 互換モード」のサーバランタイムを指定します。
動的 web モジュールバージョン	サーバランタイムが「Cosminexus J2EE」の場合は 2.3，2.4，2.5，3.0，3.1，4.0，5.0※2 の中から，「Cosminexus J2EE V9 互換モード」の場合は 2.3，2.4，2.5，3.0 の中から，開発するアプリケーションに合わせて指定します。
構成	プロジェクト構成を指定します。

注※1

半角英数字，およびアンダースコア（_）だけを使用します。

注※2

Dynamic web module バージョンの 5.0 は Jakarta EE 仕様となります。5.0 を指定したアプリケーションの使用方法については「4.6　Jakarta EE を使用したアプリケーションの開発」を参照してください。

ポイント

インストールした JDK のバージョンに関係なく，［構成］の［Java］のバージョンは「11」が選択されています。JDK17 がインストールされた環境で，［Java］のバージョンの「17」を選択する場合は，次のどちらかをしてください。

- ［構成］を変更する
- プロジェクト作成後に，プロジェクトのプロパティから［プロジェクト・ファセット］を選択し，［Java］のバージョンを変更する

必要に応じて，次の項目を指定してください。

項目名	説明
Use default location	プロジェクトルートの作成にデフォルト・ロケーションを使用するかどうかを指定します。
Location	［Use default location］をチェックしない場合に，任意のロケーションのパスを指定します。［Browse］ボタンで表示される［フォルダーの参照］ダイアログからも指定できます。
プロジェクトを EAR に追加	作成する動的 Web プロジェクトを EAR プロジェクトに追加するかどうかを指定します。
EAR プロジェクト名	［プロジェクトを EAR に追加］をチェックする場合に，追加する EAR プロジェクトを指定します。［新規プロジェクト］ボタンで表示される［New EAR Application Project］ダイアログからも指定できます。
ワーキング・セットにプロジェクトを追加	ワーキング・セットにプロジェクトを追加するかどうかを指定します。
ワーキング・セット	［ワーキング・セットにプロジェクトを追加］をチェックする場合に，追加するワーキング・セットを指定します。［選択］ボタンで表示される［ワーキング・セットの選択］ダイアログからも指定できます。

4. ［次へ］ボタンをクリックします。

［Java］ページが表示されます。

5. Java のビルドに関する情報を設定し，［次へ］ボタンをクリックします。

［Web モジュール］ページが表示されます。

6. Web モジュール設定を構成し，［終了］ボタンをクリックします。

［プロジェクト・エクスプローラー］ビューに，作成した動的 Web プロジェクトが表示されます。作成した動的 Web プロジェクトには，サーバランタイムの種類に対応する J2EE ライブラリが追加されます。サーバランタイムの種類と追加される J2EE ライブラリを次の表に示します。

項番	サーバランタイム	追加される J2EE ライブラリ
1	Cosminexus J2EE※	• javaee-api.jar（<Developer のインストールディレクトリ>¥CC¥javaee¥1100¥lib¥javaee-api.jar） • hjdk.tpb.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib） • hjdk.act.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib）
2	Cosminexus J2EE V9 互換モード	• cjjaxws.jar（<Developer のインストールディレクトリ>¥jaxws¥lib） • csmjaxb.jar（<Developer のインストールディレクトリ>¥jaxp¥lib） • csmjaxp.jar（<Developer のインストールディレクトリ>¥jaxp¥lib） • j2ee-javax.jar（<Developer のインストールディレクトリ>¥CC¥client¥lib） • hjdk.tpb.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib） • hjdk.act.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib）

注※

Cosminexus J2EE サーバランタイムを使用する場合，J2EE サーバの機能によって表に含まれないライブラリも自動的に追加されます。

4.4.2 EJB プロジェクトの作成

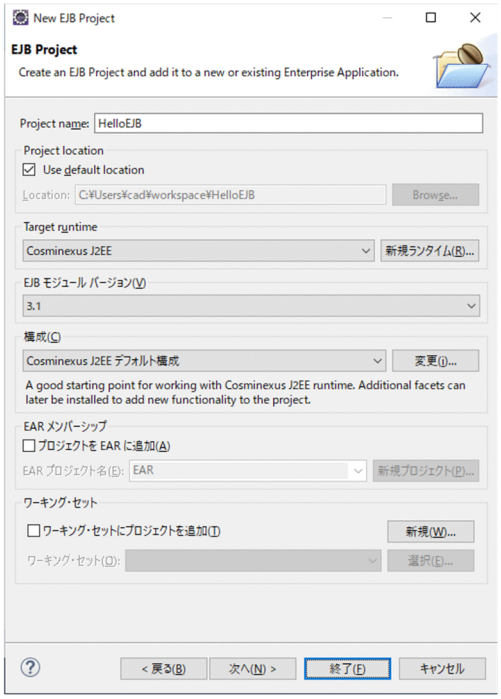
EJB プロジェクトを作成する手順を次に示します。

1. Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。

［新規］ダイアログが表示されます。

2. ［新規］ダイアログで，［EJB］－［EJB プロジェクト］を選択し，［次へ］ボタンをクリックします。

［New EJB Project］ダイアログが表示されます。



3. [EJB Project] ページで，次の項目を指定します。

項目名	説明
Project name※	作成する EJB プロジェクトのプロジェクト名を指定します。
Target runtime	「Cosminexus J2EE」または「Cosminexus J2EE V9 互換モード」のサーバランタイムを指定します。
EJB モジュールバージョン	2.0，2.1，3.0，3.1 の中から，開発するアプリケーションに合わせて指定します。Jakarta EE 仕様である 4.0 以降のバージョンは指定できません。
Configuration	プロジェクト構成を指定します。

注※

半角英数字，およびアンダースコア（_）だけを使用します。

ポイント

インストールした JDK のバージョンに関係なく，[構成] の [Java] のバージョンは「11」が選択されています。JDK17 がインストールされた環境で，[Java] のバージョンの「17」を選択する場合は，次のどちらかをしてください。

- [構成] を変更する
- プロジェクト作成後に，プロジェクトのプロパティから [プロジェクト・ファセット] を選択し，[Java] のバージョンを変更する

必要に応じて，次の項目を指定してください。

項目名	説明
Use default location	プロジェクトルートの作成にデフォルト・ロケーションを使用するかどうかを指定します。
Location	[Use default location] をチェックしない場合に，任意のロケーションのパスを指定します。[Browse] ボタンで表示される［フォルダーの参照］ダイアログからも指定できます。
プロジェクトを EAR に追加	作成する EJB プロジェクトを EAR プロジェクトに追加するかどうかを指定します。
EAR プロジェクト名	[プロジェクトを EAR に追加] をチェックする場合に，追加する EAR プロジェクトを指定します。[新規プロジェクト] ボタンで表示される［New EAR Application Project］ダイアログからも指定できます。
ワーキング・セットにプロジェクトを追加	ワーキング・セットにプロジェクトを追加するかどうかを指定します。
ワーキング・セット	[ワーキング・セットにプロジェクトを追加] をチェックする場合に，追加するワーキング・セットを指定します。[選択] ボタンで表示される［ワーキング・セットの選択］ダイアログからも指定できます。

4. [次へ] ボタンをクリックします。

[Java] ページが表示されます。

5. Java のビルドに関する情報を設定し，[次へ] ボタンをクリックします。

[EJB Module] ページが表示されます。

6. EJB モジュール設定を構成し，[終了] ボタンをクリックします。

[プロジェクト・エクスプローラー] ビューに，作成した EJB プロジェクトが表示されます。作成した EJB プロジェクトには，サーバランタイムの種類に対応する J2EE ライブラリが追加されます。サーバランタイムの種類と追加される J2EE ライブラリを次の表に示します。

項番	サーバランタイム	追加される J2EE ライブラリ
1	Cosminexus J2EE※	• javaee-api.jar（<Developer のインストールディレクトリ>¥CC¥javaee¥1100¥lib¥javaee-api.jar） • hjdk.tpb.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib） • hjdk.act.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib）
2	Cosminexus J2EE V9 互換モード	• cjjaxws.jar（<Developer のインストールディレクトリ>¥jaxws¥lib） • csmjaxb.jar（<Developer のインストールディレクトリ>¥jaxp¥lib） • csmjaxp.jar（<Developer のインストールディレクトリ>¥jaxp¥lib） • j2ee-javax.jar（<Developer のインストールディレクトリ>¥CC¥client¥lib） • hjdk.tpb.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib） • hjdk.act.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib）

注※

Cosminexus J2EE サーバランタイムを使用する場合，J2EE サーバの機能によって表に含まれないライブラリも自動的に追加されます。

4.4.3 ユーティリティプロジェクトの作成

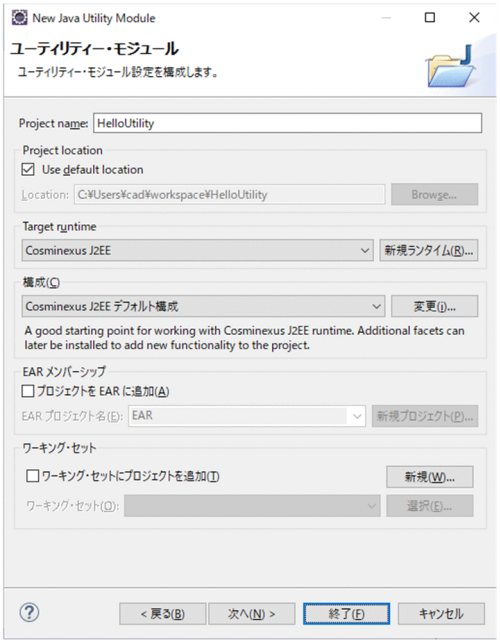
ユーティリティプロジェクトを作成する手順を次に示します。

1. Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。

   ［新規］ダイアログが表示されます。

2. ［新規］ダイアログで，［J2EE］－［ユーティリティー・プロジェクト］を選択し，［次へ］ボタンをクリックします。

   ［New Java Utility Module］ダイアログが表示されます。



3. ［ユーティリティー・モジュール］ページで，次の項目を指定します。

項目名	説明
Project name※	作成するユーティリティプロジェクトのプロジェクト名を指定します。
Target runtime	「Cosminexus J2EE」または「Cosminexus J2EE V9 互換モード」のサーバランタイムを指定します。
Configuration	プロジェクト構成を指定します。

注※

半角英数字，およびアンダースコア（_）だけを使用します。

ポイント

インストールした JDK のバージョンに関係なく，［構成］の［Java］のバージョンは「11」が選択されています。JDK17 がインストールされた環境で，［Java］のバージョンの「17」を選択する場合は，次のどちらかをしてください。

- ［構成］を変更する
- プロジェクト作成後に，プロジェクトのプロパティから［プロジェクト・ファセット］を選択し，［Java］のバージョンを変更する

必要に応じて，次の項目を指定してください。

項目名	説明
Use default location	プロジェクトルートの作成にデフォルト・ロケーションを使用するかどうかを指定します。
Location	［Use default location］をチェックしない場合に，任意のロケーションのパスを指定します。［Browse］ボタンで表示される［フォルダーの参照］ダイアログからも指定できます。
プロジェクトを EAR に追加	作成するユーティリティプロジェクトを EAR プロジェクトに追加するかどうかを指定します。
EAR プロジェクト名	［プロジェクトを EAR に追加］をチェックする場合に，追加する EAR プロジェクトを指定します。［新規プロジェクト］ボタンで表示される［New EAR Application Project］ダイアログからも指定できます。
ワーキング・セットにプロジェクトを追加	ワーキング・セットにプロジェクトを追加するかどうかを指定します。
ワーキング・セット	［ワーキング・セットにプロジェクトを追加］をチェックする場合に，追加するワーキング・セットを指定します。［選択］ボタンで表示される［ワーキング・セットの選択］ダイアログからも指定できます。

4. ［次へ］ボタンをクリックします。

［Java］ページが表示されます。

5. Java のビルドに関する情報を設定し，［終了］ボタンをクリックします。

［プロジェクト・エクスプローラー］ビューに，作成したユーティリティプロジェクトが表示されます。作成したユーティリティプロジェクトには，サーバランタイムの種類に対応する J2EE ライブラリが追加されます。サーバランタイムの種類と追加される J2EE ライブラリを次の表に示します。

項番	サーバランタイム	追加される J2EE ライブラリ
1	Cosminexus J2EE※	• javaee-api.jar（<Developer のインストールディレクトリ>¥CC¥javaee¥1100¥lib¥javaee-api.jar） • hjdk.tpb.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib） • hjdk.act.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib）

項番	サーバランタイム	追加される J2EE ライブラリ
2	Cosminexus J2EE V9 互換モード	• cjjaxws.jar（<Developer のインストールディレクトリ>¥jaxws¥lib） • csmjaxb.jar（<Developer のインストールディレクトリ>¥jaxp¥lib） • csmjaxp.jar（<Developer のインストールディレクトリ>¥jaxp¥lib） • j2ee-javax.jar（<Developer のインストールディレクトリ>¥CC¥client¥lib） • hjdk.tpb.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib） • hjdk.act.jar（<Developer のインストールディレクトリ>¥jdk¥lib¥hcompatlib）

注※

Cosminexus J2EE サーバランタイムを使用する場合，J2EE サーバの機能によって表に含まれないライブラリも自動的に追加されます。

4.4.4 エンタープライズアプリケーションプロジェクトの作成

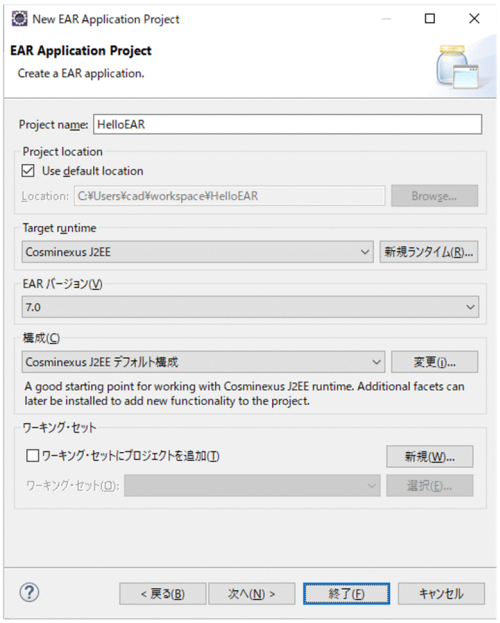
エンタープライズアプリケーションプロジェクトを作成する手順を次に示します。

1. Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。

   ［新規］ダイアログが表示されます。

2. ［新規］ダイアログで，［J2EE］－［エンタープライズ・アプリケーション・プロジェクト］を選択し，［次へ］ボタンをクリックします。

   ［New EAR Application Project］ダイアログが表示されます。



3. ［EAR Application Project］ページで，エンタープライズアプリケーションプロジェクトを定義します。

項目名	説明
Project name※	作成するエンタープライズアプリケーションプロジェクトのプロジェクト名を指定します。
Target runtime	「Cosminexus J2EE」または「Cosminexus J2EE V9 互換モード」のサーバランタイムを指定します。
EAR バージョン	サーバランタイムが「Cosminexus J2EE」の場合は 1.4，5.0，6.0，7.0，8.0 の中から，「Cosminexus J2EE V9 互換モード」の場合は 1.4，5.0，6.0 の中から，開発するアプリケーションに合わせて指定します。Jakarta EE 仕様である 9.0 以降のバージョンは指定できません。
Configuration	プロジェクト構成を指定します。

注※

半角英数字，およびアンダースコア（_）だけを使用します。

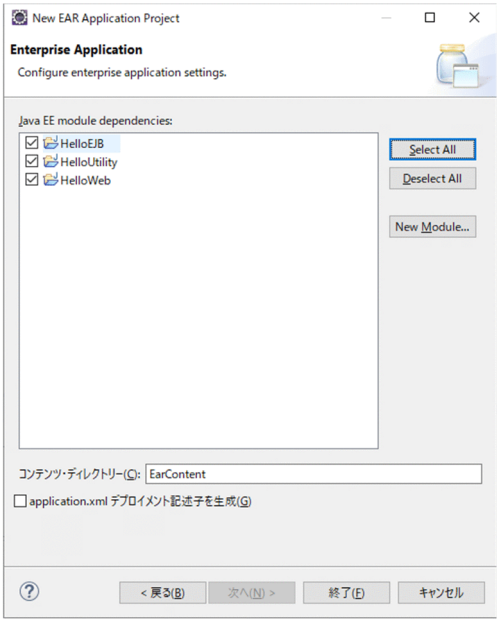
必要に応じて，次の項目を指定してください。

項目名	説明
Use default location	プロジェクトルートの作成にデフォルト・ロケーションを使用するかどうかを指定します。
Location	[Use default location] をチェックしない場合に，任意のロケーションのパスを指定します。[Browse] ボタンで表示される［フォルダーの参照］ダイアログからも指定できます。
ワーキング・セットにプロジェクトを追加	ワーキング・セットにプロジェクトを追加するかどうかを指定します。
ワーキング・セット	[ワーキング・セットにプロジェクトを追加] をチェックする場合に，追加するワーキング・セットを指定します。[選択] ボタンで表示される［ワーキング・セットの選択］ダイアログからも指定できます。

4. [次へ] ボタンをクリックします。

[Enterprise Application] ページが表示されます。

5. 新規エンタープライズアプリケーションの J2EE モジュールを選択，または追加して，[終了] ボタンをクリックします。



[プロジェクト・エクスプローラー] ビューに，作成したエンタープライズアプリケーションプロジェクトが表示されます。

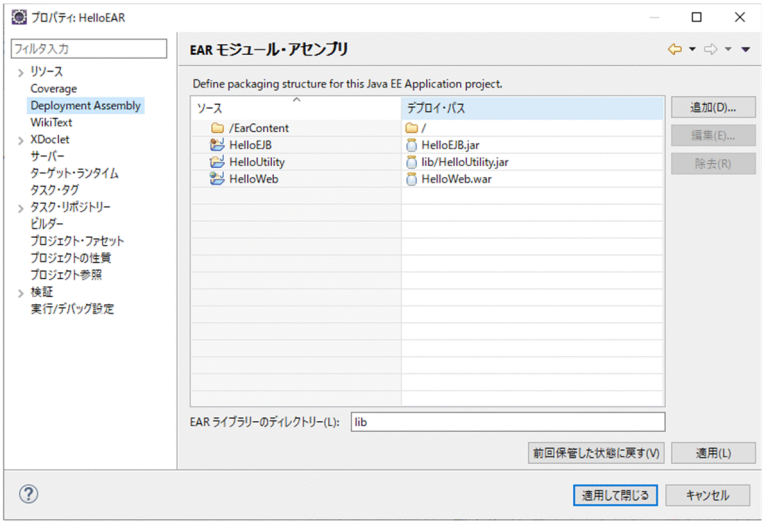
4.4.5 エンタープライズアプリケーションプロジェクトのモジュールの変更手順

エンタープライズアプリケーションプロジェクトを作成したあとで，Java EE モジュールの構成を変更できます。

次に，J2EE アプリケーションのモジュール構成を変更する手順を示します。

(1) モジュールプロジェクトを追加する場合

1. [プロジェクト・エクスプローラー] ビューで，変更対象のエンタープライズアプリケーションプロジェクトを選択します。
2. Eclipse のメニューから［プロジェクト］－［プロパティ］を選択します。
   [プロパティ： <プロジェクト名>] ダイアログが表示されます。
3. [プロパティ： <プロジェクト名>] ダイアログの左ペインで［Deployment Assembly］を選択します。
   [Ear Module Assembly] ページが表示されます。



4. [追加] ボタンをクリックします。
   [新規アセンブリーディレクティブ] ダイアログが表示されます。

5. [ディレクティブの種類を選択] ページのリストで [プロジェクト] を選択し，[次へ] ボタンをクリックします。
   [プロジェクト] ページが表示されます。

6. [プロジェクト] ページのリストで，追加するプロジェクトを選択し，[終了] ボタンをクリックします。

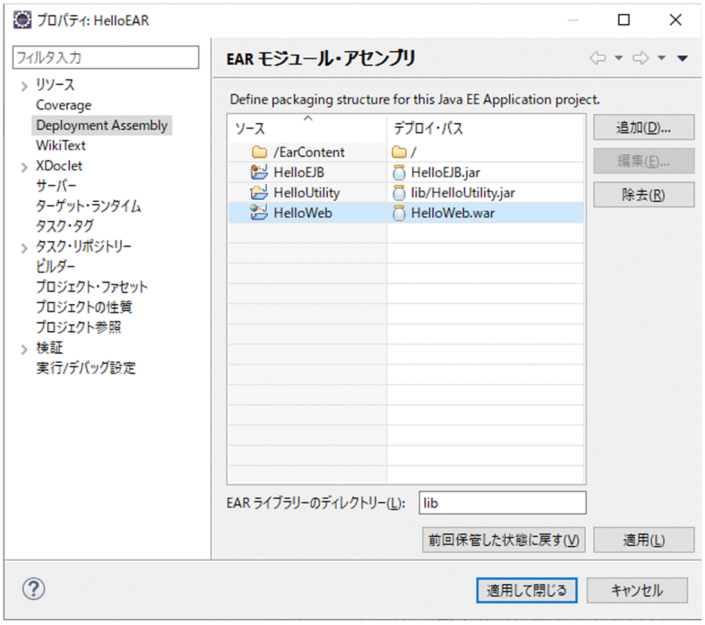
7. [プロパティ： <プロジェクト名>] ダイアログで，[OK] ボタンをクリックします。
   変更内容が反映されます。

(2) モジュールプロジェクトを削除する場合

1. [プロジェクト・エクスプローラー] ビューで，変更対象のエンタープライズアプリケーションプロジェクトを選択します。

2. Eclipse のメニューから [プロジェクト] − [プロパティ] を選択します。
   [プロパティ： <プロジェクト名>] ダイアログが表示されます。

3. [プロパティ： <プロジェクト名>] ダイアログの左ペインで [Deployment Assembly] を選択します。
   [Ear Module Assembly] ページが表示されます。



4. リストから削除するプロジェクトを選択し，［除去］ボタンをクリックします。

5. ［プロパティ： <プロジェクト名>］ダイアログで，［OK］ボタンをクリックします。
   変更内容が反映されます。

4.4.6　プロジェクト作成時の注意事項

プロジェクトを作成する場合は，次の点に注意してください。

- エンタープライズアプリケーションプロジェクトで配備記述子（application.xml）を省略した場合，プロジェクト名が J2EE アプリケーション名となります。また，EJB プロジェクト，および，動的 Web プロジェクトを単体で公開する場合もプロジェクト名が J2EE アプリケーション名となります。プロジェクト名には，半角英数字，または，アンダースコア（_）だけを使用してください。
- エンタープライズアプリケーションプロジェクトで配備記述子（application.xml）を作成した場合，<display-name>の値には，半角英数字，プラス（+），ハイフン（-），ピリオド（.），キャレット（^），またはアンダースコア（_）以外の文字は使用しないでください。また，名前の先頭または名前の末尾にピリオド(.)を指定した名前，ピリオド(.)だけの名前は指定できません。
- WTP コネクタがサポートしないプロジェクトは，WTP コネクタで公開できません。EJB プロジェクト，動的 Web プロジェクト，およびエンタープライズアプリケーションプロジェクトが，WTP コネクタがサポートしないプロジェクトに依存する場合は，WTP コネクタがサポートしないプロジェクトを手作業で Jar ファイルなどのアーカイブファイルにして，適切なフォルダに配置してください。

  例えば，動的 Web プロジェクトが依存する Java プロジェクトを J2EE アプリケーションに含める場合は，Java プロジェクトから Jar ファイルを作成して，動的 Web プロジェクトの WEB-INF/lib フォルダに配置してください。

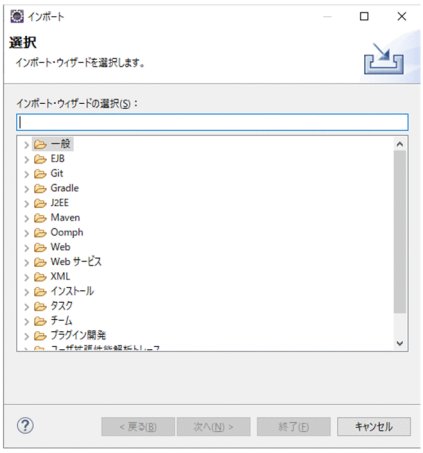
- ［ターゲット・ランタイム］に Cosminexus J2EE サーバランタイム以外を指定すると，誤動作する場合があります。
- 公開中のプロジェクト名を変更すると，エラーが発生するおそれがあります。
- プロジェクトを削除する場合は，プロジェクトを選択してから削除してください。プロジェクトを選択していない状態で削除すると，エラーが発生するおそれがあります。

4.5　リソースアダプタのインポート

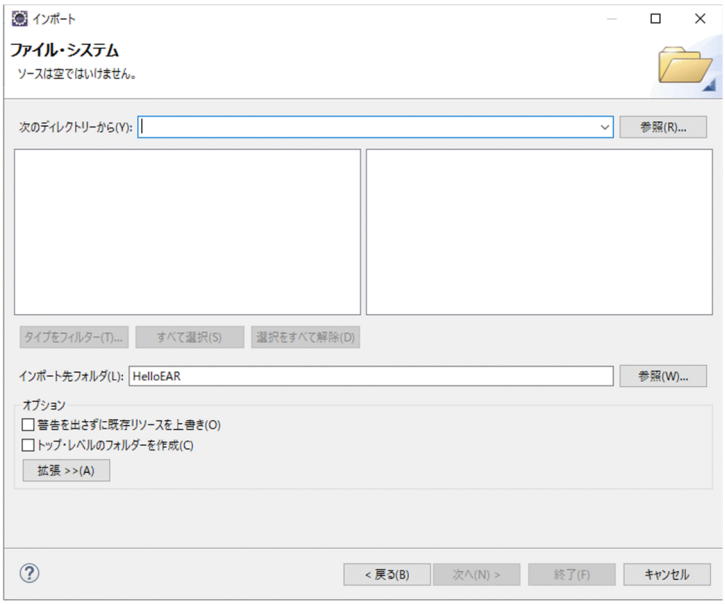
Eclipse を使った開発では，アプリケーションサーバの提供するリソースアダプタをエンタープライズアプリケーションプロジェクトにインポートできます。ここでは，Eclipse で作成したエンタープライズアプリケーションプロジェクトにリソースアダプタをインポートする手順を説明します。

なお，リソースアダプタをインポートする前に，必ずエンタープライズアプリケーションプロジェクトを作成してください。エンタープライズアプリケーションプロジェクトの作成手順については，「4.4.4　エンタープライズアプリケーションプロジェクトの作成」を参照してください。

1. [プロジェクト・エクスプローラー] ビューでインポート先のエンタープライズアプリケーションプロジェクトを選択して，コンテキストメニューから［インポート］－［インポート］を選択します。
   ［インポート］ダイアログが表示されます。



2. [General] －［ファイル・システム］を選択して，［次へ］ボタンをクリックします。
   ［ファイル・システム］ページが表示されます。



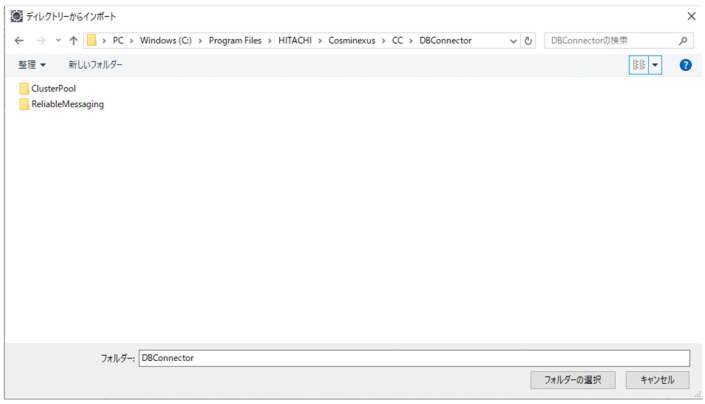
3. [次のディレクトリーから] の［参照］ボタンをクリックします。

［ディレクトリーからインポート］ダイアログが表示されます。

4. インポートするリソースアダプタが含まれるディレクトリを選択します。

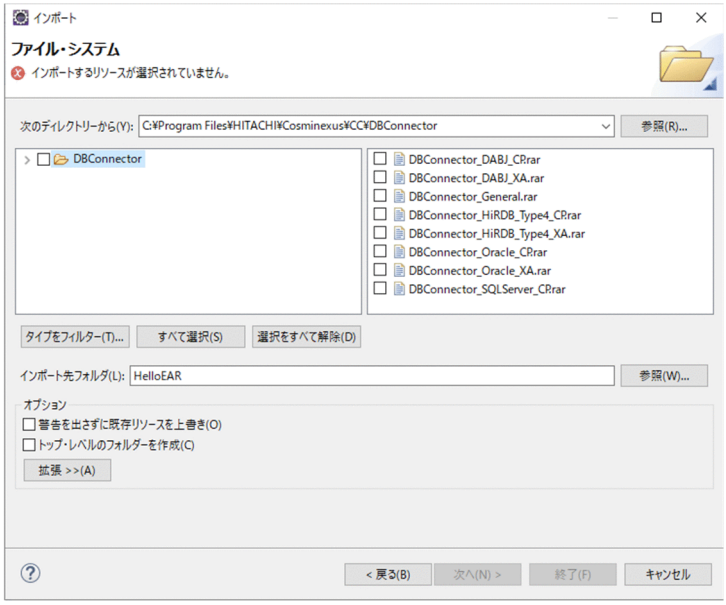
リソースアダプタは次のディレクトリに格納されています。

<Developerのインストールディレクトリ>¥CC¥DBConnector¥



5. ［OK］ボタンをクリックします。

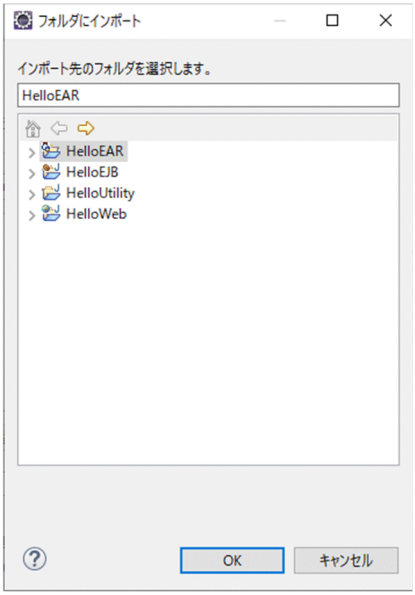
［ディレクトリーからインポート］ダイアログが閉じて，［ファイル・システム］ページの左のリストボックスに選択したディレクトリが表示されます。右のリストボックスに選択したディレクトリに含まれるファイルの一覧が表示されます。



6. インポートするファイルのチェックボックスにチェックを入れます。

7. ［インポート先フォルダ］の［参照］ボタンをクリックします。

［フォルダにインポート］ダイアログが表示されます。



8. インポート先のエンタープライズアプリケーションプロジェクトのルートフォルダを選択して，［OK］ボタンをクリックします。

   ［フォルダにインポート］ダイアログが閉じます。

9. ［ファイル・システム］ページの［終了］ボタンをクリックします。

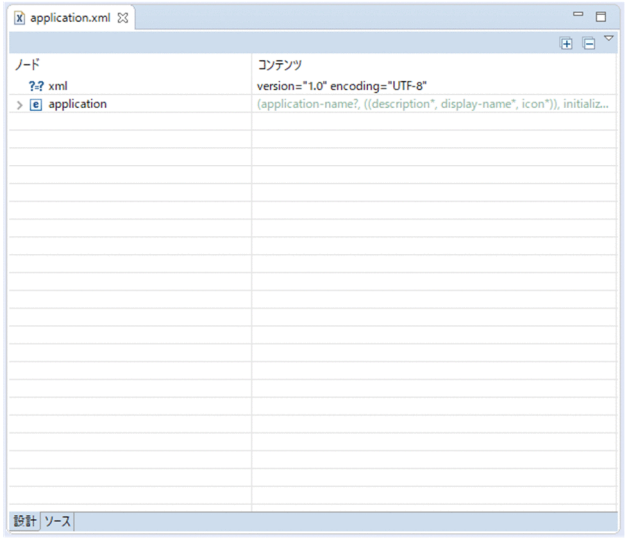
   手順 8.で選択したフォルダに，リソースアダプタがインポートされます。

10. ［プロジェクト・エクスプローラー］ビューで［<インポート先のエンタープライズアプリケーションプロジェクト>］－［<コンテンツディレクトリ>］－［META-INF］－［application.xml］を選択して，コンテキストメニューから［開く］を選択します。

    ［XML エディタ］に application.xml が表示されます。

11. ［設計］タブを選択します。

    ［設計］タブの画面が表示されます。



12. 画面上に表示されている「application」を展開します。

13.「module」を選択して，コンテキストメニューから［後に追加］－［module］を選択します。

選択した「module」の後ろに，新しい「module」が追加されます。

14. 追加された「module」を展開して，「connector」の値をインポートしたリソースアダプタのファイル名に変更します。

15. Eclipse のメニューから，［ファイル］－［保管］を選択します。

application.xml の変更内容が保管されて，エンタープライズアプリケーションプロジェクトにリソースアダプタがインポートされます。

リソースアダプタを J2EE アプリケーションに含めないでインポートする場合は，サーバ管理コマンドまたは運用管理ポータルを使用します。

サーバ管理コマンドでのインポート方法については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「4. リソースアダプタの設定」を参照してください。また，運用管理ポータルでのインポート方法については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「12. 論理サーバのアプリケーション管理」を参照してください。

4.6　Jakarta EE を使用したアプリケーションの開発

Jakarta EE を使用したアプリケーションの開発に必要な設定と注意事項について説明します。

4.6.1　Jakarta EE を使用するための設定

Jakarta EE を使用するアプリケーションを開発する場合に必用な設定を次に示します。

(1)　クラスパスの設定

Cosmienxus が提供する jar ファイルをクラスパスに追加します。この作業は，Jakarta EE を使用する各プロジェクトに対して実施してください。追加する jar ファイルのパスを次に示します。

```
<Application Serverのインストールディレクトリ>¥CC¥javaee¥1100¥lib¥jakartaee-api.jar
```

プロジェクトへのクラスパスの追加手順を次に示します。

1. [プロジェクト・エクスプローラー］ビューで，クラスパスを追加するプロジェクトを選択します。
2. Eclipse のメニューから［プロジェクト］－［プロパティ］を選択します。または，プロジェクトを右クリックして［プロパティ］を選択します。
   ［プロパティ：<プロジェクト名>］ダイアログが表示されます。
3. [プロパティ：<プロジェクト名>］ダイアログの左ペインで［Java のビルドパス］を選択します。
   ［Java のビルドパス］ページが表示されます。
4. [ライブラリー］タブの［外部 jar の追加］をクリックして，［JAR の選択］ダイアログから追加する jar ファイルのパスを指定します。

(2)　J2EE サーバへのプロパティ追加

開発したアプリケーションを J2EE サーバに公開（デプロイ）するためには，J2EE サーバのパッケージ名変換機能を有効にする必要があります。

パッケージ名変換機能を有効にする場合は，JavaVM の起動パラメタの「拡張起動パラメタ」に次のプロパティを追加してください。

```
-Dejbserver.server.packageMigration.enabled=true
```

プロパティの追加は，運用管理ポータルの論理 J2EE サーバの定義にある，［起動パラメタの設定］画面で実施してください。

詳細は，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.8.23　起動パラメタの設定（J2EE サーバ)」を参照してください。

また，パッケージ名変換機能およびプロパティの詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「20. パッケージ名変換機能」を参照してください。

4.6.2 Jakarta EE を使用する場合の注意事項

パッケージ名変換機能はデプロイ時だけ適用されます。リロード時には適用されません。そのため，Jakarta EE パッケージを使用した JSP ファイルなどを更新して，リロードした場合はエラーとなります。

Developer が提供する開発環境で，JSP など，パッケージ名変換機能のテキストファイルの変換対象となるファイルの開発，更新を行う場合は Jakarta EE パッケージではなく Java EE パッケージに改修してから実施してください。

そのほか，パッケージ名変換機能に関する注意事項については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「20. パッケージ名変換機能」を参照してください。

5 定義情報の編集

J2EE アプリケーションの定義情報のうち，Java EE 標準仕様についての定義を変更する場合は，DD を編集する必要があります。また，アプリケーションサーバ独自の定義情報を設定する場合は，cosminexus.xml を作成および編集する必要があります。この章では，各定義ファイルを編集する際の注意事項，および編集方法について説明します。

5.1　定義情報と編集するファイルの種類

Developer を使用した J2EE アプリケーション開発で J2EE アプリケーションの定義情報を変更する場合は，次に示す DD を編集します。

Java EE 標準仕様の定義情報を変更する場合

- application.xml
  編集時の注意事項については，「5.2.2　application.xml 編集時の注意事項」を参照してください。
- ejb-jar.xml
  編集時の注意事項については，「5.2.3　ejb-jar.xml 編集時の注意事項」を参照してください。
- web.xml
  編集時の注意事項については，「5.2.4　web.xml 編集時の注意事項」を参照してください。

アプリケーションサーバ独自の定義情報を変更する場合

- cosminexus.xml
  cosminexus.xml の作成方法および編集方法については，「5.3　cosminexus.xml の作成と編集」を参照してください。

5.2　DD の編集

各 DD 編集時の注意事項を説明します。また，アプリケーションサーバでサポートする各 DD のバージョンも説明します。

5.2.1　アプリケーションサーバでサポートする DD について

アプリケーションサーバでサポートする DD の範囲については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「16.2　属性の管理」の表 16-2 を参照してください。

5.2.2　application.xml 編集時の注意事項

ここでは，application.xml 編集時の注意事項を説明します。

(1)　<display-name>タグの設定

- デプロイ後に変更すると，J2EE サーバ上の J2EE アプリケーションの入れ替え，およびアンデプロイが適切に実行されません。
- <display-name>タグの設定値は，作業ディレクトリ中のディレクトリ名として使用されます。作業ディレクトリのパス長が OS の上限に達しないように<display-name>タグを指定してください。作業ディレクトリのパス長の見積もりについては，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「付録 C.1　J2EE サーバの作業ディレクトリ」を参照してください。
- アプリケーション属性ファイルで，<hitachi-application-property><scheduling-unit>に Application を設定していて，かつ<hitachi-application-property><scheduling><queue-name>を省略する場合，application.xml の<display-name>には<queue-name>で指定可能な範囲の値を指定してください。

(2)　<context-root>タグの設定

<context-root>タグにはコンテキストルートが設定されます。<context-root>タグの設定値は，作業ディレクトリ中のディレクトリ名として使用されます。作業ディレクトリのパス長が OS の上限に達しないように<context-root>タグを指定してください。作業ディレクトリのパス長の見積もりについては，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「付録 C.1　J2EE サーバの作業ディレクトリ」を参照してください。

(3)　DD 作成時の注意事項

各プロジェクトの DD では Processing Instruction，XInclusion，および名前空間接頭辞を記述できません。記述しても，値が正しく読み込まれません。

5.2.3　ejb-jar.xml 編集時の注意事項

ここでは，ejb-jar.xml 編集時の注意事項を説明します。

(1)　DTD に従っていない ejb-jar.xml の取り扱い

必須項目の指定がない，または指定順序の不正などで，EJB-JAR の DTD（ejb-jar_1_1.dtd，ejb-jar_2_0.dtd，ejb-jar_2_1.xsd，ejb-jar_3.0.xsd，および ejb-jar_3.1.xsd）に従っていない DD は，EJB-JAR のインポート時にエラーとなり，インポートできません。インポートできない場合は，ejb-jar.xml の設定を見直してください。

(2)　CMP フィールドおよび CMR フィールドの命名規則

Entity Bean の CMP では，アンダースコア（_）で始まる CMP フィールド名および CMR フィールド名を指定できません。

(3)　EJB QL での 2 バイトコードの使用について

EJB QL では 2 バイトコードを使用しないでください。

(4)　セキュリティロールリファレンスの設定について

<security-role-ref>タグに含まれる role-link には，<security-role>タグに含まれる role-name で指定したロール名を指定します。

(5)　クエリメソッドのタグを記述するときの注意事項

ejb-jar.xml の<query-method>タグに含まれる<method-name>タグでは，「*（アスタリスク）」を記述できません。<query-method>タグに「*」を指定した ejb-jar.xml を含む EJB-JAR は，インポートできません。

(6)　アブストラクトスキーマ名の指定時の注意事項

<abstract-schema-name>タグは，同じファイルにあるほかの<abstract-schema-name>タグで指定する名称や，cmp-field，cmr-field で指定する名称と重複できません。また，<abstract-schema-name>タグには EJB QL の予約語を使用できません。

(7)　<display-name>タグ編集時の注意事項

<ejb-jar>タグに含まれる<display-name>タグは作業ディレクトリ中のファイル名として使用されます。作業ディレクトリのパス長が OS の上限に達しないように<display-name>タグを指定してください（デフォルトは EJB-JAR ファイル名を基に付けられます）。作業ディレクトリのパス長の見積もりについては，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「付録 C.1　J2EE サーバの作業ディレクトリ」を参照してください。

また，CMR を利用時，ejb-jar.xml の<ejb-jar><enterprise-beans><entity><display-name>には，Java 識別子に使用できる文字列を指定してください。

(8) <ejb-client-jar>タグの使用について

ejb-jar.xml に記述する DD の<ejb-client-jar>タグの機能をサポートしていません。

(9) Relationship の設定の注意事項

ejb-jar.xml に<relationships>タグを設定する場合は，二つの<ejb-relationship-role>タグのどちらか一方に<cmr-field>タグを設定します。

(10) DD 作成時の注意事項

各プロジェクトの DD では Processing Instruction，XInclusion，および名前空間接頭辞を記述できません。記述しても，値が正しく読み込まれません。

(11) <ejb-jar><enterprise-beans><session><ejb-name>タグ編集時の注意事項

アプリケーション属性ファイルで，<hitachi-application-property><scheduling-unit>に Bean を設定していて，かつ Session Bean 属性ファイルの<hitachi-session-bean-property><runtime><scheduling><queue-name>を省略している場合，ejb-jar.xml の<ejb-name>には，<queue-name>で指定可能な範囲の値を定義してください。

5.2.4 web.xml 編集時の注意事項

ここでは，web.xml 編集時の注意事項を説明します。

(1) セッションタイムアウトの設定時の注意事項

セッションタイムアウトを設定するときの注意事項を示します。

- セッションタイムアウトの設定値

  セッションタイムアウトを無期限に設定しないでください。無期限に設定した場合，セッション情報を保持する領域が解放されないため，メモリを消費し続けます。

- web.xml の<session-timeout>タグの指定

  web.xml の<session-timeout>タグに指定する値の有効範囲は，-35,791,394～35,791,394 です。この範囲内の値を指定してください。

(2) アクセスするURLパターンの定義の注意事項

web.xml上の<servlet>タグに含まれる<init-param>，<load-on-startup>，および<security-role-ref>タグの指定は，<servlet-mapping>タグに定義したURLパターンに該当するサーブレットまたはJSPファイルにアクセスした場合にだけ有効になります。このため，マッピング定義なしで直接JSPファイルのパスをURL指定した場合，または/servlet/のプリフィックスでサーブレットクラスをURLに指定して実行した場合は有効になりません。

(3) セキュリティロール使用時の設定について

web.xmlの<security-role>タグを使用する場合は，J2EEサーバモードの実行環境で，サーバ管理コマンドでセキュリティロールのリファレンスを解決する必要があります。

(4) エラーページ設定時の注意事項

Servlet 2.2仕様およびServlet 2.3仕様の場合，web.xmlで<error-page>タグを指定すると，そのエラーページが表示されるときに返されるステータスコードが200になります。ステータスコード401のエラーページを指定した場合，401のステータスコードが200になってしまうため，Basic認証と併用できません。Basic認証を使用する場合には，ステータスコード401用のエラーページを指定しないでください。

(5) <run-as>タグとWebコンテナの認証の関連について

web.xmlの<servlet>タグ要素に指定する<run-as>タグは，指定されたサーブレットまたはJSPからEnterprise Beanを呼び出すときに使用されるものであり，Webコンテナでの認証とは無関係です。

このため，Webコンテナでの認証結果を参照するためのjavax.servlet.http.HttpServletRequestクラスのisUserInRoleメソッドやgetUserPrincipalメソッドには影響しません。例えば，サーブレットまたはJSPファイルから<run-as>タグに記述したロール名を引数に指定したjavax.servlet.http.HttpServletRequestのisUserInRoleメソッドを呼び出しても，戻り値はfalseとなります。

(6) <load-on-startup>タグ指定時の注意事項

<load-on-startup>タグに空文字を指定（<load-on-startup></load-on-startup>または<load-on-startup/>と指定）したサーブレットおよびJSPは，<load-on-startup>タグに2,147,483,647を指定されたものとしてWebモジュールのデプロイ時にロードされます。

(7) <display-name>タグ編集時の注意事項

<web-app>タグ下に含まれる<display-name>タグは，作業ディレクトリ中のファイル名として使用されます。作業ディレクトリのパス長がプラットフォームで規定されているパス長の上限に達しないようにdisplay-nameを指定してください（デフォルトはWARファイル名を基に付けられます）。作業ディレクトリのパス長の見積もりについては，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「付録C.1　J2EEサーバの作業ディレクトリ」を参照してください。

(8) web.xml の記述内容とサーバの動作

Servlet 2.3 に対応した J2EE アプリケーションの場合の，web.xml の記述内容と J2EE サーバの動作を次の表に示します。

表 5-1 web.xml の記述内容とサーバの動作

項番	web.xml の記述内容	サーバの動作
1	プロパティに webserver.xml.validate=false を設定した場合に，DTD の仕様で必須となっているタグを記述しないとき	WAR ファイルのインポート時にエラーとなります。
2	<filter>タグに含まれる<filter-name>タグ要素を空要素にして，<filter-mapping>タグに含まれる<filter-name>タグ要素も空要素にした場合	実行されません。
3	<listener>タグに含まれる<listener-class>タグ要素を空要素にした場合	<listener>タグの指定を無視して正常に実行されます。
4	プロパティに webserver.xml.validate=false を設定した場合に，DTD の仕様で一つだけ指定が許されているタグを複数記述したとき	WAR ファイルのインポート時にエラーになります。
5	要素が同じ<filter-mapping>タグを複数記述した場合	doFilter メソッドは 1 回だけ呼び出されます。
6	親タグ，その下に要素としてキーとなるタグ，およびそのほかの情報を持つタグを複数記述した場合に，キーとなるタグの要素が同じでそのほかの情報が異なるとき	最初に記述された親タグを有効にします。
7	プロパティに webserver.xml.validate=false を設定している場合に，DTD の仕様で規定されていない順序でタグを記述したとき	WAR ファイルのインポート時にエラーになります。
8	<jsp-file>タグに「/」で始まらない文字列を記述した場合	先頭に「/」を付けて，正常に実行されます。
9	次に示す<url-pattern>タグに，「*.」以外の「/」で始まらない文字列を記述した場合 • <servlet-mapping>-<url-pattern> • <security-constraint>-<web-resource-collection>-<url-pattern>	先頭に「/」を付けて，正常に実行されます。
10	<mime-mapping>タグに含まれる<extension>タグ要素に空文字を記述した場合	WAR ファイルのインポート時にエラーになります。
11	<mime-mapping>タグに含まれる<mime-type>タグ要素に空文字を記述した場合	WAR ファイルのインポート時にエラーになります。
12	<error-page>タグに含まれる<error-code>タグ要素に空文字を記述した場合	WAR ファイルのインポート時にエラーになります。
13	<transport-guarantee>タグに空文字を記述した場合	WAR ファイルのインポート時にエラーになります。
14	<form-login-page>タグまたは<form-error-page>タグに空文字を記述した場合	WAR ファイルのインポート時に設定が無視されます。

項番	web.xml の記述内容	サーバの動作
15	<env-entry-type>タグ，<ejb-ref-type>タグ，または<ejb-ref-type>タグに空文字を設定した場合	WAR ファイルのインポート時にエラーになります。
16	<security-constraint>タグを設定して<auth-method>タグを省略した場合	Basic 認証としてインポートされます。

(9) ゲートウェイ指定機能を使用する場合の注意事項

ゲートウェイ指定機能でスキームを https と見なすように設定した場合，Web サーバへのリクエストが http であっても https とみなされます。したがって，web.xml の<transport-guarantee>タグで INTEGRAL や CONFIDENTIAL を指定しても，https の URL へリダイレクトされません。

(10) <taglib-location>タグに指定したパスの大文字，小文字が異なる場合の動作

web.xml の<taglib-location>タグまたは JSP の taglib ディレクティブに指定したタグライブラリ・ディスクリプタ（TLD ファイル）のパスが，実際のパスと大文字，小文字が異なっている場合，Windows 上では正常に動作しますが，UNIX 上ではエラーになります。

Windows 上で動作していたユーザプログラムを UNIX に移行する場合は，web.xml の<taglib-location>タグまたは JSP の taglib ディレクティブに指定したタグライブラリ・ディスクリプタ（TLD ファイル）のパスが，実際のパスと大文字・小文字が異なっていないか確認してください。

(11) web.xml の DOCTYPE 宣言の注意事項

web.xml で DOCTYPE 宣言に内部サブセットを記述しないでください。Java EE 仕様で定義された DTD／XML スキーマだけを使用してください。

(12) DD 作成時の注意事項

各プロジェクトの DD では Processing Instruction，XInclusion，および名前空間接頭辞を記述できません。記述しても，値が正しく読み込まれません。

5.2.5 Servlet 2.4 以降で追加，変更された仕様についての注意事項（web.xml）

Servlet 2.4 以降で追加および変更された仕様を持つ web.xml をアプリケーションサーバ上で使用する場合の注意事項を示します。Servlet 2.4 以降の仕様については，該当する Servlet の仕様書を参照してください。

(1) Servlet 2.4 以降の仕様でサポートされない web.xml の要素

アプリケーションサーバでは，次に示す Servlet 2.4 以降の仕様の web.xml の要素は定義できません。定義した場合，デプロイ時にエラーになります。

- message-destination
- message-destination-ref
- service-ref

(2) <security-constraint>タグの設定

Servlet 2.4 以降の仕様では，web.xml の<security-constraint>タグに特別な指定をした場合の動作について追記されています。

アプリケーションサーバでは，Web アプリケーションのバージョンに関係なく，次の動作をします。

- <auth-constraint>タグのサブ要素，<role-name>タグに「*（アスタリスク）」を指定した場合，すべてのロールを許可します。
- <auth-constraint>タグを指定しない場合，Web コンテナは認証しないでリクエストを許可します。
- <transport-guarantee>タグを指定しない場合，Web コンテナはどのような接続も受け付けます。

(3) <security-constraint>タグの複数定義

Servlet 2.4 以降の仕様では，web.xml の<security-constraint>タグを複数定義した場合の動作について追記されています。

アプリケーションサーバ上で<security-constraint>タグを複数定義した場合の動作を，Servlet 2.4 以降および Servlet 2.3 に分けて示します。

Servlet 2.4 以降

web.xml に<security-constraint>タグを複数定義した場合，アクセス制御の対象となる<security-constraint>タグは，Servlet の仕様書に記述された規則に従い選択されます。

Servlet 2.3

web.xml に定義された<security-constraint>タグを，ファイルの上部に記述されたものから順に確認し，リクエストの URI，および HTTP メソッドとマッチする<security-constraint>タグを使用してアクセス制御します。<http-method>タグを定義していない場合，すべての HTTP メソッドを対象とします。

(4) <url-pattern>の改行コード

Servlet 2.4 以降の仕様では，web.xml に記述する URL パターンに改行コードを含む場合の動作について追記されています。

アプリケーションサーバでは，Web アプリケーションのバージョンに関係なく，KDJE39304-W の警告メッセージが出力されます。ただし，エラーにはならないでアプリケーションは開始されます。また，該当するマッピングは無視されます。

なお，web.xml でこのような動作をするタグは，<servlet-mapping>タグ，<filter-mapping>タグ，<jsp-property-group>タグ，および<web-resource-collection>タグと各タグに含まれるサブクラスの<url-pattern>タグです。

(5) 指定したエラーページが表示されたレスポンスのステータスコード

Servlet 2.4 以降の仕様に対応した Web アプリケーションでは，web.xml によって表示するエラーページを指定した場合でも，エラーが発生した時点のレスポンスのステータスコードがクライアントに送信されます。

また，Servlet 2.2 仕様および Servlet 2.3 仕様に対応したアプリケーションでは，ステータスコード 200 がクライアントに送信されます。

(6) Web コンテナ単位でのエラーページのカスタマイズ

Servlet 2.4 以降の仕様に対応したアプリケーションでは，web.xml で指定したエラーページを表示させる場合，エラーページはカスタマイズされません。また，同じステータスコードに対してカスタマイズしたエラーページを指定した場合，Web アプリケーションで問題が発生し，web.xml で指定したエラーページの出力に失敗したときだけ，カスタマイズが有効となります。

エラーページのカスタマイズについては，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「2.18　エラーページのカスタマイズ」を参照してください。

(7) リダイレクトによるエラーページの生成

Servlet 2.4 以降の仕様に対応したアプリケーションでは，web.xml で指定したエラーページが表示されたあとのステータスコードが，エラーページの生成を委任するエラーステータスコードと一致すると，Web サーバによって生成されるエラーページが有効となります。このとき，web.xml で指定したエラーページの内容は Web サーバに転送されたあとに破棄されます。

web.xml で指定したエラーページでレスポンスのステータスコードを 200 に変更することで，web.xml で指定したエラーページを有効にできます。

(8) フィルタ機能を使用する場合の定義

Servlet 2.4 以降の仕様に対応した Web アプリケーションでは，web.xml の<filter-mapping>タグを定義するとき，サブ要素として<dispatcher>タグを記述することで，リクエストのフォワード時，インクルード時，および web.xml に記述したエラーページへの転送時にフィルタを動作させることができます。

JSP では，page ディレクティブの errorPage 属性を使用することで，JSP での例外発生時にエラーページを出力できます。ただし，このときに実行されるリクエストの転送はフォワードです。フィルタを適用する場合に<dispatcher>タグへ必要となる定義は「ERROR」ではなく，「FORWARD」となります。

5.3　cosminexus.xml の作成と編集

cosminexus.xml とは，アプリケーションサーバ独自の定義情報を記述したファイルです。ここでは，cosminexus.xml を作成および編集する方法を説明します。

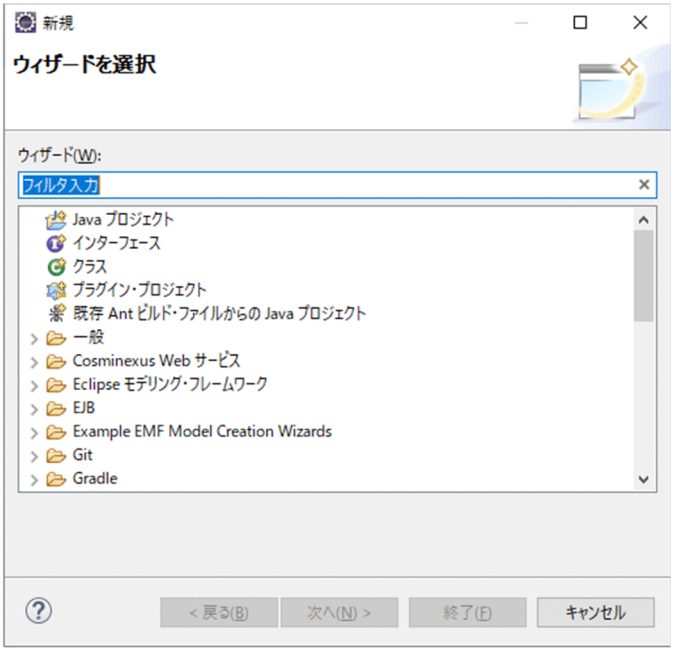
5.3.1　cosminexus.xml の作成

Eclipse の機能を使用して，cosminexus.xml を作成します。次の手順で作成してください。

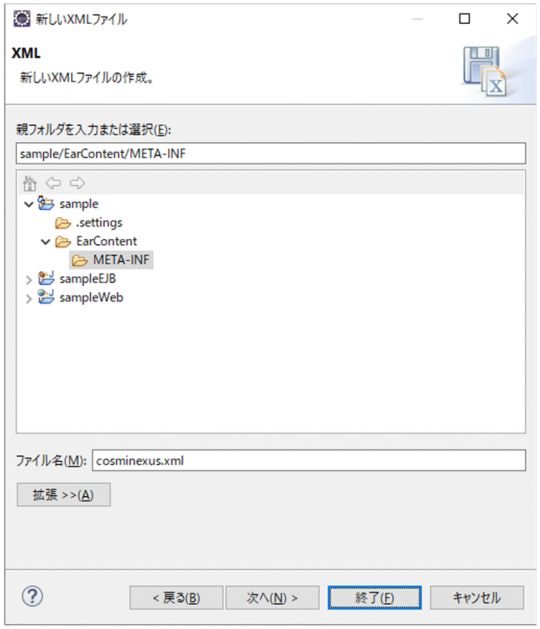
1. [J2EE] パースペクティブの［プロジェクト・エクスプローラー］ビューで，cosminexus.xml を作成するエンタープライズアプリケーションプロジェクトの META-INF フォルダを選択します。

> **参考**
>
> META-INF フォルダがない場合は，META-INF フォルダを作成してください。
>
> META-INF フォルダは，［プロジェクト・エクスプローラー］ビューで cosminexus.xml を作成するエンタープライズアプリケーションプロジェクトを選択した状態で，コンテキストメニューから［新規作成］を選択すると作成できます。

2. Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。
［新規］ダイアログが表示されます。



3. [XML] －［XML ファイル］を選択して，［次へ］ボタンをクリックします。
［新しい XML ファイル］ダイアログの［XML］ページが表示されます。



4. 次に示す項目を指定します。

項目名	指定値
親フォルダを入力または選択	cosminexus.xml の作成先となるフォルダを選択します。 デフォルトでは，手順 1.で選択したエンタープライズアプリケーションプロジェクトを選択した状態になっています。
ファイル名	「cosminexus.xml」と入力します。

5. [次へ] ボタンをクリックします。

[次から XML ファイルを作成] ページが表示されます。

6. [DTD ファイルから XML ファイルを作成] をチェックして，[次へ] ボタンをクリックします。
   [DTD ファイルの選択] ページが表示されます。

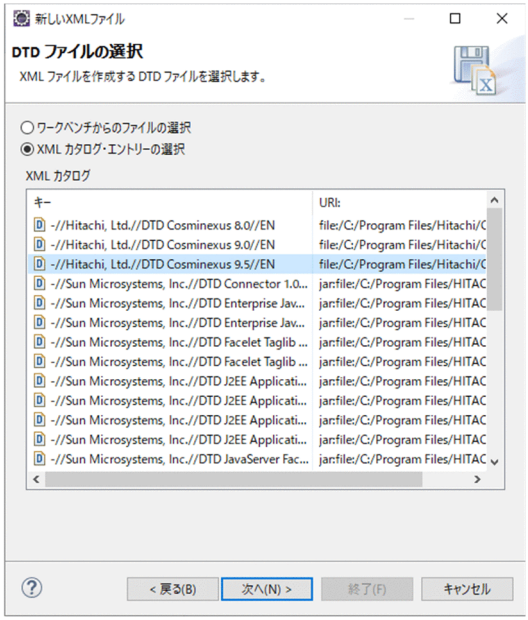
7. [XML カタログ・エントリーの選択] を選択して，[XML カタログ] エリアで cosminexus.xml の DTD を選択します。
   選択する DTD は，次の項目です。

   [キー] 列：

   -//Hitachi, Ltd.//DTD Cosminexus 9.5//EN

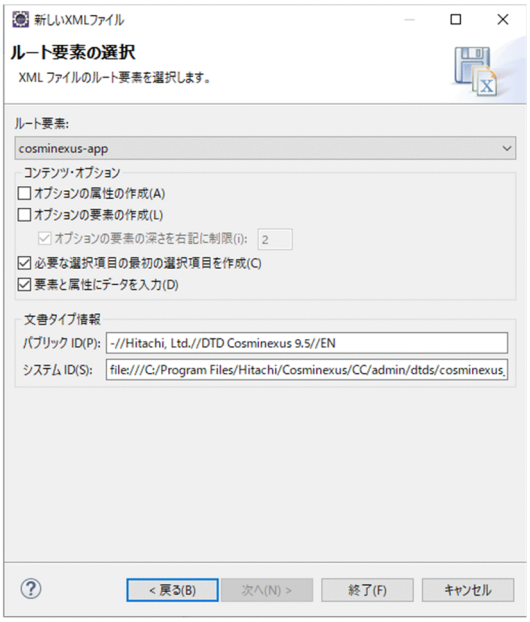
   [URI] 列：

   file:///<Developer のインストールディレクトリ>/CC/admin/dtds/cosminexus_9_5.dtd



8. [次へ] ボタンをクリックします。

[ルート要素の選択] ページが表示されます。



9. 表示された内容を変更しないで，[終了] ボタンをクリックします。

手順 1.で選択したエンタープライズアプリケーションプロジェクトに cosminexus.xml が作成されます。

作成された cosminexus.xml の定義内容は次のとおりです。属性がない状態の定義ファイルが作成されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cosminexus-app PUBLIC "-//Hitachi, Ltd.//DTD Cosminexus 9.5//EN" "file:///<Developerのインストールディレクトリ>/CC/admin/dtds/cosminexus_9_5.dtd">
<cosminexus-app>
</cosminexus-app>
```

5.3.2　cosminexus.xml エディタの操作方法

新規で作成した cosminexus.xml には属性がありません。このため，属性を追加する必要があります。

cosminexus.xml の定義内容を編集する場合，cosminexus.xml エディタを使用します。ここでは，cosminexus.xml エディタを使って，cosminexus.xml の属性を追加および編集する方法を説明します。

(1)　cosminexus.xml エディタの表示方法

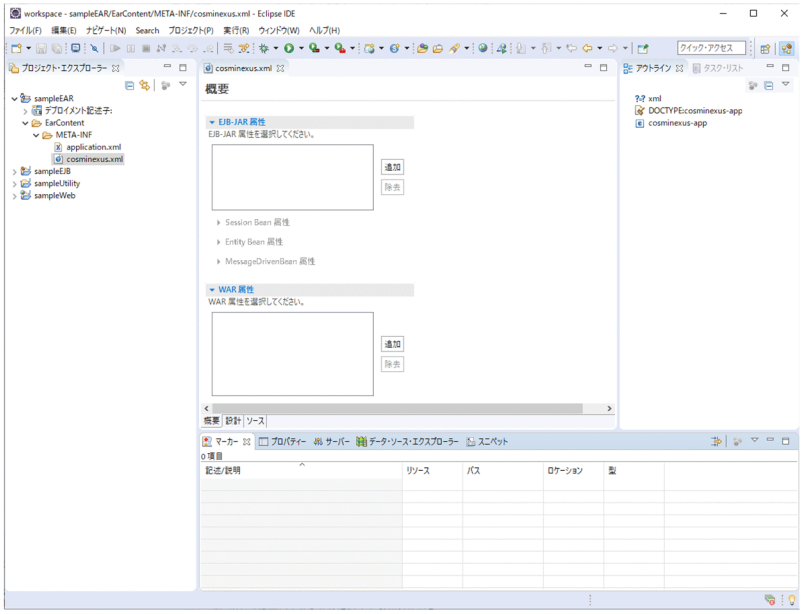
cosminexus.xml エディタを表示する手順を次に示します。

1. ［J2EE］パースペクティブの［プロジェクト・エクスプローラー］ビューで編集する cosminexus.xml を選択します。

2. コンテキストメニューから［開く］を選択します。

   選択した cosminexus.xml が cosminexus.xml エディタで表示されます。

   この状態で cosminexus.xml の編集をします。

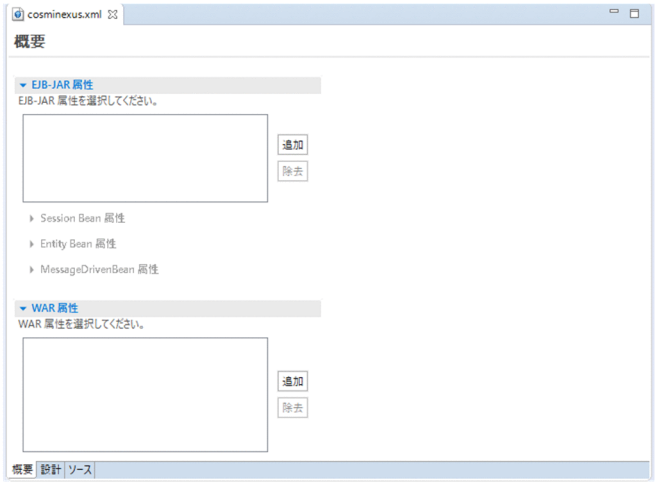


(2)　cosminexus.xml エディタの構成

cosminexus.xml エディタは，次の 3 種類の表示形式があります。このマニュアルでは，［概要］タブを使用した編集手順を説明します。

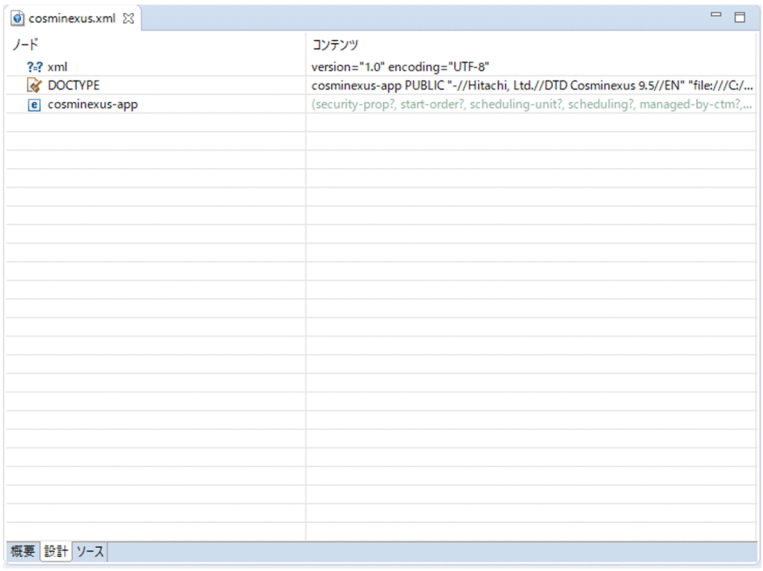
- ［概要］タブ

  GUI を使って属性を編集できる画面です。画面左側（左ペイン）に編集対象の属性の設定が表示され，画面右側（右ペイン）で属性の設定内容を編集します。



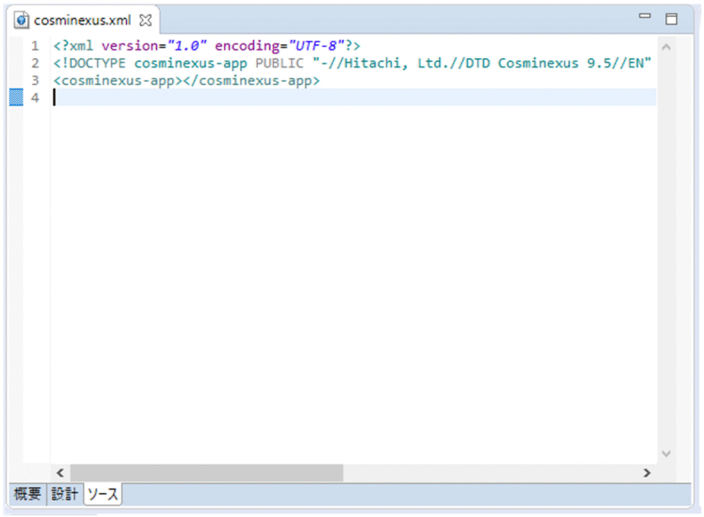
- ［設計］タブ

  cosminexus.xml の XML タグをツリー形式で表示，編集する画面です。タグ名の右側の欄で値を入力します。なお，このページではキーボードのショートカットキー（［Ctrl］＋［A］，［Ctrl］＋［C］など）は使用できません。



- ［ソース］タブ

  cosminexus.xml の内容をソースそのまま表示，編集する画面です。



(3) cosminexus.xml エディタの［概要］タブの基本操作

cosminexus.xml エディタの［概要］タブを使用して，属性を追加する手順を説明します。ここでは，Session Bean 属性を追加する手順を例にして説明します。

1. 左ペインにある［EJB-JAR 属性］の［追加］ボタンをクリックします。

   ［EJB-JAR 属性］のエリアに「EJB-JAR 属性のモジュール名を指定してください。」が追加され，右ペインに［EJB-JAR 属性の詳細］が表示されます。

2. 右ペインにある［EJB-JAR 属性の詳細］の［モジュール名］に属性を追加するモジュールのモジュール名を入力します。

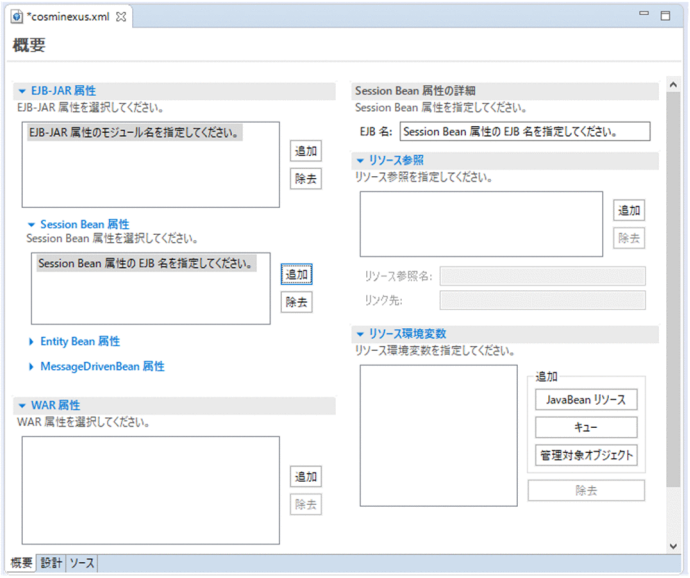
   EJB-JAR 属性のモジュール名が指定されます。

3. 左ペインにある［Session Bean 属性］を選択します。

   ［Session Bean 属性］の入力エリアが表示されます。

4. ［Session Bean 属性］の［追加］ボタンをクリックして，Session Bean 属性を追加します。

［Session Bean 属性］のエリアに項目が追加され，右ペインに［Session Bean 属性の 詳細］が表示されます。



5. 右ペインで，EJB 名，リソース参照，およびリソース環境変数を設定します。

これで，Session Bean 属性の追加は終了です。

リソース参照，およびリソース環境変数以外の設定が必要な場合は，［設計］タブ，および［ソース］タブで設定してください。

(4) 属性の編集（cosminexus.xml）

cosminexus.xml エディタを使用した各属性の編集については，次を参照してください。

- 5.3.3 EJB-JAR 属性の編集
- 5.3.4 Session Bean 属性の編集
- 5.3.5 Entity Bean 属性の編集
- 5.3.6 MessageDrivenBean 属性の編集
- 5.3.7 WAR 属性の編集

属性を編集するときの注意事項については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「16.3.6 cosminexus.xml を含むアプリケーションの運用」を参照してください。

5.3.3 EJB-JAR 属性の編集

EJB-JAR 属性を追加および編集するときに使用する cosminexus.xml エディタの［概要］タブの項目および設定内容を説明します。

(1) 左ペイン

[EJB-JAR 属性] を選択します。次の項目が表示され，属性を追加または削除できます。

[EJB-JAR 属性]

次のボタンをクリックして，属性を追加または削除します。

[追加] ボタン

EJB-JAR 属性を追加します。

なお，[追加] ボタンをクリックすると，ソースファイル上では EJB-JAR 属性の最後に次のタグが挿入されます。

```
<ejb-jar>
<module-name>EJB-JAR 属性のモジュール名を指定してください。</module-name>
</ejb-jar>
```

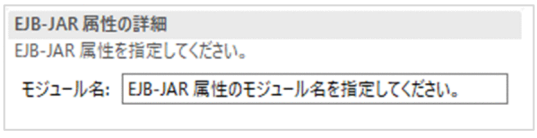
ただし，誤った操作で DTD ファイルが削除されて挿入位置が特定できない場合は，cosminexus-app 属性直下の属性群の最後に挿入されます。

[除去] ボタン

リストで選択した EJB-JAR 属性を削除します。

(2) 右ペイン

左ペインで [EJB-JAR 属性] を選択すると，右ペインに次の項目が表示され，属性の内容を設定できます。



[EJB-JAR 属性の詳細]

EJB-JAR 属性の詳細を設定します。

[モジュール名]

EJB-JAR を特定するためのキーを指定します。<ejb-jar>タグ下の<module-name>タグに対応します。

5.3.4　Session Bean 属性の編集

EJB-JAR 属性を追加したあとで，Session Bean 属性を追加および編集します。Session Bean 属性を追加および編集するときに使用する cosminexus.xml エディタの項目および設定内容を説明します。

(1)　左ペイン

［Session Bean 属性］を選択します。次の項目が表示され，属性を追加または削除できます。

［Session Bean 属性］

次のボタンをクリックして，属性を追加または削除します。

［追加］ボタン

Session Bean 属性を追加します。

なお，［追加］ボタンをクリックすると，ソースファイル上では Session Bean 属性の最後に次のタグが挿入されます。

```
<session>
<ejb-name>Session Bean 属性の EJB 名を指定してください。</ejb-name>
</session>
```

［除去］ボタン

リストで選択した Session Bean 属性を削除します。

(2)　右ペイン

左ペインで［Session Bean 属性］を選択すると，右ペインに次の項目が表示され，属性の内容を設定できます。

[Session Bean 属性の詳細]

Session Bean 属性の詳細を設定します。

[EJB 名]

Session Bean を特定するためのキーを指定します。<ejb-jar>タグ下にある<session>タグ下の<ejb-name>タグの値に対応します。

[リソース参照]

リソース参照を設定します。

[追加] ボタン

リソース参照を追加します。

なお，[追加] ボタンをクリックすると，ソースファイル上ではリソース参照のタグの最後に次のタグが挿入されます。

```
<resource-ref>
<res-ref-name>リソース参照名を指定してください。</res-ref-name>
<linked-to>リンク先を指定してください。</linked-to>
</resource-ref>
```

追加したリソース参照を選択すると，次の項目を設定できます。

- **[リソース参照名]**

  リストで選択した変数のリソース参照名を設定します。<resource-ref>タグ下の<res-ref-name>タグの値に対応します。

- **[リンク先]**

  リストで選択した変数のリソース参照のリンク先を設定します。<resource-ref>タグ下の<linked-to>タグの値に対応します。

[除去] ボタン

リストで選択したリソース参照を削除します。

[リソース環境変数]

リソース環境変数を設定します。

[JavaBean リソース] ボタン

JavaBean のリソース環境変数を追加します。

なお，[JavaBean リソース] ボタンをクリックすると，ソースファイル上ではリソース環境変数のタグの最後に次のタグが挿入されます。

```
<resource-env-ref>
<resource-env-ref-name>リソース環境変数名を指定してください。</resource-env-ref-name>
<linked-to>JavaBeanリソース名を指定してください。</linked-to>
</resource-env-ref >
```

追加した JavaBean のリソース環境変数を選択すると，次の項目を設定できます。

- [リソース環境変数名]

  リストで選択した変数のリソース環境変数名を指定します。<resource-env-ref>タグ下の<resource-env-ref-name>タグの値に対応します。

- [JavaBean リソース名]

  リストで選択した変数のリンク先の JavaBean リソース名を指定します。<resource-env-ref>タグ下の<linked-to>タグの値に対応します。

[キュー] ボタン

キューのリソース環境変数を追加します。

なお，[キュー] ボタンをクリックすると，ソースファイル上ではリソース環境変数のタグの最後に次のタグが挿入されます。

```
<resource-env-ref>
<resource-env-ref-name>リソース環境変数名を指定してください。</resource-env-ref-name>
<linked-queue>
<resource-adapter>リソース・アダプター名を指定してください。</resource-adapter>
<queue>キュー名を指定してください。</queue>
</linked-queue>
</resource-env-ref >
```

追加したキューのリソース環境変数を選択すると，次の項目を設定できます。

- [リソース環境変数名]

  リストで選択した変数のリソース環境変数名を設定します。<resource-env-ref>タグ下の<resource-env-ref-name>タグの値に対応します。

- [リソース・アダプター名]

  リストで選択した変数のリソースアダプタ名を設定します。<resource-env-ref>タグ下にある<linked-queue>タグ下の<resource-adapter>タグの値に対応します。

- [キュー名]

リストで選択した変数のキュー名を設定します。<resource-env-ref>タグ下にある<linked-queue>タグ下の<queue>タグの値に対応します。

[管理対象オブジェクト] ボタン

管理対象オブジェクトのリソース環境変数を追加します。

なお，[管理対象オブジェクト] ボタンをクリックすると，ソースファイル上ではリソース環境変数のタグの最後に次のタグが挿入されます。

```
<resource-env-ref>
<resource-env-ref-name>リソース環境変数名を指定してください。</resource-env-ref-name>
<linked-adminobject>
<resourceadapter-name>リソース・アダプター名を指定してください。</resourceadapter-name
>
<adminobject-name>管理対象オブジェクト名を指定してください。</adminobject-name>
</linked-adminobject>
</resource-env-ref >
```

追加した管理対象オブジェクトのリソース環境変数を選択すると，次の項目を設定できます。

- **[リソース環境変数名]**

  リストで選択した変数のリソース環境変数名を設定します。<resource-env-ref>タグ下の<resource-env-ref-name>タグの値に対応します。

- **[リソース・アダプター名]**

  リストで選択した変数のリソースアダプタ名を設定します。<resource-env-ref>タグ下にある<linked-adminobject>タグ下の<resourceadapter-name>タグの値に対応します。

- **[管理対象オブジェクト名]**

  リストで選択した変数の管理対象オブジェクト名を設定します。<resource-env-ref>タグ下にある<linked-adminobject>タグ下の<adminobject-name>タグの値に対応します。

[除去] ボタン

リストで選択したリソース環境変数を削除します。

5.3.5 Entity Bean 属性の編集

EJB-JAR 属性を追加したあとで，Entity Baen 属性を追加および編集します。Entity Bean 属性を追加および編集するときに使用する cosminexus.xml エディタの項目および設定内容を説明します。

(1) 左ペイン

[Entity Bean 属性] を選択します。次の項目が表示され，属性を追加または削除できます。

[Entity Bean 属性]

次のボタンをクリックして，属性を追加または削除します。

[追加] ボタン

Entity Bean 属性を追加します。

なお，[追加] ボタンをクリックすると，ソースファイル上では Entity Bean 属性の最後に次のタグが挿入されます。

```
<entity>
<ejb-name>Entity Bean 属性の EJB 名を指定してください。</ejb-name>
</entity>
```

[除去] ボタン

リストで選択した Entity Bean 属性を削除します。

(2) 右ペイン

左ペインで［Entity Bean 属性］を選択すると，次の項目が表示され，属性の内容を設定できます。

[Entity Bean 属性の詳細]

Entity Bean 属性の詳細を設定します。

[EJB 名]

Entity Bean を特定するためのキーを指定します。<ejb-jar>タグ下にある<entity>タグ下の<ejb-name>タグの値に対応します。

[リソース参照]

リソース参照を編集します。設定できる内容については，「5.3.4(2)　右ペイン」にあるリソース参照の説明を参照してください。

[リソース環境変数]

リソース環境変数を編集します。設定できる内容については，「5.3.4(2)　右ペイン」にあるリソース環境変数の説明を参照してください。

5.3.6　MessageDrivenBean 属性の編集

EJB-JAR 属性を追加したあとで，MessageDrivenBean 属性を追加および編集します。
MessageDrivenBean 属性を追加および編集するときに使用する cosminexus.xml エディタの項目および設定内容を説明します。

(1)　左ペイン

[MessageDrivenBean 属性] を選択します。次の項目が表示され，属性を追加または削除できます。

[MessageDrivenBean 属性]

次のボタンをクリックして，属性を追加または削除します。

[追加] ボタン

MessageDrivenBean 属性を追加します。

なお，[追加] ボタンをクリックすると，ソースファイル上では MessageDrivenBean 属性の最後に次のタグが挿入されます。

```
<message>
<ejb-name>MessageDrivenBean 属性の EJB 名を指定してください。</ejb-name>
</message>
```

[除去] ボタン

リストで選択した MessageDrivenBean 属性を削除します。

(2) 右ペイン

左ペインで［MessageDrivenBean 属性］を選択すると，右ペインに次の項目が表示され，属性の内容を設定できます。

[MessageDrivenBean 属性の詳細]

MessageDrivenBean 属性の詳細を設定します。

[EJB 名]

MessageDrivenBean を特定するためのキーを指定します。<ejb-jar>タグ下にある<message>タグ下の<ejb-name>タグの値に対応します。

[リソース参照]

リソース参照を編集します。設定できる内容については，「5.3.4(2)　右ペイン」にあるリソース参照の説明を参照してください。

[リソース環境変数]

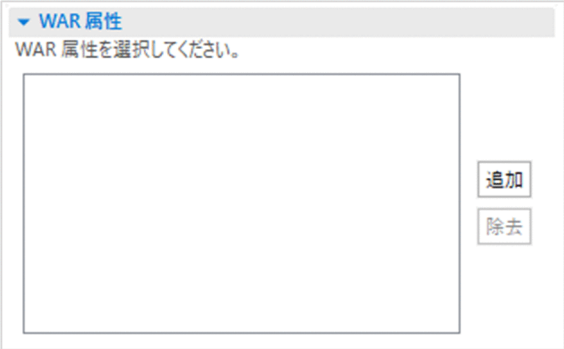
リソース環境変数を編集します。設定できる内容については，「5.3.4(2)　右ペイン」にあるリソース環境変数の説明を参照してください。

5.3.7　WAR 属性の編集

WAR 属性を追加および編集するときに使用する cosminexus.xml エディタの項目および設定内容を説明します。

(1)　左ペイン

［WAR 属性］を選択します。次の項目が表示され，属性を追加または削除できます。



[WAR 属性]

次のボタンをクリックして，属性を追加または削除します。

[追加] ボタン

WAR 属性を追加します。

なお，[追加] ボタンをクリックすると，ソースファイル上では WAR 属性の最後に次のタグが挿入されます。

```
<war>
<module-name>WAR 属性のモジュール名を指定してください。</module-name>
</war>
```

ただし，誤った操作で DTD ファイルが削除されて挿入位置が特定できない場合は，cosminexus-app 属性直下の属性群の最後に挿入されます。

[除去] ボタン

リストで選択した WAR 属性を削除します。

(2) 右ペイン

左ペインで［WAR 属性］を選択すると，右ペインに次の項目が表示され，属性の内容を設定できます。

[WAR 属性の詳細]

WAR 属性の詳細を設定します。

[モジュール名]

Entity Bean を特定するためのキーを指定します。<war>タグ下の<module-name>タグの値に対応します。

[リソース参照]

リソース参照を編集します。設定できる内容については，「5.3.4(2)　右ペイン」にあるリソース参照の説明を参照してください。

[リソース環境変数]

リソース環境変数を編集します。設定できる内容については，「5.3.4(2)　右ペイン」にあるリソース環境変数の説明を参照してください。

6 J2EE アプリケーションのテスト

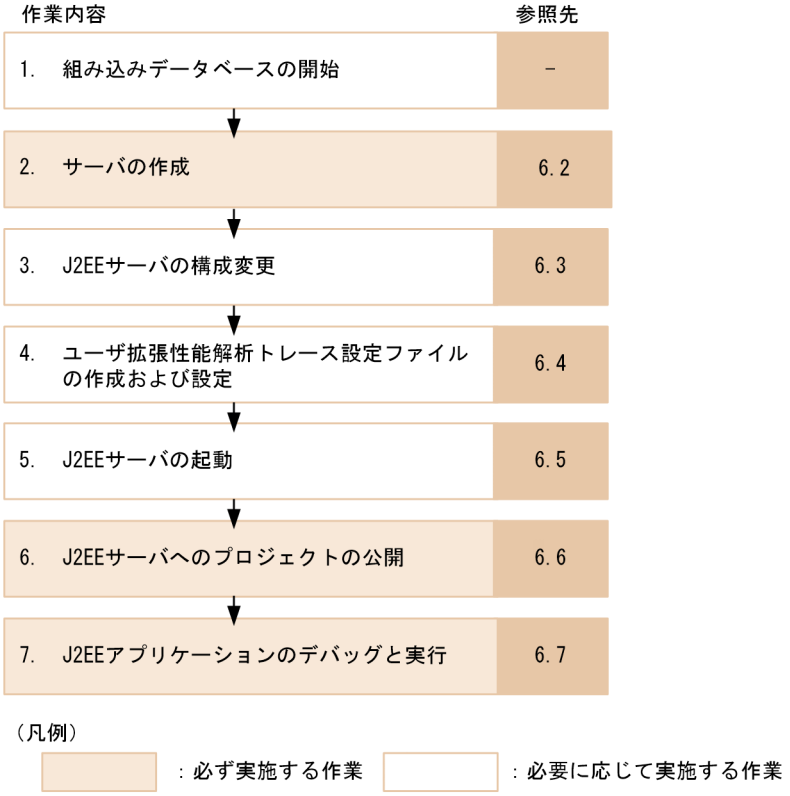
作成した J2EE アプリケーションを実行環境に配布する前に，デバッグ環境で実行して，テストします。ここでは，J2EE アプリケーションのテストに必要な準備，およびテストの手順を説明します。

6.1　J2EE アプリケーションのテストの流れ

作成した Eclipse のプロジェクトを J2EE サーバにデプロイして，J2EE アプリケーションをデバッグ環境で実行します。

J2EE アプリケーションのテストの流れを次の図に示します。

図 6-1　J2EE アプリケーションのテストの流れ図



それぞれの手順の概要を説明します。

1. 組み込みデータベースの開始

組み込みデータベースの開始については，HiRDB のマニュアルを参照してください。

2. サーバの作成

Eclipse で J2EE サーバを操作するためには，J2EE サーバ名やランタイム環境などの構成情報を作成する必要があります。この構成情報をサーバといいます。このサーバを作成することで，Eclipse で J2EE サーバを操作できるようになります。詳細は，「6.2　サーバの作成」を参照してください。

3. サーバの構成変更

サーバの構成は，サーバエディタを使用して変更できます。詳細は，「6.3　サーバの構成変更」を参照してください。

4. ユーザ拡張性能解析トレース設定ファイルの作成および設定

J2EE アプリケーションに対して，性能解析トレースを利用した性能解析を実施したい場合は，ユーザ拡張性能解析トレース設定ファイルを作成します。詳細は，「6.4　ユーザ拡張性能解析トレース設定ファイルの作成および設定」を参照してください。

5. J2EE サーバの起動

J2EE サーバをデバッグモードで起動します。詳細は，「6.5　J2EE サーバの起動と停止」を参照してください。

6. J2EE サーバへのプロジェクトの公開

Eclipse のプロジェクトで開発した J2EE アプリケーションを J2EE サーバへ公開（デプロイ）して，デバッグおよびテストができる状態にします。詳細は，「6.6　J2EE サーバへのプロジェクトの公開」を参照してください。

7. J2EE アプリケーションのデバッグと実行

J2EE アプリケーションをデバッグしたり，実行したりします。詳細は，「6.7　J2EE アプリケーションのデバッグと実行」を参照してください。

以降の節では，この流れに沿って J2EE アプリケーションのテストの手順を説明します。

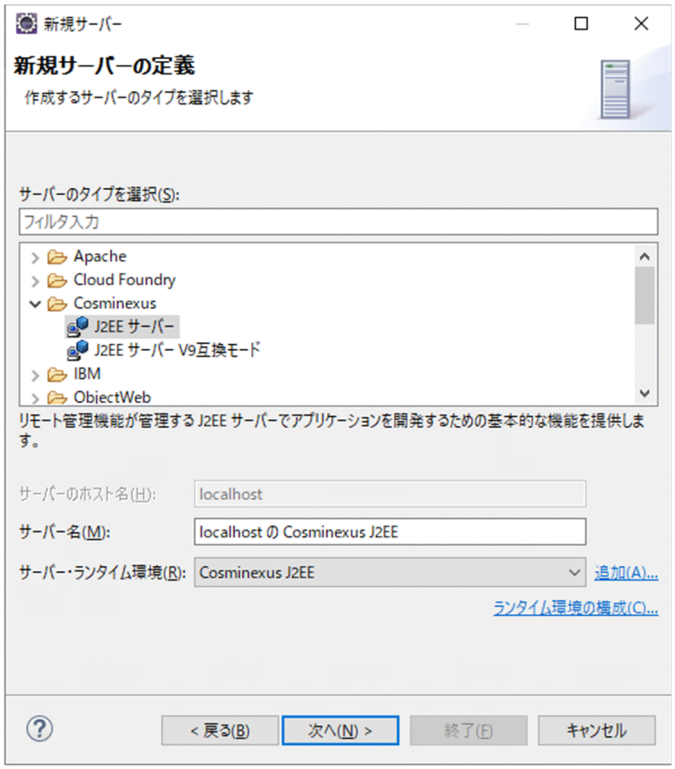
6.2　サーバの作成

Eclipse で J2EE サーバを操作するためには，J2EE サーバ名やランタイム環境などの構成情報を作成する必要があります。この構成情報をサーバといいます。このサーバを作成することで，Eclipse で J2EE サーバを操作できるようになります。

サーバを作成する手順を次に示します。

1. Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。
   ［新規］ダイアログが表示されます。

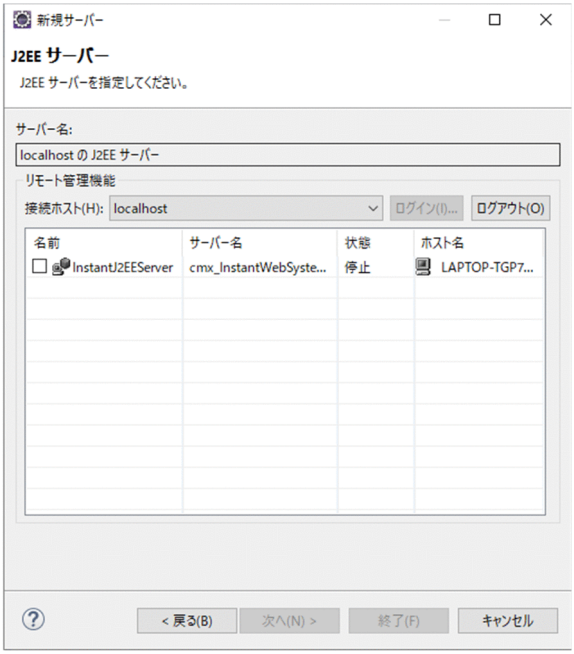
2. ［サーバー］－［サーバー］を選択して，［次へ］ボタンをクリックします。
   ［新規サーバー］ダイアログの［新規サーバーの定義］ページが表示されます。



3. 次の項目を指定します。

項目名	指定値
サーバーのタイプを選択	作成するサーバの種別を選択します。WTP コネクタを使用する場合は，［Cosminexus］－［J2EE サーバー］を選択します。
サーバー名	サーバ名は，［J2EE サーバー］ページの値が使用されるため，ここでは設定不要です。
サーバー・ランタイム環境	Cosminexus J2EE サーバランタイムを指定してください。

4. ［次へ］ボタンをクリックします。
   ［J2EE サーバー］ページが表示されます。



5. リモート管理機能の接続ホストに，接続先のホストを選択し，ログインしてない場合は［ログイン］ボタンをクリックします。

   接続先の J2EE サーバが表示されます。

   開発環境インスタントセットアップ機能で J2EE サーバを構築した場合は，次の名称が表示されます。

   ```
   cmx_InstantWebSystem_unit1_J2EE_01
   ```

6. J2EE サーバを選択し，［終了］ボタンをクリックします。

   ［サーバー］ビューに作成した Cosminexus J2EE サーバが表示されます。

●サーバ作成時の注意事項

サーバを作成する場合は，次の点に注意してください。

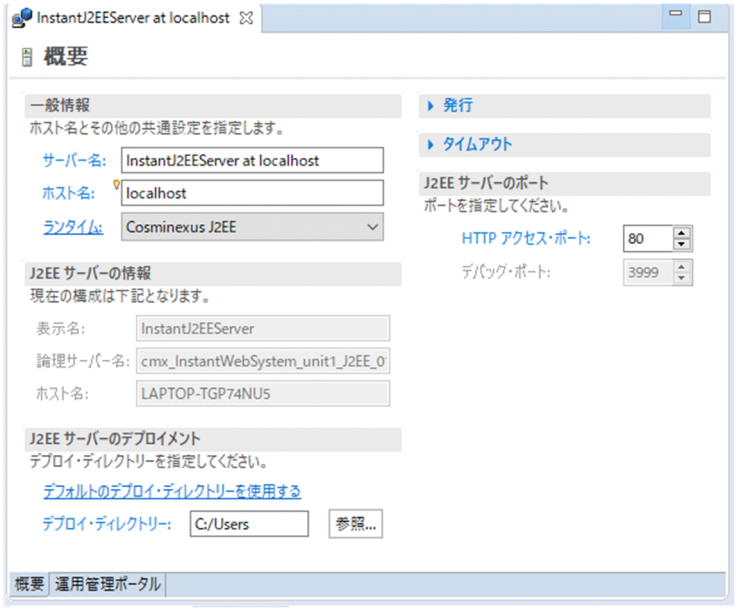
- ［サーバー］ビューからサーバを削除する場合は，サーバを選択してから削除してください。サーバを選択していない状態で削除すると，エラーが発生するおそれがあります。
- ［サーバー］ビューに表示されるサーバは，［新規サーバーの定義］ページで表示されるサーバ名ではありません。［J2EE サーバー］ページのサーバ名が表示されます。
- ［サーバー］ビューでコピーして作成したサーバは操作対象にはなりません。

6.3　サーバの構成変更

サーバの構成は，サーバエディタを使用して変更できます。Cosminexus J2EE サーバの構成を変更する手順を次に示します。

1. [サーバー] ビューで構成を変更する Cosminexus J2EE サーバをダブルクリックします。

   サーバの概要ページが表示されます。

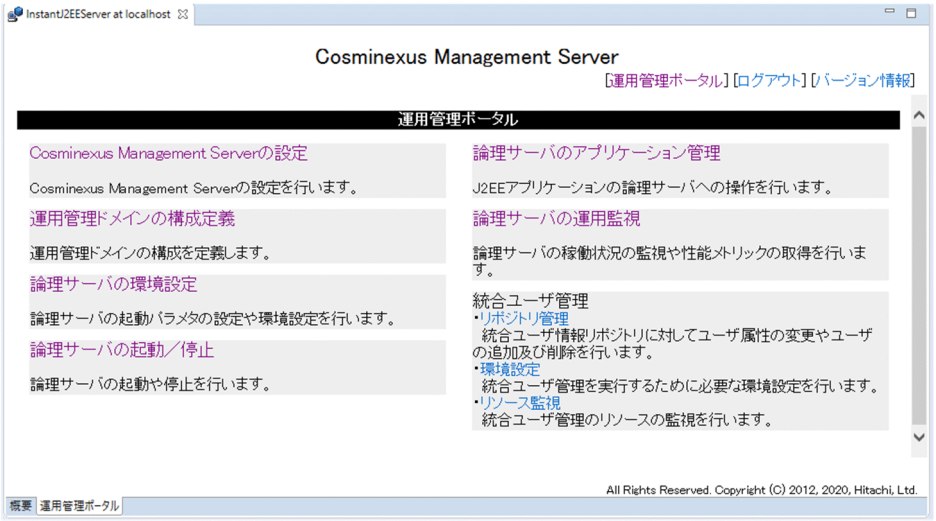


2. 次の項目を任意の値に変更します。

項目名	説明
一般情報	[サーバー] ビューに表示するサーバ名を変更できます。なお，ランタイムは変更しないでください。
J2EE サーバーの情報	J2EE サーバの表示名，論理サーバ名，および J2EE サーバのあるホスト名を確認できます。
J2EE サーバーのデプロイメント	J2EE アプリケーションをデプロイするディレクトリを変更できます。ディレクトリのパスは，絶対パスで存在するディレクトリを指定してください。
J2EE サーバーのポート	[サーバーでデバッグ] ダイアログからの J2EE アプリケーションのデバッグ，および [サーバーで実行] ダイアログからの J2EE アプリケーションの実行時に使用する HTTP アクセスポートとデバッグポートを変更できます。デバッグポートは，リモートの J2EE サーバを使用する場合だけ変更できます。 HTTP アクセスポートのデフォルト値は 80 です。1～65535 の整数で指定できます。 デバッグポートのデフォルト値は 3999 です。1～65535 の整数で指定できます。

3. 運用管理ポータルを使用する場合は，[運用管理ポータル] タブをクリックします。

   運用管理ポータルの画面が表示されます。



●J2EE サーバの構成変更時の注意事項

J2EE サーバの構成を変更する場合は，次の点に注意してください。

- 公開する J2EE アプリケーションは，デプロイディレクトリに作成されます。デプロイディレクトリおよび J2EE アプリケーションの階層が深い場合は，必要に応じて，短いパスをデプロイディレクトリに指定してください。
- プロジェクトを公開する場合に，複数のサーバで同じデプロイディレクトリを指定しないでください。エラーが発生するおそれがあります。
- サーバの概要ページの表示で J2EE サーバの情報の取得に失敗した場合，リモート管理機能の設定などを見直してください。見直し後，再度サーバエディタを開いてください。
- J2EE サーバの操作に対するタイムアウトは，［設定］ダイアログの［サーバー］－［リモート管理］－［接続］ページで設定します。正しくタイムアウトを動作させるために，サーバの概要ページの［タイムアウト］セクションは，［接続］ページの設定より大きな値を指定してください。
- サーバを削除しても，作成したデプロイディレクトリは削除されません。必要に応じて，手動で削除してください。
- デプロイディレクトリには，円マーク（¥）だけやスラッシュ（/）だけのパスは指定しないでください。
- Eclipse を使用してローカルの J2EE サーバで動作する J2EE アプリケーションをデバッグする場合，デバッグポートとして OS によって割り当てられるポート番号を使用します。ほかのプログラムで使用するポート番号との衝突を避けるため，必要に応じて，J2EE サーバを起動する前に，ポートを使用しているほかのプログラムを起動してください。

6.4　ユーザ拡張性能解析トレース設定ファイルの作成および設定

ユーザ拡張性能解析トレースを利用するために必要な，ユーザ拡張性能解析トレース設定ファイルの作成，および設定方法について説明します。

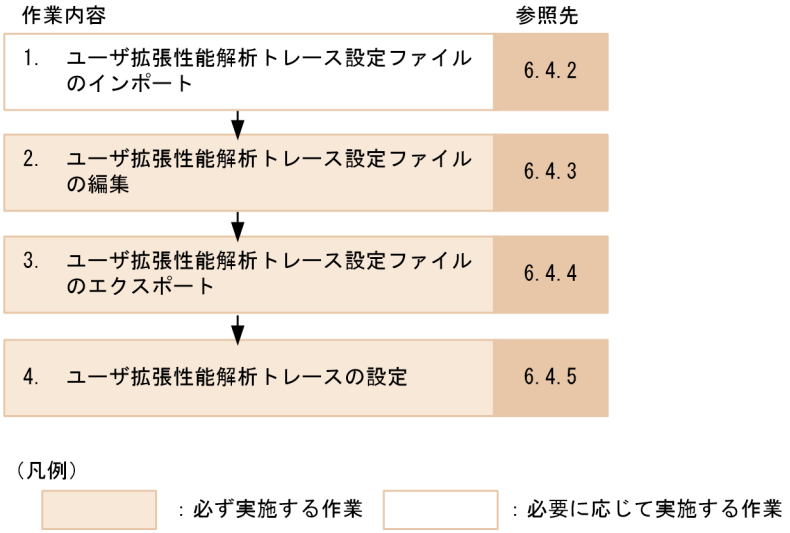
ポイント

ユーザ拡張性能解析トレースの概要については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「7.2.2　アプリケーションの性能解析トレースの概要」を参照してください。

6.4.1　ユーザ拡張性能解析トレース設定ファイルの作成および設定の流れ

ユーザ拡張性能解析トレース設定ファイルの作成および設定の流れを次の図に示します。

図 6-2　ユーザ拡張性能解析トレース設定ファイルの作成および設定の流れ



それぞれの手順の概要を説明します。

1. ユーザ拡張性能解析トレース設定ファイルのインポート

作成済みのユーザ拡張性能解析トレース設定ファイルがある場合，ユーザ拡張性能解析トレース設定ファイルをインポートできます。詳細は，「6.4.2　ユーザ拡張性能解析トレース設定ファイルのインポート」を参照してください。

2. ユーザ拡張性能解析トレース設定ファイルの編集

ユーザ拡張性能解析トレース設定ファイルにトレース取得ポイントを設定したり，削除したりする手順について説明します。操作はビュー上で実施します。詳細は，「6.4.3　ユーザ拡張性能解析トレース設定ファイルの編集」を参照してください。

3. ユーザ拡張性能解析トレース設定ファイルのエクスポート

作成または編集したユーザ拡張性能解析トレース設定ファイルをエクスポートし，実行環境に配置します。詳細は，「6.4.4　ユーザ拡張性能解析トレース設定ファイルのエクスポート」

4. ユーザ拡張性能解析トレースの設定

ユーザ拡張性能解析トレースを利用するために必要なパラメタを設定します。詳細については，「6.4.5 ユーザ拡張性能解析トレースの設定」を参照してください。

以降の項では，この流れに沿ってユーザ拡張性能解析トレース設定ファイルを作成および設定する手順を説明します。

6.4.2　ユーザ拡張性能解析トレース設定ファイルのインポート

作成済みのユーザ拡張性能解析トレース設定ファイルがある場合，ファイルをインポートできます。なお，ユーザ拡張性能解析トレース設定ファイルを新規で作成する場合は，この手順は不要です。

インポートの手順を次に示します。

注意事項

- アプリケーションサーバのバージョンが 09-00 で作成したユーザ拡張性能解析トレース設定ファイルをインポートできます。

  ただし，09-00 と 09-50 以降では，ユーザ拡張性能解析トレース設定ファイルで指定できるトレースポイントの上限値やファイルの書式に違いがあります。

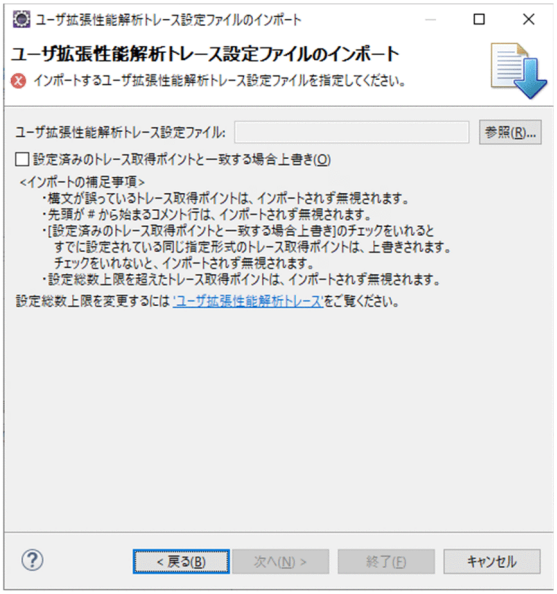
  09-00 で作成したユーザ拡張性能解析トレース設定ファイルをインポートしたあとにエクスポートした場合，出力結果は 09-50 以降の形式となります。
- コンストラクタをトレース対象のメソッドとして記述する場合，<init>で指定してください。クラス名と同じ名前のメソッドの形式で記述した場合，インポートしたときにコンストラクタにはトレース取得ポイントは設定されません。

1. Eclipse のメニューから，［ファイル］－［インポート］を選択します。

［インポート］ダイアログが表示されます。

2. ［ユーザ拡張性能解析トレース］－［ユーザ拡張性能解析トレース設定ファイル］を選択して，［次へ］ボタンをクリックします。

［ユーザ拡張性能解析トレース設定ファイルのインポート］ダイアログが表示されます。



参考

[ユーザ拡張性能解析トレース設定ファイルのインポート] ダイアログは，次に示す方法でも表示できます。

- [トレース取得ポイント] ビューのビューツールバーで [ユーザ拡張性能解析トレース設定ファイルのインポート] ボタンをクリックする。
- [トレース取得ポイント] ビューのプルダウンメニューで [ユーザ拡張性能解析トレース設定ファイルのインポート] を選択する。

3. **[ユーザ拡張性能解析トレース設定ファイル] の [参照] ボタンをクリックします。**

   [ファイルを開く] ダイアログが表示されます。インポート対象とするユーザ拡張性能解析トレース設定ファイルのパスを指定します。

   [設定済みのトレース取得ポイントと一致する場合上書き] チェックボックスをチェックすると，すでに設定されているトレース取得ポイントを上書きします。

4. **[終了] ボタンをクリックします。**

   指定したユーザ拡張性能解析トレース設定ファイルがインポートされます。

6.4.3 ユーザ拡張性能解析トレース設定ファイルの編集

ここでは，ユーザ拡張性能解析トレース設定ファイルの編集について説明します。

Developer では，トレース取得ポイントを Eclipse 上で設定できます。Eclipse のビュー上でできる編集作業を次に示します。

- トレース取得ポイントの切り替え

設定済みのトレース取得ポイントを削除したり，新規にトレース取得ポイントとして設定したりします。詳細については「6.4.3(1)　トレース取得ポイントの切り替え」を参照してください。

- トレース取得ポイントの設定

  J2EE パースペクティブに組み込まれたプロジェクト・エクスプローラービュー，およびアウトラインビューなどのビュー上の Java 要素に対して，トレース取得ポイントを設定します。詳細については「6.4.3(2)　トレース取得ポイントの設定」を参照してください。

- トレース取得ポイントの一覧表示，および削除

  設定済みのトレース取得ポイントを一覧表示します。また，一覧表示したビュー上で，トレース取得ポイントの編集，および削除を実施できます。詳細については「6.4.3(3)　トレース取得ポイントの一覧表示，および編集・削除」を参照してください。

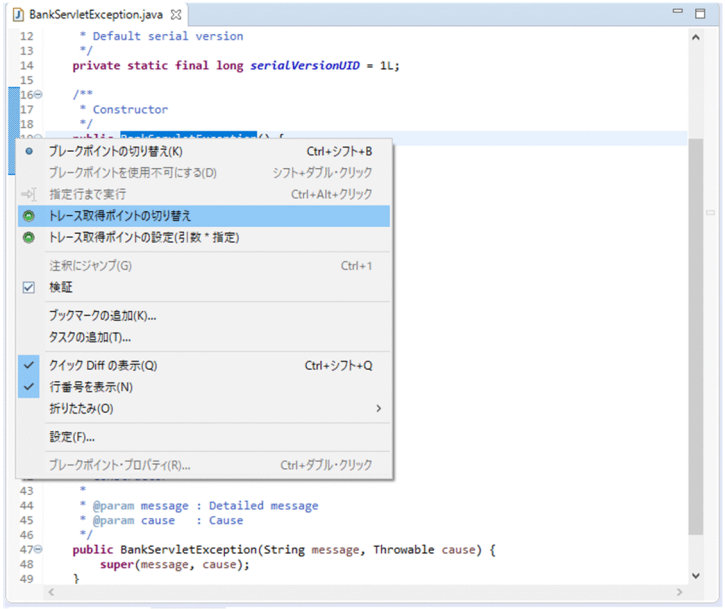
それぞれの手順を次に示します。

(1)　トレース取得ポイントの切り替え

設定済みのトレース取得ポイントを削除したり，新規にトレース取得ポイントとして設定したりします。

手順を次に示します。

1. [プロジェクト・エクスプローラー] ビューで編集するプロジェクトの Java ソースファイルを開きます。
2. メソッドを選択し，コンテキストメニューから［トレース取得ポイントの切り替え］を選択します。



トレース取得ポイントが次のように切り替わります。

状態	説明
トレース取得ポイントが設定済みのメソッドの場合	• エディターの垂直ルーラーに付与された［トレース取得ポイント ON］（ ）マーカーが削除されます。 • ［トレース取得ポイント］ビューからトレース取得ポイントが削除されます。

状態	説明
トレース取得ポイントが設定されていないメソッドの場合	• エディターの垂直ルーラーに［トレース取得ポイント ON］（ ）マーカーが付与されます。 • ［トレース取得ポイント］ビューにトレース取得ポイントが追加されます。

参考

設定済みのトレース取得ポイントや，新規に設定したトレース取得ポイントには，マーカーが表示されます。マーカーが表示された例を次に示します。

注意事項

- エディター内で選択できない（垂直ルーラーが反転していない）場合は，コンテキストメニューは表示されません。操作によって垂直ルーラーの反転が間に合わない場合は，反転を待ってからトレース取得ポイントの切り替え操作を実施してください。
- メソッドの位置で操作しないとコンテキストメニューは表示されません。
- トレース取得ポイントには上限値を設けています。上限値として設定した値以上のトレース取得ポイントを設定しようとした場合，ダイアログが表示されてトレース取得ポイントを設定できません。トレース取得ポイントの上限値の設定手順を次に示します。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
2. 左ペインで［ユーザ拡張性能解析トレース］を選択します。
   ［ユーザ拡張性能解析トレース］ページが表示されます。
3. ［トレース取得ポイント設定総数上限］にトレース取得ポイントの上限値を指定します。
4. ［適用］ボタンまたは［OK］ボタンをクリックします。

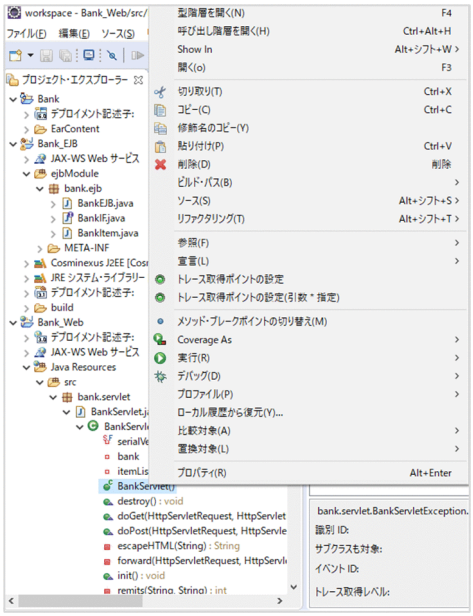
(2)　トレース取得ポイントの設定

J2EE パースペクティブに組み込まれたプロジェクト・エクスプローラービュー，およびアウトラインビューなどのビュー上の Java 要素に対して，トレース取得ポイントを設定します。

手順を次に示します。

1. **[プロジェクト・エクスプローラー] ビューでパッケージ，クラス，またはメソッドを選択します。**
   パッケージ，クラス，またはメソッドを選択します。
   JAR ファイルに含まれるパッケージ，クラス，またはメソッドを選択することもできます。

2. **コンテキストメニューから［トレース取得ポイントの設定］または［トレース取得ポイントの設定(引数 * 指定)］を選択します。**



   トレース取得ポイントが設定されます。

   ［トレース取得ポイントの設定］および［トレース取得ポイントの設定(引数 * 指定)］について説明します。

コンテキストメニューの表示名	説明
トレース取得ポイントの設定	選択したメソッドに対して，トレース取得ポイントを設定できます。［トレース取得ポイントの設定］を選択した場合のエディターおよびビューの表示について説明します。 • エディターの垂直ルーラーに［トレース取得ポイント ON］（ ）マーカーが付与されます。 • ［トレース取得ポイント］ビューにトレース取得ポイントが追加されます。
トレース取得ポイントの設定(引数 * 指定)	選択したメソッドと同名のメソッドすべてに対して，トレース取得ポイントを設定できます。 引数が異なるものも対象となります。［トレース取得ポイントの設定(引数 * 指定)］を選択した場合のエディターおよびビューの表示について説明します。 • エディターには何も表示されません。

コンテキストメニューの表示名	説明
	• ［トレース取得ポイント］ビューにトレース取得ポイントが追加されます。

トレース取得ポイントのマーカーは，ソースファイルが存在し，かつトレース取得ポイントの対象がコンストラクタ，またはメソッドの場合に付与されます。なお，JAR ファイル内の Java 要素にトレース取得ポイントを設定した場合，マーカーは付与されません。

注意事項

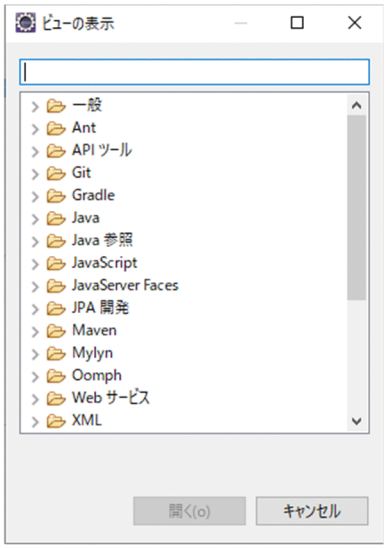
- JAR ファイルに static イニシャライザが含まれる場合，名称を持たないメソッドとして表示されますが，トレース取得ポイントを設定しないでください。トレース取得ポイントを設定した場合の動作はサポートしません。
- メソッドの引数が可変長やジェネリクスの場合には，変換してトレース取得ポイントを設定します。引数が可変長であるメソッドの場合，「...」を「[ ]」に変換します。また，ジェネリクスを使用したクラス名が含まれる場合，ジェネリクスの記述を削除します。

(3) トレース取得ポイントの一覧表示，および編集・削除

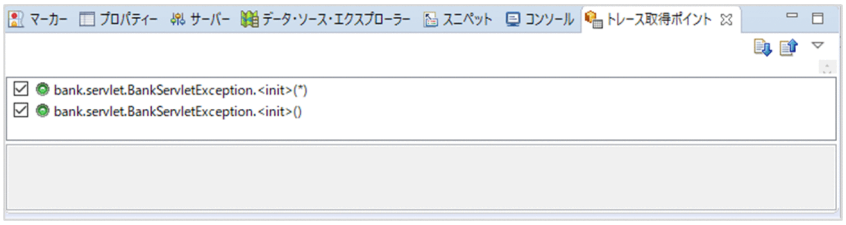
設定済みのトレース取得ポイントを一覧表示します。また，一覧表示したビュー上で，トレース取得ポイントの編集，および削除を実施できます。

手順を次に示します。

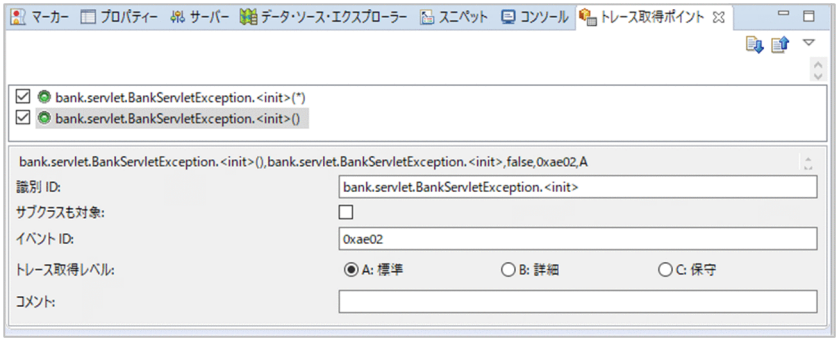
1. Eclipse のメニューから，［ウィンドウ］－［ビューの表示］－［その他］を選択します。
   ［ビューの表示］ダイアログが表示されます。



2. ［ビューの表示］ダイアログで，［ユーザ拡張性能解析トレース］－［トレース取得ポイント］を選択し，［OK］ボタンをクリックします。
   設定済みトレース取得ポイントが，［トレース取得ポイント］ビューに表示されます。



3. [トレース取得ポイント] ビューに表示されたトレースポイントのうち，編集したいトレース取得ポイントを選択します。



トレース取得ポイントは，Ctrl キーまたは Shift キーを押しながらクリックすると，複数選択できます。

選択したトレース取得ポイントの詳細が［ユーザ拡張性能解析トレース詳細設定］ビューに表示されます。必要に応じて設定を変更します。

［ユーザ拡張性能解析トレース詳細設定］ビューに表示される項目を次に示します。

項目名	説明
記述形式※	ユーザ拡張性能解析トレース設定ファイルの記述形式が表示されます。
識別 ID	任意の文字列を指定します。
サブクラスも対象	サブクラスも対象にするかどうかを指定します。なお，サブクラスのチェックボックスをチェックしても，継承先の同名のメソッドに対して，トレース取得ポイントのマーカーは付与されません。 チェックボックスをチェックする 　サブクラスフラグを true に設定します。 チェックボックスをチェックしない 　サブクラスフラグを false に設定します。
イベント ID	4 桁の 16 進数を指定します。 0xae02～0xae7e，または 0xc000～0xcffe の範囲で指定します。
トレース取得レベル	記述形式に指定したトレース取得レベルを，アルファベットで選択します。 A：標準 B：詳細 C：保守
コメント	トレース取得ポイントにコメントを指定します。

注※

記述形式の引数が「*（アスタリスク）」の場合，エディターのルーラー上にトレース取得ポイントのマーカーは表示されません。

4. 一覧に表示されたトレース取得ポイントを削除したい場合，［トレース取得ポイント］ビューに表示されたトレース取得ポイントを選択し，コンテキストメニューから［トレース取得ポイントの削除］を選択します。

一覧に表示されたトレース取得ポイントを選択し，［Delete］キーを押しても削除できます。

参考

トレース取得ポイントを選択していない場合は，コンテキストメニューは表示されません。

(4) トレース取得ポイントの指定内容

トレース取得ポイントは，パッケージ，クラス，インタフェース，メソッド（abstract メソッド含む），およびコンストラクタなどの Java 要素に対して設定します。それぞれの指定内容および指定例を次に示します。

参考

トレース取得ポイントの指定内容を次に示します。

<指定形式>,<識別 ID>,<サブクラスフラグ>,<イベント ID>,<トレース取得レベル>#<コメント>

なお，指定内容のうち<イベント ID>，<トレース取得レベル>，および<コメント>については，Java 要素の種類にかかわらず次の値が指定されます。

- <イベント ID>
  「0xae02」
- <トレース取得レベル>
  「A」
- <コメント>
  なし。

表 6-1 トレース取得ポイントの指定内容および指定例

項番	Java 要素	指定内容および指定例
1	パッケージ	<指定形式> 「完全修飾パッケージ名.*」を指定します。 <識別 ID> 「完全修飾パッケージ名.*」を指定します。 <サブクラスフラグ>

項番	Java 要素	指定内容および指定例
		「false」を指定します。 • 指定例 package.*,package.*,false,0xae02,A
2	クラス	<指定形式> 「完全修飾クラス名」を指定します。 <識別 ID> 「完全修飾クラス名」を指定します。 <サブクラスフラグ> 「false」を指定します。 • 指定例 package.Class,package.Class,false,0xae02,A
3	インタフェース	<指定形式> 「完全修飾インタフェース名」を指定します。 <識別 ID> 「完全修飾インタフェース名」を指定します。 <サブクラスフラグ> 「true」を指定します。 • 指定例 package.Interface,package.Interface,true,0xae02,A
4	メソッド（クラス配下）	<指定形式> 「完全修飾メソッド名（引数の型名）」を指定します。※ <識別 ID> 「完全修飾メソッド名」を指定します。 <サブクラスフラグ> 「false」を指定します。 • 指定例 package.Class.method(int),package.Class.method,false,0xae02,A
5	メソッド（インタフェース配下） （abstract メソッドを含む）	<指定形式> 「完全修飾メソッド名（引数の型名）」を指定します。※ <識別 ID> 「完全修飾メソッド名」を指定します。 <サブクラスフラグ> 「true」を指定します。 • 指定例 package.Class.method(int),package.Class.method,true,0xae02,A
6	コンストラクタ（クラス配下）	<指定形式> 「完全修飾クラス名.<init>（引数の型名）」を指定します。※ <識別 ID>

項番	Java 要素	指定内容および指定例
		「完全修飾クラス名.<init>」を指定します。 <サブクラスフラグ> 「false」を指定します。 • 指定例 package.Class.<init>(int),package.Class.<init>,false,0xae02,A

注※　引数の型名は「*（アスタリスク）」となる場合があります。

パッケージ単位の設定や定義ファイルのフォーマットについてはマニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「7.5.3　ユーザ拡張性能解析トレースのトレース対象メソッドの設定」を参照してください。

注意事項

- Developer ではトレース取得ポイントをワークスペース単位で管理します。ワークスペース内に完全修飾名が同名のパッケージ，クラス，メソッド（引数が一致）が複数存在する場合，誤動作するおそれがあります。

  ワークスペース内に完全修飾名が同名のパッケージ，クラス，メソッド（引数が一致）が複数存在する場合，動作の保証はできません。
- Eclipse の操作による［ワークスペースの切り替え］を実行後は，切り替え前のワークスペースで管理していたトレース取得ポイントについてはサポートしていません。
- Eclipse の操作である［プロジェクトを閉じる］を実行するとプロジェクトに紐付いたトレース取得ポイントは［トレース取得ポイント一覧］から削除されます。閉じたプロジェクトを再び Eclipse の操作で［プロジェクトを開く］を実行すると，削除されたトレース取得ポイントは［トレース取得ポイント一覧］の元の位置に復活します。
- ローカルクラス，またはローカル匿名クラスには，トレース取得ポイントを設定できません。
- トレース取得ポイントが設定されたメソッド先頭定義が，エディターで変更されたり，削除されたりした場合，トレース取得ポイントが削除されることがあります。メソッド先頭定義を変更したことによって，トレース取得ポイントが削除された場合，UNDO などでメソッド開始行を元に戻しても，トレース取得ポイントは復元しません。必要に応じて，再度設定し直してください。

6.4.4　ユーザ拡張性能解析トレース設定ファイルのエクスポート

作成，または編集済みのユーザ拡張性能解析トレース設定ファイルをエクスポートし，実行環境へ配置します。

エクスポートの手順を次に示します。

注意事項

- ユーザ拡張性能解析トレース設定ファイルをエクスポートする場合，改行コードは Eclipse の設定に依存します。改行コードを LF（Unix 形式）でエクスポートしたい場合は，エクスポートする前に設定を変更してください。改行コードの設定を変更する手順を次に示します。

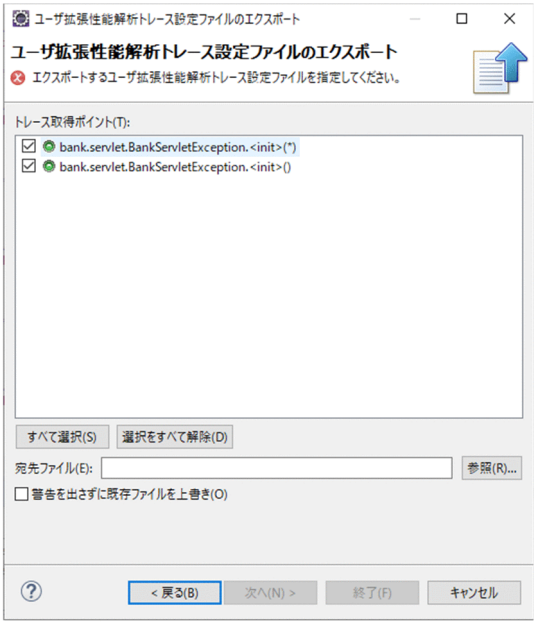
  1. Eclipse のメニューから，［ウィンドウ］－［設定］を選択します。
  2. ［設定］ダイアログの［一般］－［ワークスペース］の［新規テキスト・ファイルの行区切り文字］で，［その他］－［Unix］を選択します。

1. Eclipse のメニューから，［ファイル］－［エクスポート］を選択します。

   ［エクスポート］ダイアログが表示されます。

2. ［ユーザ拡張性能解析トレース］－［ユーザ拡張性能解析トレース設定ファイル］を選択して，［次へ］ボタンをクリックします。

   ［ユーザ拡張性能解析トレース設定ファイルのエクスポート］ダイアログが表示されます。



参考

［ユーザ拡張性能解析トレース設定ファイルのエクスポート］ダイアログは，次に示す方法でも表示できます。

- ［トレース取得ポイント］ビューのビューツールバーで［ユーザ拡張性能解析トレース設定ファイルのエクスポート］ボタンをクリックする。
- ［トレース取得ポイント］ビューのプルダウンメニューで［ユーザ拡張性能解析トレース設定ファイルのエクスポート］を選択する。

3. ［トレース取得ポイント］に表示されたトレース取得ポイントのうち，エクスポートするものを選択します。

4. エクスポートするユーザ拡張性能解析トレース設定ファイルのパスを宛先ファイルに指定します。

   ［参照］ボタンで表示される［ファイルを開く］ダイアログからも指定できます。

   「警告を出さずに既存ファイルを上書き」チェックボックスをチェックすると，指定したパスに同じファイル名のファイルがあった場合に，警告メッセージを出さないで上書きします。

5. ［終了］ボタンをクリックします。

   指定したユーザ拡張性能解析トレース設定ファイルがエクスポートされます。

   エクスポートしたファイルを実行環境に配置してください。

6.4.5　ユーザ拡張性能解析トレースの設定

ユーザ拡張性能解析トレースを使用するには，性能解析トレースを使用するための設定，およびユーザ拡張性能解析トレースを使用するための設定が必要です。

それぞれについて次に示します。

性能解析トレースを使用するために必要な設定

ユーザ拡張性能解析トレースを使用するには，性能解析トレースを使用するための設定が必要です。性能解析トレースを使用するために設定が必要な項目，編集する定義ファイルおよびパラメタの参照先を次に示します。

設定が必要な項目

- Management Server
- パフォーマンストレーサ

編集する定義ファイルおよび指定するパラメタ

マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「7.5.1　性能解析トレースを使用するための設定」を参照してください。

ユーザ拡張性能解析トレースを使用するために必要な設定

ユーザ拡張性能解析トレースを使用するために設定が必要な項目，編集する定義ファイルおよびパラメタの参照先を次に示します。

設定が必要な項目

- J2EE サーバの JavaVM の起動パラメタ

編集する定義ファイルおよび指定するパラメタ

マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.8.23　起動パラメタの設定（J2EE サーバ)」を参照してください。

マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「7.5.2　ユーザ拡張性能解析トレースを使用するための設定」の J2EE サーバの設定に関する説明を参照して，必要なパラメタを運用管理ポータルの画面で拡張起動パラメタに設定してください。

また，運用管理ポータルの画面の「ユーザ拡張性能解析トレースの使用」は「する」を選択し，「ユーザ拡張性能解析トレース設定ファイルの内容」の項目は指定しないで下さい。

Developer で作成したユーザ拡張性能解析トレース設定ファイルを使用する場合，jvm.userprf.ExtendedSetting パラメタが設定されていることが前提となります。

また，jvm.userprf.File パラメタにはユーザ拡張性能解析トレース設定ファイルのファイルパスを指定します。

拡張起動パラメタの設定例を次に示します。

```
-Djvm.userprf.ExtendedSetting=true
-Djvm.userprf.File=<製品のインストールディレクトリ>/CC/server/usrconf/ejb/<実サーバ名>/userprf.cfg
```

6.5　J2EE サーバの起動と停止

J2EE サーバはデバッグモード，および実行モードで起動できます。デバッグモードで起動すると，JSP やサーブレットをデバッグできます。

J2EE サーバの起動，停止，および注意事項について説明します。

6.5.1　J2EE サーバの起動

次のどちらかの方法で J2EE サーバを起動します。

- ［サーバー］ビューで，起動する J2EE サーバを選択し，コンテキストメニューから［起動］または［デバッグ］を選択します。
- ［サーバー］ビューで，起動する J2EE サーバを選択し，［サーバー］ビューのツールバーの［サーバーを始動］または［デバッグ・モードでサーバーを始動］ボタンをクリックします。

［サーバー］ビューで J2EE サーバを起動すると，［サーバー］ビューに起動したサーバの状態が表示されます。

注意事項

usrconf.properties（J2EE サーバ用ユーザプロパティファイル）などの更新について

WTP コネクタを使用して Eclipse から J2EE サーバを開始する場合，Management Server を使用して J2EE サーバの設定情報を配布します。

このため，usrconf.cfg（J2EE サーバ用オプション定義ファイル），usrconf.properties（J2EE サーバ用ユーザプロパティファイル）などを直接編集して指定した設定情報については破棄されます。J2EE サーバの設定は，運用管理ポータルを使用して指定してください。

運用管理ポータルを使用した J2EE サーバの設定については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.8　論理 J2EE サーバの定義」を参照してください。

6.5.2　J2EE サーバの停止

次のどちらかの方法で J2EE サーバを停止します。

- ［サーバー］ビューで，起動している J2EE サーバを選択し，コンテキストメニューから［停止］を選択します。
- ［サーバー］ビューで，起動している J2EE サーバを選択し，［サーバー］ビューのツールバーの［サーバーを停止］ボタンをクリックします。

［サーバー］ビューで J2EE サーバを停止すると，［サーバー］ビューに停止したサーバの状態が表示されます。

6.5.3　J2EE サーバの起動および停止時の注意事項

J2EE サーバを起動および停止する場合の注意事項を次に示します。

- WTP コネクタ以外で J2EE サーバを起動した場合，WTP コネクタの操作をすると，エラーが発生するおそれがあります。WTP コネクタで J2EE サーバを操作する前に操作対象の J2EE サーバが停止されていることを確認してください。
- Eclipse の J2EE サーバのモニタリング機能，およびロケーションの切り替えは使用できません。［モニタリング］ページで［追加］ボタンを選択したり，［一般］ページで［ロケーションの切り替え］ボタンを選択したりしないでください。
- J2EE サーバの起動，デバッグ，および停止の操作を，連続して実行（ボタンの連打など）しないでください。エラーが発生するおそれがあります。
- J2EE サーバを起動している状態で Eclipse を終了すると，エラーが発生するおそれがあります。

6.6　J2EE サーバへのプロジェクトの公開

J2EE サーバへ作成したプロジェクトを公開（デプロイ）できます。公開先のサーバがローカルホストに存在する場合は，展開ディレクトリ形式で公開されます。また，公開先のサーバがリモートホストに存在する場合は，アーカイブ形式で公開されます。

J2EE サーバへプロジェクトを公開する手順を次に示します。

1. **［サーバー］ビューでプロジェクト公開先のサーバを選択して，コンテキストメニューから［追加と削除］を選択します。**
   ［追加と削除］ダイアログが表示されます。

2. **［使用可能］で追加するプロジェクトを選択して，［追加］ボタンをクリックします。**
   追加したプロジェクトが［構成済み］に表示されます。

3. **［終了］ボタンをクリックします。**
   ［サーバー］ビューに，追加したプロジェクトが表示されます。J2EE サーバが起動している場合は，プロジェクトが自動的に公開されます（手順 4.の操作は不要です）。

4. **J2EE サーバが起動していない場合は，［サーバー］ビューでプロジェクト公開先のサーバを選択して，コンテキストメニューから［公開］を選択します。**
   ［サーバー］ビューの［状況］に，公開したプロジェクトの状況が表示されます。

> **参考**
>
> 手順 4.の操作を実施すると，すでに公開しているプロジェクトを入れ替えまたはリロードすることができます。
>
> また，ソースコードの修正など，プロジェクトの変更を J2EE アプリケーションに反映することもできます。
>
> アプリケーションがリロードできない場合およびリロードが禁止されたアプリケーションの場合，次のどちらかの方法でアプリケーションを更新してください。なお，この場合，サーバの公開設定を「自動公開しない」に設定してください。
>
> - アプリケーションを更新後，配布中のアプリケーションをサーバから削除してから再度公開する。
> - 更新したアプリケーションを再公開後，アプリケーション実行前に J2EE サーバを再起動する。

●J2EE サーバへのプロジェクトの公開時の注意事項

J2EE サーバへのプロジェクトの公開時には，次の点に注意してください。

- ［追加と削除］ダイアログで表示されるエンタープライズアプリケーションプロジェクトには，少なくとも一つは，動的 Web プロジェクトまたは EJB プロジェクトを含む必要があります。

- 動的 Web プロジェクトおよび EJB プロジェクトを単体で公開する場合，WTP コネクタは擬似的にアプリケーション構造を作成します。アプリケーションディレクトリ名または EAR ファイル名には，プロジェクト名が使用されます。
- プロジェクトの依存関係や構成，配備記述子（application.xml）を変更した場合は，変更内容を正しく反映するために［公開］するのではなく，［クリーン］を実施してください。
- プロジェクトの追加時，すでに同じ名前の J2EE アプリケーションが J2EE サーバ上にあると，正常にプロジェクトを公開できません。

  この場合，運用管理ポータル，またはサーバ管理コマンドで J2EE アプリケーションを削除するか，または J2EE アプリケーション名を変更してください。
- ブレークポイントで中断しているアプリケーションがある状態でプロジェクトを公開すると，タイムアウトが発生して，プロジェクトが正常に公開されない場合があります。
- 公開処理中に Eclipse を終了したり，公開中のプロジェクトに対して［プロジェクトを閉じる］の操作をしたりしないでください。エラーが発生するおそれがあります。
- サーバの概要ページで編集中の場合，公開操作の前に編集内容を保存してください。編集中の状態で公開操作した場合は，一度サーバエディタを閉じてください。
- プロジェクトの公開でエラーが発生した場合に，java.lang.OutOfMemoryError によって運用管理エージェントが終了していることがあります。その場合は，運用管理エージェントの Java ヒープの最大サイズを変更したあとに，操作してください。なお，変更後の値は，アプリケーションのファイルサイズの 5 倍程度が目安となります。

6.7　J2EE アプリケーションのデバッグと実行

Eclipse から J2EE アプリケーションをデバッグしたり実行したりする方法について説明します。

6.7.1　J2EE アプリケーションのデバッグ

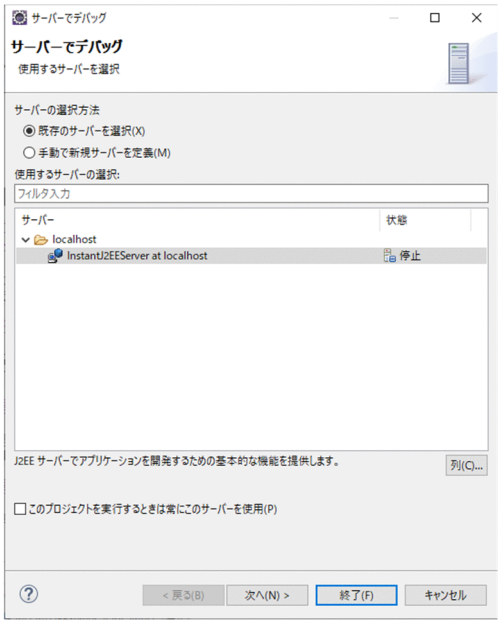
［サーバーでデバッグ］ダイアログからデバッグすると，J2EE サーバの起動，J2EE サーバへのプロジェクトの公開，および Web ブラウザへのアクセスを一度で実施でき，効率的にデバッグできます。

(1)　［サーバーでデバッグ］ダイアログからのデバッグ

すでにサーバが存在する場合に［サーバーでデバッグ］ダイアログからデバッグする手順を次に示します。

1. ［プロジェクト・エクスプローラー］ビューで，デバッグ対象のリソースを選択して，コンテキストメニューから［デバッグ］－［サーバーでデバッグ］を選択します。

   ［サーバーでデバッグ］ダイアログが表示されます。



   デバッグ対象のリソースについては，「(2)　デバッグの対象」を参照してください。

2. 次の項目を指定します。

項目名	指定値
使用するサーバーの選択	登録済みの J2EE サーバを指定します。 Cosminexus J2EE サーバを指定してください。
このプロジェクトを実行するときは常にこのサーバーを使用	このプロジェクトを実行するときには常に［使用するサーバーの選択］で選択した J2EE サーバを使用するかどうかを指定します。 • このプロジェクトを実行するときに常にこのサーバを使用する場合 チェックします。

項目名	指定値
	• このプロジェクトを実行するときに常にこのサーバを使用しない場合 チェックしません。

3. [終了] ボタンをクリックします。

J2EE サーバの起動，およびプロジェクトの公開が完了すると，Web ブラウザが表示されます。

(2) デバッグの対象

[サーバーでデバッグ] ダイアログからデバッグする場合の，デバッグ対象となるリソースを次の表に示します。

リソース	ブラウザ表示	説明
エンタープライズアプリケーションプロジェクト	△	モジュールプロジェクトに動的 Web プロジェクトがある場合，そのコンテキストルートの URL が Web ブラウザに表示されます。※ 複数の動的 Web プロジェクトがある場合，プロジェクト名のアルファベット順で先頭の動的 Web プロジェクトが，Web ブラウザ表示の対象になります。先頭の動的 Web プロジェクト以外を対象とする場合は，動的 Web プロジェクトを選択して，デバッグを実行してください。 (例) http://localhost/<コンテキストルート>/ モジュールプロジェクトに動的 Web プロジェクトがない場合は，Web ブラウザは表示されません。
EJB プロジェクト	×	Web ブラウザは表示されません。
動的 Web プロジェクト	○	指定したプロジェクトのコンテキストルートの URL が Web ブラウザに表示されます。※ (例) http://localhost/<コンテキストルート>/
動的 Web プロジェクトのコンテンツディレクトリ下の JSP または HTML ファイル	○	選択したファイルが Web ブラウザに表示されます。 (例) http://localhost/<コンテキストルート>/<ファイルパス>
動的 Web プロジェクトの Java ソースディレクトリ下のサーブレットクラス	○	選択したサーブレットクラスが Web ブラウザに表示されます。 (例) • サーブレットマッピングタグが指定されている場合 http://localhost/<コンテキストルート>/<サーブレット URL> • サーブレットマッピングタグが指定されていない場合 http://localhost/<コンテキストルート>/servlet/<サーブレットクラスの完全修飾名>
その他	×	上記以外のリソースは，デバッグの対象にはなりません。

（凡例）

○：Web ブラウザを表示する

△：Web ブラウザの表示に制限がある

×：Web ブラウザを表示しない

注※

動的 Web プロジェクトに［ウェルカム］ページがない場合は，エラーになります（エラーステータスコード 404）。

(3)　J2EE アプリケーションのデバッグ時の注意事項

J2EE アプリケーションのデバッグ時には，次の点に注意してください。

- コンテキストルートには，application.xml で指定した値またはプロジェクト名が使用されます。動的 Web プロジェクトの［Web プロジェクトの設定］ページで指定した値は使用されません。なお，コンテキストルートには，URI（RFC3986）で使用できる文字だけを指定できます。
- ワークスペース内の複数のプロジェクトに同じ完全修飾名のクラスがある場合，デバッグ実行時にデバッグ対象外のプロジェクトのソースファイルに設定されたブレークポイントが有効になることがあります。
- ［サーバーでデバッグ］または［サーバーで実行］を実行した場合，アプリケーションが開始する前に Web ブラウザが起動されることで，Web ページのアクセスに失敗する場合があります。この場合，アプリケーションの開始後，再度アクセスしてください。
- 一度表示した Web 画面を［サーバーでデバッグ］操作で表示させた場合，設定済みのブレークポイントで停止しないときがあります。この場合，ブラウザのリロード操作を実行することによって，Web 画面の表示およびブレークポイントでの停止が正しく実行されます。

6.7.2　J2EE アプリケーションの実行

［サーバーで実行］ダイアログから実行すると，実行モードで J2EE アプリケーションを実行できます。［サーバーで実行］ダイアログからの手順や注意事項は，デバッグの場合と同じです。詳細については「6.7.1　J2EE アプリケーションのデバッグ」を参照してください。なお，操作や説明中の「デバッグ」は「実行」に読み替えてください。

6.7.3　サーバ管理コマンドで起動した J2EE サーバでのデバッグ

Eclipse では，サーバ管理コマンドを使って，デバッグ起動している J2EE サーバにある J2EE アプリケーションをデバッグできます。

ここでは，サーバ管理コマンドを使用した J2EE サーバのデバッグ起動から Eclipse を使用したデバッグまでの手順を説明します。

(1) J2EE サーバのデバッグ起動

サーバ管理コマンドを使用して J2EE サーバをデバッグ起動します。

1. <Developer のインストールディレクトリ>¥CC¥server¥usrconf¥ejb¥<デバッグ対象となる J2EE サーバの名称>¥usrconf.cfg ファイルに次の内容を記述します。

```
add.jvm.arg=-agentlib:jdwp=transport=dt_socket,server=y,address=*:<デバッグ・ポート番号>,
suspend=n
```

開発環境インスタントセットアップ機能で構築した環境を使用する場合，デバッグ・ポート番号は「3999」で設定されています。

2. サーバ管理コマンドで J2EE サーバを起動します。

サーバ管理コマンドで J2EE サーバを起動するには，次のコマンドを実行します。

```
cjstartsv［<デバッグするJ2EEサーバの名称※>］
```

注※

開発環境インスタントセットアップ機能で構築した環境を使用する場合は，デバッグする J2EE サーバの名称に「cmx_InstantWebSystem_unit1_J2EE_01」を指定します。

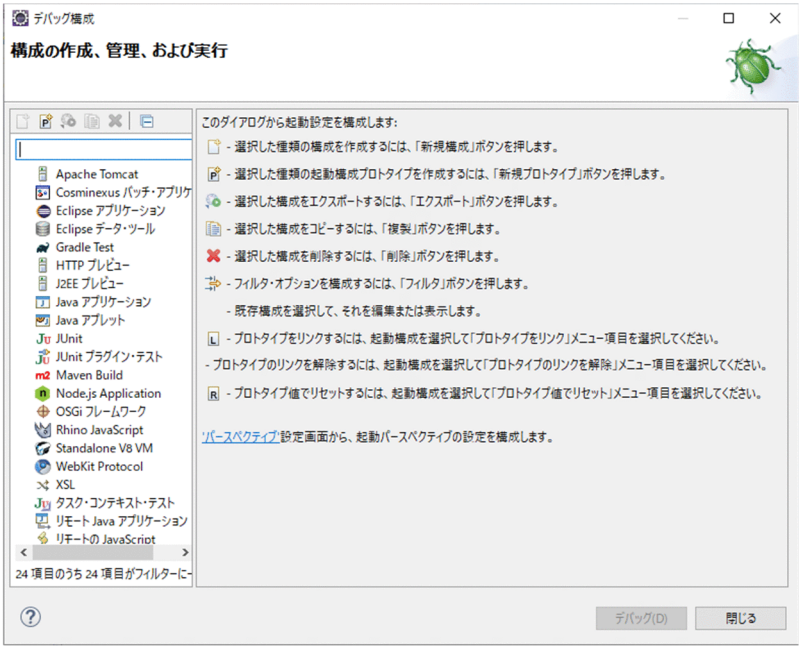
cjstartsv コマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cjstartsv（J2EE サーバの開始)」を参照してください。

(2) プロジェクトのデバッグの設定

デバッグに必要な起動構成を作成します。

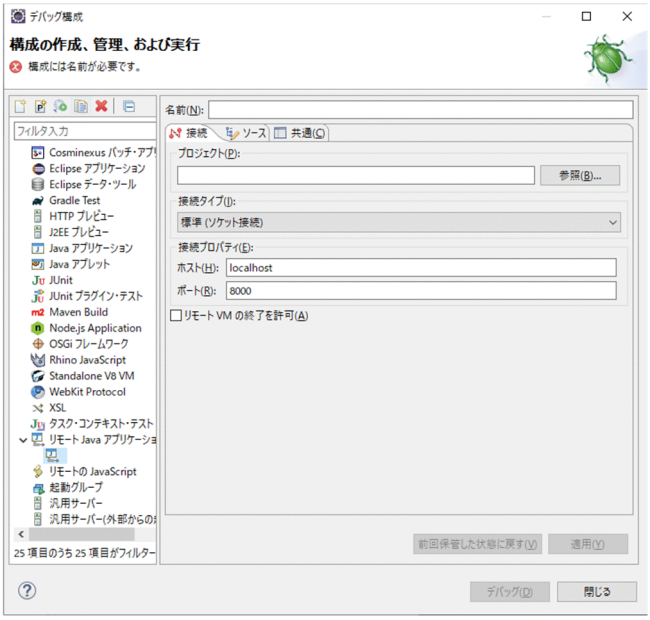
1. ［プロジェクト・エクスプローラー］ビューでデバッグするプロジェクトを選択して，Eclipse のメニューから［実行］－［デバッグの構成］を選択します。

［デバッグ構成］ダイアログが表示されます。



2. 左ペインのツリーで［リモート Java アプリケーション］を選択して，ツールバーの［ （新規の起動構成)］をクリックします。
   右ペインに起動構成を作成するページが表示されます。

3. ［接続］タブを選択し，次の内容を指定します。



項目名	指定値
名前	任意の名称を指定します。
プロジェクト	デバッグするプロジェクトのプロジェクト名を指定します。※
接続タイプ	「標準（ソケット接続)」を選択します。

項目名		指定値
接続プロパティ	ホスト	接続する J2EE サーバのホスト名を指定します。 開発環境インスタントセットアップ機能で構築した環境を使用する場合は，「localhost」を指定します。
	ポート	usrconf.cfg ファイルに指定したデバッグ・ポート番号を指定します。 開発環境インスタントセットアップ機能で構築した環境を使用する場合は，セットアップ時または設定変更時に設定したデバッグ接続のためのポート番号を指定します。 なお，開発環境インスタントセットアップ機能の標準セットアップを実行した場合は，「3999」を指定します。

注※

エンタープライズアプリケーションプロジェクトをデバッグする場合は，プロジェクトを空白にして，［ソース］タブの［ソース・ルックアップ・パス］に使用するソースを追加します。

4. ［閉じる］ボタンをクリックします。

［変更を保管します］ダイアログが表示された場合は，［はい］ボタンをクリックしてください。［デバッグ構成］ダイアログが閉じて，デバッグのための起動構成が作成されます。

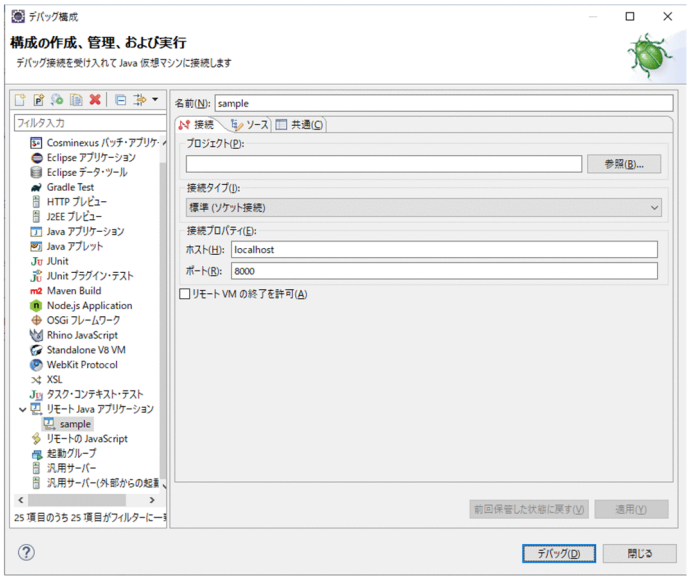
(3) デバッグの実行

Eclipse からデバッグを実行します。

1. ［プロジェクト・エクスプローラー］ビューでデバッグするプロジェクトの Java ソースファイルを開いて，ブレークポイントを設定します。

2. Eclipse のメニューから［実行］－［デバッグの構成］を選択します。

［デバッグ構成］ダイアログが表示されます。



3. 作成した起動構成を選択して，［デバッグ］ボタンをクリックします。

デバッグが開始されます。J2EE アプリケーションを実行してブレークポイントに達すると，［デバッグ］パースペクティブが表示されます。［デバッグ］パースペクティブの［デバッグ］ビューには，外部起動サーバの起動構成が表示されます。

参考

［デバッグ構成］ダイアログからデバッグしたあとに，再度デバッグする場合は Eclipse のメニューバーにある［  （デバッグ)］のプルダウンメニューから起動構成を選択することで開始することもできます。

4. **デバッグが終わったら，［デバッグ］ビューのツールバーにある［  （切断)］をクリックします。**

デバッグが終了します。

6.8 Eclipse と Eclipse 以外のツールを併用する場合の注意事項

Eclipse と Eclipse 以外のツールの両方を使用する場合の注意事項について説明します。

Eclipse 以外のツールで J2EE サーバを起動している場合

Eclipse から J2EE サーバを起動するときは，J2EE サーバが停止していることを確認してください。

Eclipse 以外のツールで起動した J2EE サーバを，Eclipse から操作することはできません。Eclipse で J2EE アプリケーションを操作する場合は Eclipse で J2EE サーバを起動してください。

Eclipse 以外のツールで同じ名称の J2EE アプリケーションをインポートしている場合

Eclipse 以外のツールで，同じ名称で同じ形式の J2EE アプリケーションがすでにインポートされている場合に，Eclipse で J2EE サーバにプロジェクトをデプロイすると，J2EE アプリケーションの入れ替えが実施されます。インポート済みの J2EE アプリケーションのデプロイ形式がアーカイブ形式の場合は，J2EE アプリケーションはリデプロイされます。また，インポート済みの J2EE アプリケーションのデプロイ形式が展開ディレクトリ形式の場合は，J2EE アプリケーションはリロードされます。このとき，J2EE アプリケーションの構成が異なる場合は，正しく入れ替えできません。入れ替えができなかった場合は，Eclipse でいったん J2EE アプリケーションをアンデプロイしてから，再度デプロイしてください。

7 実行環境への J2EE アプリケーションの配布

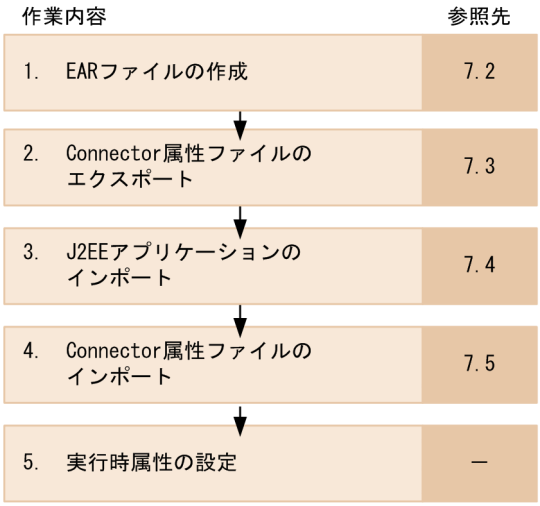
この章では，開発した J2EE アプリケーションを実行環境に配布する方法について説明します。

7.1　J2EE アプリケーション配布の流れ

開発環境にある J2EE アプリケーションをエクスポートして，実行環境に配布します。J2EE アプリケーション配布の流れを次に示します。

なお，実行環境での初回リクエストのレスポンスをスムーズにしたい場合は，EAR ファイル作成前に JSP 事前コンパイルを実行してください。マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「2.5　JSP 事前コンパイル機能とコンパイル結果の保持」を参照してください。

図 7-1　J2EE アプリケーション配布の流れ



（凡例）－：なし

それぞれの作業の概要を説明します。

1. EAR ファイルの作成

ビルドファイルを使用してプロジェクトをビルドして，EAR ファイルを作成します。詳細は，「7.2 EAR ファイルの作成」を参照してください。

2. Connector 属性ファイルのエクスポート※

開発環境で設定したリソースアダプタの属性を取得するために，Connector 属性ファイルをエクスポートします。エクスポートにはサーバ管理コマンドを使用します。詳細は，「7.3　Connector 属性ファイルのエクスポート」を参照してください。

3. J2EE アプリケーションのインポート

ビルドファイルを使用して作成した J2EE アプリケーション（EAR ファイル）を実行環境にインポートします。インポートには，サーバ管理コマンドまたは運用管理ポータルを使用します。詳細は，「7.4 実行環境への J2EE アプリケーションのインポート」を参照してください。

4. Connector 属性ファイルのインポート※

Connector 属性ファイルを，実行環境にインポートします。インポートには，サーバ管理コマンドまたは運用管理ポータルを使用します。詳細は，「7.5　実行環境への Connector 属性ファイルのインポート」を参照してください。

5. 実行時属性の設定※

実行環境にインポートした J2EE アプリケーションに実行時属性を設定します。また，必要に応じて，J2EE リソースにも実行時属性を設定します。J2EE アプリケーションおよび J2EE リソースの実行時属性の設定には，サーバ管理コマンドを使用します。実行時属性の設定方法については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」を参照してください。

注※

cosminexus.xml を含むアプリケーションでは J2EE サーバ上でのコマンドによるアプリケーションの属性の設定は不要です。

以降の節では，この流れに沿って J2EE アプリケーションを配布する手順を説明します。

7.2　EAR ファイルの作成

開発した J2EE アプリケーションを実行環境に配布する場合には，ビルドファイル（build.xml）を使用して Eclipse のプロジェクトをビルドし，EAR ファイルを作成します。ここで作成した EAR ファイルを実行環境に配布します。なお，Java ソースファイルのコンパイルには，アプリケーションサーバの javac コマンドおよび J2EE ライブラリが使用されます。

ここでは，build.xml の作成手順および EAR ファイルの作成手順を説明します。

7.2.1　ビルドファイルの作成

ビルドファイル（build.xml）の作成手順を説明します。

1. **［プロジェクト・エクスプローラー］ビューで build.xml を追加するプロジェクトを選択して，コンテキストメニューから［新規］－［ファイル］を選択します。**

   ［新規ファイル］ダイアログが表示されます。

2. **build.xml を追加するプロジェクトを選択した状態で［ファイル名］に「build.xml」を入力して，［終了］ボタンをクリックします。**

   選択したプロジェクトに build.xml が追加されます。

7.2.2　ビルドファイルの編集・実行

プロジェクトの種類によって，ビルドファイル（build.xml）に記述する内容が異なります。それぞれのプロジェクトの種類に合わせて，ビルドファイルを編集してください。ビルドファイルを作成する順序は次のとおりです。

1. EJB プロジェクトのビルド
2. 動的 Web プロジェクトのビルド
3. エンタープライズアプリケーションプロジェクトのビルド

この流れに沿って，各プロジェクトのビルドファイルの編集と実行について説明します。

(1) EJB プロジェクトのビルド

EJB プロジェクトのビルドファイルに次の内容を記述します。ビルドファイルを実行すると，EJB-JAR ファイルが生成されます。

表 7-1 EJB プロジェクトのビルドファイルの例

行番号	記述例
1	<?xml version="1.0" encoding="UTF-8"?>
2	<project name="Bank_EJB" default="create" basedir="./">
3	<property environment="myEnv"/>
4	<property name="classPath" value="${myEnv.COSMINEXUS_HOME}/CC/client/lib/j2ee-
5	javax.jar"/>
6	<property name="ejbArchiveName" value="Bank_EJB.jar"/>
7	<property name="javacPath" value="${myEnv.COSMINEXUS_HOME}/jdk/bin/javac.exe"/>
8	<property name="tempFolder" value="ant"/>
9	<target name="create">
10	<delete file="./${ejbArchiveName}"/>
11	<mkdir dir="./${tempFolder}"/>
12	<copy todir="./${tempFolder}">
13	<fileset dir="./src" excludes="**/*.java"/>
14	</copy>
15	<javac srcdir="./src" destdir="./${tempFolder}" executable="${javacPath}" classpath="${classPath}"/>
16	<jar destfile="./${ejbArchiveName}" basedir="./${tempFolder}"/>
17	<delete dir="./${tempFolder}"/>
18	</target>
	</project>

ビルドファイルの各行の意味は，次のとおりです。

表 7-2 EJB プロジェクトのビルドファイルの記述内容

行番号	記述内容の意味
1	XML 宣言です。
2	ビルドファイルのルートを指定します。
3	プロパティに，ビルドファイルからシステム環境変数を参照する設定を追加します。
4	プロパティに，アプリケーションサーバの J2EE ライブラリのパスを指定します。

行番号	記述内容の意味
5	プロパティに，EJB プロジェクトで生成するアーカイブファイル名の指定を追加します。
6	プロパティに，アプリケーションサーバの javac コマンドのパスを指定します。
7	プロパティに，ビルドで使用する一時ディレクトリ名を指定します。ディレクトリ名は任意の名称を指定します。
8	ターゲット「create」の開始タグです。
9	既存のアーカイブファイルを削除します。
10	ビルド用の一時ディレクトリを作成します。
11	ビルド用の一時ディレクトリに，アーカイブに含めるファイルをコピーします。
12	Java ソースをコピー対象外にします。
13	11 行目のコピータスクの終了タグです。
14	Java ソースをコンパイルします。コンパイルにはアプリケーションサーバの javac コマンドを使用します。
15	ビルド用の一時ディレクトリ以下のフォルダおよびファイルをアーカイブして，jar ファイルを生成します。
16	ビルド用の一時ディレクトリを削除します。
17	8 行目のターゲット「create」の終了タグです。
18	ビルドファイルの終了タグです。

(2) 動的 Web プロジェクトのビルド

動的 Web プロジェクトのビルドファイルに，次の内容を記述します。ビルドファイルを実行すると，WAR ファイルが生成されます。

なお，動的 Web プロジェクトのビルドファイルは EJB プロジェクトのアーカイブファイルを参照しています。動的 Web プロジェクトをビルドする前に，必ず EJB プロジェクトをビルドしてください。

表 7-3 動的 Web プロジェクトのビルドファイルの例

行番号	記述例
1	<?xml version="1.0" encoding="UTF-8"?>
2	<project name="Bank_Web" default="create" basedir="./">
3	<property environment="myEnv"/>
4	<property name="cjjspcdir" value="${myEnv.COSMINEXUS_HOME}/CC/web/bin"/>
5	<property name="ejbJarPath" value="../Bank_EJB/Bank_EJB.jar"/>
6	<property name="classPath" value="${myEnv.COSMINEXUS_HOME}/CC/client/lib/j2ee-javax.jar;$
7	{ejbJarPath}"/>
8	<property name="javacPath" value="${myEnv.COSMINEXUS_HOME}/jdk/bin/javac.exe"/>
9	<property name="tempFolder" value="ant"/>
10	<property name="webArchiveName" value="Bank_Web.war"/>
11	<property name="webProjectPath" value="E:/eclipse/workspace/Bank_Web/"/>
12	<property name="webRoot" value="WebRoot"/>

行番号	記述例
13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	<target name="compile"> <exec executable="${cjjspcdir}/cjjspc.bat" newenvironment="true"> <arg line='-classpath ${ejbJarPath}'/> <arg line='-source 1.5'/> <arg line='-pageencoding UTF-8'/> <arg line='-root "${webProjectPath}/${tempFolder}"'/> </exec> </target> <target name="create"> <delete file="./${webArchiveName}"/> <mkdir dir="./${tempFolder}"/> <copy todir="./${tempFolder}"> <fileset dir="./${webRoot}" excludes="**/classes/**/*.class"/> </copy> <javac srcdir="./src" destdir="./${tempFolder}/WEB-INF/classes" executable="${javacPath}" classpath="${classPath}"/> <antcall target="compile"/> <war destfile="./${webArchiveName}" basedir="./${tempFolder}" webxml="./${tempFolder}/WEB-INF/web.xml" excludes="WEB-INF/web.xml"/> <delete dir="./${tempFolder}"/> </target> </project>

ビルドファイルの各行の意味は，次のとおりです。

表 7-4　動的 Web プロジェクトのビルドファイルの記述内容

行番号	記述内容の意味
1	XML 宣言です。
2	ビルドファイルのルートを指定します。
3	プロパティに，ビルドファイルからシステム環境変数を参照する指定を追加します。
4	プロパティに，アプリケーションサーバの JSP 事前コンパイル用コマンドのパスを指定します。
5	プロパティに，EJB プロジェクトのアーカイブのパスを指定します。
6	プロパティに，アプリケーションサーバの J2EE ライブラリ，および EJB プロジェクトのアーカイブのパスを指定します。
7	プロパティに，アプリケーションサーバの javac コマンドのパスを指定します。
8	プロパティに，ビルドで使用する一時ディレクトリ名を指定します。ディレクトリ名は任意の名称を指定します。
9	プロパティに，動的 Web プロジェクトで生成するアーカイブファイル名を指定します。
10	プロパティに，動的 Web プロジェクトの絶対パスを指定します。

行番号	記述内容の意味
11	プロパティに，動的 Web プロジェクトのルートを指定します。
12	JSP 事前コンパイル用のターゲット「compile」の開始タグです。
13	アプリケーションサーバの JSP 事前コンパイル用のコマンドを実行します。
14	JSP 事前コンパイルで EJB プロジェクトのクラスを参照する場合は，EJB プロジェクトのクラスパスを指定します。
15	コマンドの引数を指定します。指定したバージョンの Java 言語を使用して事前コンパイルします。
16	コマンドの引数として，JSP のデフォルトの文字エンコーディングを指定します。
17	コマンドの引数として，動的 Web プロジェクトのルートに一時ディレクトリを指定します。
18	12 行目のコマンド実行の終了タグです。
19	11 行目のターゲット「compile」の終了タグです。
20	ターゲット「create」のルートを指定します。
21	既存のアーカイブファイルを削除します。
22	ビルド用の一時ディレクトリを作成します。
23	アーカイブに含めるファイルを一時ディレクトリにコピーします。
24	Java クラスをコピー対象外にします。
25	21 行目のコピータスクの終了タグです。
26	Java ソースをコンパイルします。コンパイルにはアプリケーションサーバの javac コマンドを使用します。
27	JSP 事前コンパイル用のターゲット「compile」を呼び出します。JSP 事前コンパイルをしない場合は記述しません。
28	一時ディレクトリ以下のディレクトリおよびファイルをアーカイブして，WAR ファイルを生成します。
29	一時ディレクトリを削除します。
30	18 行目のターゲット「create」の終了タグです。
31	ビルドファイルの終了タグです。

JSP 事前コンパイルで，配布先の J2EE サーバで JSP デバッグ機能が有効な場合，cjjspc コマンドに-debugging オプションを指定する必要があります。詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「2.4.2　JSP デバッグ機能の使用手順」を参照してください。

(3)　エンタープライズアプリケーションプロジェクトのビルド

エンタープライズアプリケーションプロジェクトのビルドファイルに，次の内容を記述します。ビルドファイルを実行すると，EAR ファイルが生成されます。

なお，エンタープライズアプリケーションプロジェクトをビルドすると，EJB プロジェクト，動的 Web プロジェクト，エンタープライズアプリケーションプロジェクトの順にビルドされて，EJB-JAR ファイルと WAR ファイルを含む EAR ファイルが生成されます。

エンタープライズアプリケーションプロジェクトをビルドする前に，必ず EJB プロジェクトおよび動的 Web プロジェクトのビルドファイルを作成してください。

表 7-5　エンタープライズアプリケーションプロジェクトのビルドファイルの例

行番号	記述例
1	<?xml version="1.0" encoding="UTF-8"?>
2	<project name="Bank" default="create" basedir="./">
3	<property name="earArchiveName" value="bank.ear"/>
4	<property name="ejbArchiveName" value="Bank_EJB.jar"/>
5	<property name="ejbProjectName" value="Bank_EJB"/>
6	<property name="webArchiveName" value="Bank_Web.war"/>
7	<property name="webProjectName" value="Bank_Web"/>
8	<property name="excludes" value=".classpath,.mymetadata,.project,build-user.xml"/>
9	<target name="create">
10	<ant antfile="../${ejbProjectName}/build.xml" dir="../${ejbProjectName}" inheritall="false"/>
11	<ant antfile="../${webProjectName}/build.xml" dir="../${webProjectName}" inheritall="false"/>
12	<delete file="./${earArchiveName}"/>
13 14	<ear destfile="./${earArchiveName}" basedir="./" appxml="./META-INF/application.xml" excludes="${excludes}">
15	<fileset file="../${ejbProjectName}/${ejbArchiveName}"/>
16	<fileset file="../${webProjectName}/${webArchiveName}"/>
17	</ear>
18	</target>
	</project>

ビルドファイルの各行の意味は，次のとおりです。

表 7-6　エンタープライズアプリケーションプロジェクトのビルドファイルの記述内容

行番号	記述例
1	XML 宣言です。
2	ビルドファイルのルートを指定します。
3	プロパティに，エンタープライズアプリケーションプロジェクトで生成するアーカイブファイル名を指定します。
4	プロパティに，EJB プロジェクトで生成するアーカイブファイル名を指定します。
5	プロパティに，EJB プロジェクトの名称を指定します。
6	プロパティに，動的 Web プロジェクトで生成するアーカイブファイル名を指定します。
7	プロパティに，動的 Web プロジェクトの名称を指定します。
8	プロパティに，EAR ファイル生成時に除くファイルを指定します。
9	ターゲット「create」の開始タグです。
10	EJB プロジェクトのビルドファイルを実行します。

行番号	記述例
11	動的 Web プロジェクトのビルドファイルを実行します。
12	既存のアーカイブファイルを削除します。
13	EAR ファイルを生成します。
14	EAR ファイルに含める EJB-JAR ファイルを指定します。
15	EAR ファイルに含める WAR ファイルを指定します。
16	13 行目の ear タスクの終了タグです。
17	9 行目のターゲット「create」の終了タグです。
18	ビルドファイルの終了タグです。

7.3　Connector 属性ファイルのエクスポート

開発環境でリソースアダプタの属性を設定している場合，設定した属性情報を取得するために，Connector 属性ファイルをエクスポートします。エクスポートした Connector 属性ファイルは，実行環境にインポートします。

Connector 属性ファイルのエクスポートにはサーバ管理コマンドを使用します。Connector 属性ファイルのエクスポートについては，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「3.5.1　J2EE リソースのプロパティの設定手順」を参照してください。

参考

アプリケーションに含まれるリソースアダプタの属性は，アプリケーション統合属性ファイルの Connector 属性に設定されています。アプリケーション統合属性ファイルをエクスポートして，実行環境にインポートします。

7.4　実行環境への J2EE アプリケーションのインポート

ビルドファイルを使用して作成した J2EE アプリケーション（EAR ファイル）を実行環境にインポートします。なお，EAR ファイルを実行環境にインポートするときの方法には，EAR ファイルを，アーカイブ形式としてインポートする方法と，展開ディレクトリ形式としてインポートする方法があります。どちらの方法の場合も，インポートにはサーバ管理コマンドまたは運用管理ポータルを使用できます。J2EE アプリケーションの形式ごとに，インポート方法を説明します。

- アーカイブ形式の J2EE アプリケーションとしてインポートする
  - サーバ管理コマンドを使用する場合

    cjimportapp コマンドに-f オプションを指定して，コマンドを実行します。-f オプションには EAR ファイルを指定します。
  - 運用管理ポータルを使用する場合

    運用管理ポータルの［J2EE アプリケーションのインポート］画面を使用します。

  サーバ管理コマンドでのインポート手順については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「8.1.1　アーカイブ形式の J2EE アプリケーションのインポート」を参照してください。また，運用管理ポータルでのインポート手順については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「12.3.3　J2EE アプリケーションのインポート」を参照してください。
- 展開ディレクトリ形式の J2EE アプリケーションとしてインポートする
  - サーバ管理コマンドを使用する場合

    cjimportapp コマンドに-f オプションおよび-d オプションを指定して，コマンドを実行します。-f オプションには EAR ファイルを，-d オプションには EAR ファイルの展開先ディレクトリを指定します。
  - 運用管理ポータルを使用する場合

    EAR ファイルをいったん解凍してからインポートします。インポートには，［アプリケーションディレクトリのインポート］画面を使用します。

  サーバ管理コマンドでのインポート手順については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「8.1.2　展開ディレクトリ形式の J2EE アプリケーションのインポート」を参照してください。また，運用管理ポータルでのインポート手順については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「12.3.4　アプリケーションディレクトリのインポート」を参照してください。

7.5　実行環境への Connector 属性ファイルのインポート

開発環境でエクスポートした Connector 属性ファイルを実行環境にインポートします。Connector 属性ファイルを実行環境にインポートするには，サーバ管理コマンドを使用します。

Connector 属性ファイルのインポートについては，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「3.5.1　J2EE リソースのプロパティの設定手順」を参照してください。

参考

アプリケーションに含まれるリソースアダプタの属性は，アプリケーション統合属性ファイルの Connector 属性に設定されています。アプリケーション統合属性ファイルを実行環境にインポートします。

8 Web サービスの開発

この章では，Eclipse プラグインを使用した Web サービスの開発として，Web サービスおよび Web サービスクライアントの作成から実行までの手順，および注意事項について説明します。

8.1 Web サービスの開発の前提環境

Web サービスの種類には，Web アプリケーションアーカイブ（WAR ファイル）に含まれる POJO の Web サービスと，EJB-JAR ファイルに含まれる EJB の Web サービスがあります。これらの Web サービスは，Eclipse を使用して，次に示す方法で開発できます。

- WSDL を起点とした Web サービスの開発
  WSDL を作成して，これを起点として Web サービスを開発します。
- SEI（サービスエンドポイントインタフェース）を起点とした Web サービスの開発
  Web サービス実装クラスを作成して，これを起点として Web サービスを開発します。
- Web サービスクライアントの開発
  WSDL を基にして Web サービスクライアントを開発します。

8.1.1 Web サービス開発時の前提条件

Web サービス開発時の前提条件を次に示します。

(1) Web サービスの開発をする場合の実行時の権限

UAC（ユーザアカウント制御）が有効な Windows で，Web サービス開発プラグインを実行する場合の注意事項を次に示します。

(a) 管理者が実行する場合の注意事項

管理者として Windows にログオンしているユーザが Web サービス開発プラグインを実行する場合，権限を管理者に昇格させて起動した Eclipse で実行してください。<Eclipse のインストールディレクトリ>¥eclipse¥eclipse.exe を右クリックして［管理者として実行］を選択する，などの方法で，権限を管理者に昇格させて Eclipse を起動してください。

管理者のパスワードまたは確認を求められた場合は，画面の指示に従って，パスワードを入力してください。

(b) 一般ユーザが実行する場合の注意事項

- インストール時

  Developer を<システムドライブ>:¥Program Files など UAC 保護されたディレクトリ以外にインストールする場合は，インストール後，Web サービスの開発に使用するコマンド（CUI）のログ出力先ディレクトリに，Web サービスを開発するユーザが書き込みできるようにアクセス権を設定してください。コマンド（CUI）については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「14.1.1　cjwsimport コマンド」，および「14.1.2　hwsgen コマンド」を参照してください。ログ出力先ディレクトリを次に示します。

  - 稼働ログ

<Developerのインストールディレクトリ>¥jaxws¥logs

- 例外ログ

  <Developerのインストールディレクトリ>¥jaxws¥logs

- 保守ログ

  <Developerのインストールディレクトリ>¥jaxws¥logs¥maintenance

注

ログの出力先は，<Developerのインストールディレクトリ>¥jaxws¥conf にある共通定義ファイル(cjwconf.properties)の com.cosminexus.jaxws.tool.log.directory キーで設定できます。プロパティの詳細は，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「10.1.2　共通定義ファイルの設定項目」を参照してください。

- 実行時

  一般ユーザが Web サービス開発プラグインの機能を実行する場合，Eclipse のワークスペースは UAC が保護対象とするディレクトリ（%ProgramData%など）以外に設定してください。設定する生成物の出力先も UAC が保護対象とするディレクトリ以外を指定して実行してください。また，Developer を<システムドライブ>:¥Program Files など UAC 保護されたディレクトリにインストールしている場合，UAC によって CUI のログは，%LoadlAppData%¥VirtualStore¥Program Files ディレクトリ以下の対応するディレクトリにリダイレクトされます。

(2)　環境変数

Eclipse の起動時に，設定されている環境変数が引き継がれます。

(3)　JAX-WS エンジンを利用するための設定（V9 互換モードで J2EE サーバを使用する場合）

Web サービスの開発では JAX-WS エンジンを利用します。V9 互換モードで J2EE サーバを使用する場合，JAX-WS エンジンを利用するには，次に示す設定を J2EE サーバに追加して，JAX-WS エンジンを有効にする必要があります。

add.class.path=<Developerのインストールディレクトリ>¥jaxws¥lib¥cjjaxws.jar

(a)　サーバ管理コマンド（CUI）を使ってサーバを運用する場合

サーバ管理コマンド（CUI）を使ってサーバを運用する場合，次の J2EE サーバ用オプション定義ファイルをテキストエディタで開いて設定を追加します。

<Developerのインストールディレクトリ>¥CC¥server¥usrconf¥ejb¥<J2EEサーバ名>¥usrconf.cfg

(b)　Management Server の運用管理ポータルを利用する場合

Management Server の運用管理ポータルを利用する場合，[J2EE コンテナの設定］画面の［拡張パラメタ］で設定します。[J2EE コンテナの設定］画面の［拡張パラメタ］の設定については，マニュアル「ア

プリケーションサーバ 運用管理ポータル操作ガイド」の「10.8.2　J2EE コンテナの設定」を参照してください。

(c)　Smart Composer 機能を利用する場合

Smart Composer 機能を利用する場合は，簡易構築定義ファイルに，J2EE の拡張パラメタとして追加します。Smart Composer 機能については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「1.1.3　Smart Composer 機能とは」を参照してください。簡易構築定義ファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

8.1.2　Web サービス開発時の注意事項

Web サービス開発時の注意事項を次に示します。

(1)　操作

Eclipse や Web サービス開発プラグインが誤動作する場合があるため，Eclipse のワークスペースの下のリソース（フォルダおよびファイル）に対して，エクスプローラなどで直接操作しないでください。

(2)　複数の Web サービスの開発

以降で説明する開発の流れは，一つのプロジェクトに一つの Web サービスを実装する場合を前提に説明しています。一つのプロジェクトに複数の Web サービスが実装される場合，または一つの EAR ファイルに複数の Web サービスのプロジェクトが実装される場合などは，以降で説明する手順を必要な回数だけ実行してください。

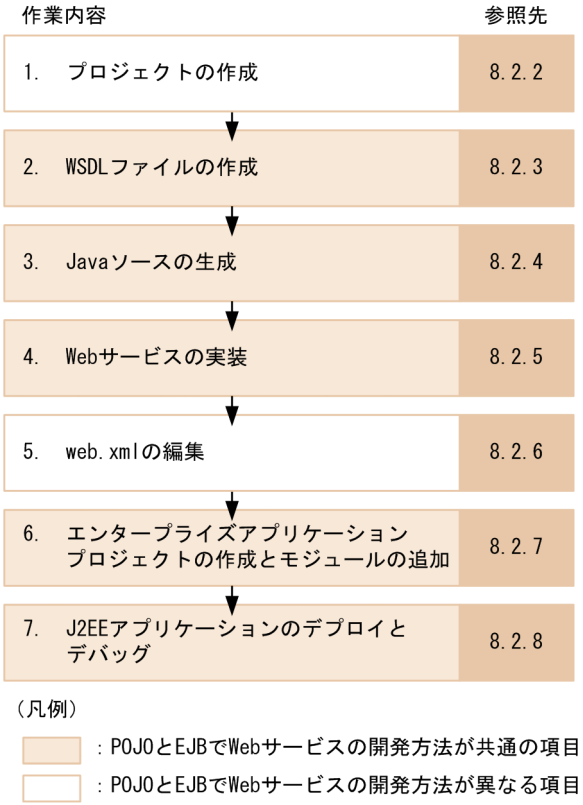
8.2　WSDL を起点とした Web サービスの開発

WSDL を起点とした Web サービスの開発方法について説明します。

8.2.1　WSDL を起点とした Web サービスの開発の流れ

Eclipse を利用して，WSDL を起点とした，POJO の Web サービスおよび EJB の Web サービスを開発できます。POJO の Web サービスおよび EJB の Web サービスの開発の流れは基本的に同じですが，一部異なる個所があります。WSDL を起点とした Web サービスの開発の流れを，次の図に示します。

図 8-1　WSDL を起点とした Web サービスの開発の流れ



それぞれの作業の概要を次に示します。

1. プロジェクトの作成

   POJO の Web サービスを開発する場合，Web サービスの開発に使用する動的 Web プロジェクトを作成します。詳細は，「8.2.2　プロジェクトの作成」を参照してください。

   EJB の Web サービスを開発する場合，EJB の Web サービスは EJB-JAR ファイルに含まれるため，EJB プロジェクトを作成します。詳細は，「4.4.2　EJB プロジェクトの作成」を参照してください。

2. WSDL ファイルの作成

   WSDL ファイルを作成します。または，公開されている WSDL ファイルの URL を取得します。詳細は，「8.2.3　WSDL ファイルの作成」を参照してください。

3. Java ソースの生成

Web サービスの開発に必要な Java ソースを生成します。詳細は，「8.2.4　Java ソースの生成」を参照してください。

4. Web サービスの実装

Web サービス実装クラスのスケルトンに Web サービスを実装します。詳細は，「8.2.5　Web サービスの実装」を参照してください。

5. web.xml の編集

POJO の Web サービスを開発する場合，Web サービス用の設定を追加します。詳細は，「8.2.6 web.xml の編集」を参照してください。

EJB の Web サービスを開発する場合，EJB の Web サービスは EJB-JAR ファイルに含まれるため，web.xml の編集は不要です。

6. エンタープライズアプリケーションプロジェクトの作成とモジュールの追加

エンタープライズアプリケーションプロジェクトを作成して，動的 Web プロジェクトまたは EJB プロジェクトを追加します。詳細は，「8.2.7　エンタープライズアプリケーションプロジェクトの作成とモジュールの追加」を参照してください。

7. J2EE アプリケーションのデプロイとデバッグ

J2EE アプリケーションを J2EE サーバにデプロイして，デバッグします。詳細は，「8.2.8　J2EE アプリケーションのデプロイとデバッグ」を参照してください。

以降の項では，この流れに沿って WSDL を起点とした Web サービスの開発の手順を説明します。

8.2.2　プロジェクトの作成

POJO の Web サービスを開発する場合，WSDL ファイルを起点とした Web サービスの開発に使用する動的 Web プロジェクトを作成します。動的 Web プロジェクトの作成方法については，「4.4.1　動的 Web プロジェクトの作成」を参照してください。

EJB の Web サービスを開発する場合，EJB の Web サービスは EJB-JAR ファイルに含まれるため，EJB プロジェクトを作成します。詳細は，「4.4.2　EJB プロジェクトの作成」を参照してください。

注意事項

Web サービスの開発に使用するプロジェクトを作成する場合，次の点に注意してください。

- **Web サービスを開発するためのプロジェクトを Eclipse で作成する場合**

  次に示す値を設定してください。次に示した値以外を設定すると，Web サービスを実行できないことがあります。

  動的 Web プロジェクト作成時

［動的 web モジュールバージョン］を［2.5］または「3.0」に設定。

EJB プロジェクト作成時

［EJB モジュールバージョン］を［3.0］または「3.1」に設定。

EAR プロジェクト作成時

［EAR バージョン］を［5.0］または「6.0」に設定。

8.2.3　WSDL ファイルの作成

Web サービスのメタ情報を記載した WSDL ファイルを作成します。WSDL ファイルがすでに作成されている場合は，その WSDL ファイルを入手します。なお，WSDL ファイルが公開されている場合は，その WSDL ファイルの URL を取得します。WSDL の作成については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「3.1　WSDL の作成」を参照してください。

8.2.4　Java ソースの生成

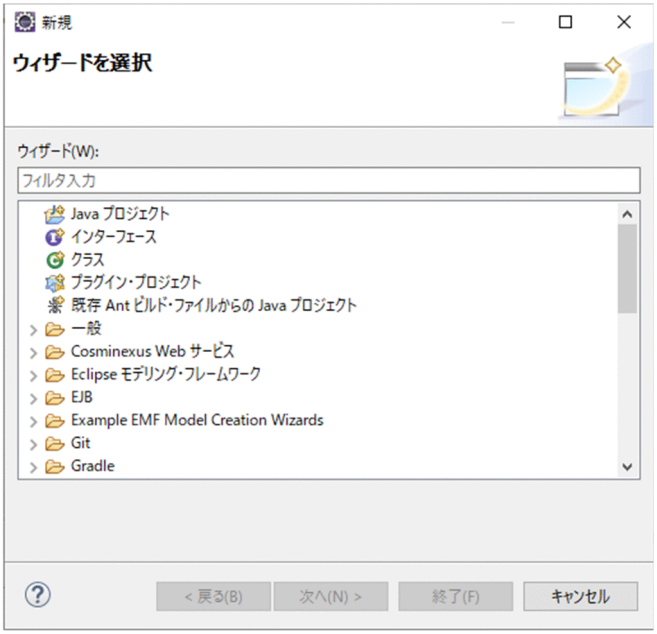
作成した WSDL ファイルを基に，Web サービスの開発や実行に必要な Java ソース（Web サービス実装クラスのスケルトン，SEI，および Java Bean（スタブ））を生成します。Java ソースは，Eclipse から生成できます。Java ソースを生成する手順を次に示します。

1. **［プロジェクト・エクスプローラー］ビューで，Web サービスの開発に使用するプロジェクトを選択します。**

   POJO の Web サービスを開発する場合は動的 Web プロジェクトを，EJB の Web サービスを開発する場合は EJB プロジェクトを選択します。

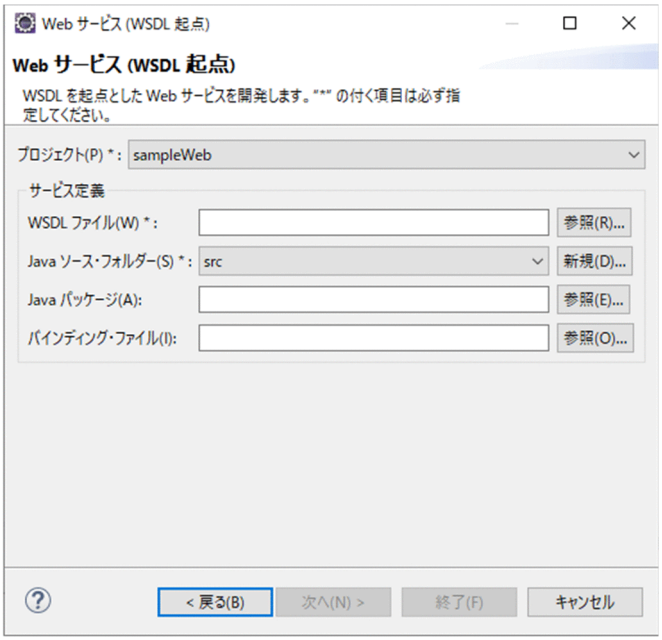
2. **Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。**

   ［新規］ダイアログが表示されます。



3. [Cosminexus Web サービス] − [Web サービス (WSDL 起点)] を選択して，[次へ] ボタンをクリックします。

   [Web サービス (WSDL 起点)] ダイアログの [Web サービス (WSDL 起点)] ページが表示されます。



注意事項

[プロジェクト・エクスプローラー] ビューなどの選択状態によっては，[Web サービス (WSDL 起点)] ダイアログを表示した場合に，[プロジェクト] が空の状態に関わらず，[終了] ボタンが活性化していることがあります。[終了] ボタンをクリックすると，Eclipse の API の挙動により，エラーダイアログとエラーログに InvocationTargetException，および NullPointerException が出力されることがあります。この場合，[プロジェクト]，および，[Java ソース・フォルダー] を指定すると，処理を続行できます。

4. 次の項目を指定します。

項目名	指定値
プロジェクト	ワークスペースの動的 Web プロジェクト名または EJB プロジェクト名を選択します。
WSDL ファイル	WSDL ファイル，または WSDL ファイルの URL を指定します。
Java ソース・フォルダー	Java ソースを生成するフォルダを指定します。

次に示す設定をした場合，［Web サービス (WSDL 起点)］ダイアログを表示すると，［Java ソース・フォルダー］が空欄となり，以降の操作ができなくなります。そのため，［Java ソース・フォルダー］には，プロジェクトルート以外を設定してください。

- Web サービスの開発に使用するプロジェクトの［プロパティ］ダイアログの［Java のビルド・パス］ページで，［ソース］タブの［ビルド・パス上のソース・フォルダ］にプロジェクトルートを設定した場合

また，［プロジェクト］および［Java ソース・フォルダー］に入力した値を変更すると，次に示す項目の値が初期状態に戻ります。

- ［プロジェクト］の値を変更した場合

  ［WSDL ファイル］，［Java ソース・フォルダー］，［Java パッケージ］，および［バインディング・ファイル］の値が初期状態に戻ります。

- ［Java ソース・フォルダー］の値を変更した場合

  ［WSDL ファイル］，［Java パッケージ］，および［バインディング・ファイル］の値が初期状態に戻ります。

なお，必要に応じて，次の項目を指定してください。

項目名	指定値
Java パッケージ	生成する Java ソースのパッケージ名を指定します。
バインディング・ファイル	バインディングファイルを指定します。

各項目の詳細については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「14.1.1　cjwsimport コマンド」も合わせて参照してください。

5. ［終了］ボタンをクリックします。

処理が実行されたことを示すメッセージが表示され，Java ソースが生成されます。

注意事項

Web サービスのバージョンアップなどの理由で，WSDL ファイルやバインディングファイルを修正して，［Web サービス (WSDL 起点)］ダイアログで Java ソースを再生成する場合，必ず［Java ソース・フォルダー］に指定するフォルダに出力ファイルと同名のファイルがないか確認してください。

出力ファイルと同名のファイルがある場合，上書きされたり，エラーになったりします。エラーが発生してファイルが出力された場合は，［Web サービス (WSDL 起点)］ダイアログで Java ソースを再生成する前にも同様に［Java ソース・フォルダー］に指定するフォルダに出力ファイルと同名のファイルがないか確認してください。

- 指定するフォルダに出力ファイルと同名の SEI または Java Bean（スタブ）がある場合，上書きされます。
- 指定するフォルダに出力ファイルと同名の Web サービス実装クラスのスケルトンがある場合，［コンソール］ビューに警告を示すメッセージが表示され，Web サービス実装クラスのスケルトンは生成されません。そのため，プロジェクト内の Web サービス実装クラスと，SEI および Java Bean との整合性が取れなくなり，Web サービスを正しく実行できないおそれがあります。Web サービス実装クラスと，SEI および Java Bean の整合性を取るためには，次に示すどちらかの処理をする必要があります。

  ・ファイルを出力するフォルダを空にして，［Web サービス (WSDL 起点)］ダイアログを再度実行し，Web サービス実装クラスのスケルトンを生成する。

  ・旧バージョンの Web サービス実装クラスをバックアップ後，ファイルを出力するフォルダを空にして，［Web サービス (WSDL 起点)］ダイアログを再度実行し，新バージョンの Web サービス実装クラスのスケルトンを生成する。

  生成された新バージョンの Web サービス実装クラスのスケルトンの実装に，旧バージョンの Web サービス実装クラスの実装内容が利用できる場合は，新バージョンの Web サービス実装クラスのスケルトンに，バックアップした旧バージョンの Web サービス実装クラスの実装部分，および新バージョンの実装を追加する。

8.2.5　Web サービスの実装

生成されたスタブを利用し，Web サービス実装クラスのスケルトンに必要な処理を記述して，Web サービスを実装します。また，Eclipse で［プロジェクト］－［自動的にビルド］を設定している場合，Web サービス実装クラスの Java ソースファイルを保存したときに，Eclipse によって自動的にコンパイルされ，クラスファイル（*.class）が作成されます。EJB の Web サービスの場合，Web サービス実装クラスに必ず@javax.ejb.Stateless を付与してください。Web サービスの実装方法については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「4.3.3　Web サービス実装クラスを作成する」を参照してください。

8.2.6　web.xml の編集

POJO の Web サービスを開発する場合，Web サービスをデプロイするために，web.xml に Web サービスとして動作するための設定を追加します。EJB の Web サービスを開発する場合，EJB の Web サービスは EJB-JAR ファイルに含まれるため，web.xml の編集は不要です。

追加する要素と指定する値を次の表に示します。

表 8-1　web.xml で追加する要素と指定する値

web.xml の要素名	指定する値
web-app	−
├ listener	−
│　└ listener-class │	com.cosminexus.xml.ws.transport.http.servlet.WSServletContextListener
├ servlet	−
│　├ servlet-name	WSServlet
│　├ servlet-class │　│	com.cosminexus.xml.ws.transport.http.servlet.WSServlet
│　└ load-on-startup	1
└ servlet-mapping	−
├ servlet-name	WSServlet
└ url-pattern	マッピングする URL※

（凡例）　−：指定する値はありません。

注※

「"/" + WSDL の wsdl:service 要素の name 属性値」を指定してください（「"/" + WSDL の wsdl:service 要素の name 属性値」は，WSDL の wsdl:service 要素の name 属性値の先頭に「/」（スラッシュ）が付くことを示します）。

指定例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
          http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <listener>
        <listener-class>com.cosminexus.xml.ws.transport.http.servlet.WSServletContextListene
r</listener-class>
    </listener>

    <servlet>
        <servlet-name>WSServlet</servlet-name>
        <servlet-class>com.cosminexus.xml.ws.transport.http.servlet.WSServlet</servlet-class
>
        <load-on-startup>1</load-on-startup>
    </servlet>
```

```
    <servlet-mapping>
        <servlet-name>WSServlet</servlet-name>
        <url-pattern>/TestJaxWsService</url-pattern>
    </servlet-mapping>

</web-app>
```

web.xml の編集の詳細は，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「3.4 web.xml の作成」を参照してください。

8.2.7 エンタープライズアプリケーションプロジェクトの作成とモジュールの追加

(1) エンタープライズアプリケーションプロジェクトの作成

エンタープライズアプリケーションプロジェクトを作成します。エンタープライズアプリケーションプロジェクトの作成手順については，「4.4.4 エンタープライズアプリケーションプロジェクトの作成」を参照してください。

(2) エンタープライズアプリケーションプロジェクトへのモジュールの追加

エンタープライズアプリケーションプロジェクトに，「8.2.2 プロジェクトの作成」で作成した WSDL を起点とした Web サービス用の動的 Web プロジェクトまたは EJB プロジェクトを追加します。動的 Web プロジェクトまたは EJB プロジェクトの追加手順を次に示します。

1. [New EAR Application Project] ダイアログの［EAR Application Project］ページの［次へ］ボタンをクリックします。

   [Enterprise Application] ページが表示されます。

2. 追加するモジュールをチェックして，［終了］ボタンをクリックします。

8.2.8 J2EE アプリケーションのデプロイとデバッグ

作成したプロジェクトをデプロイしてデバッグします。

(1) J2EE アプリケーションのデプロイ

エンタープライズアプリケーションプロジェクトから作成された J2EE アプリケーションを J2EE サーバにデプロイします。デプロイ方法については，「6.6 J2EE サーバへのプロジェクトの公開」を参照してください。

(2)　J2EE アプリケーションのデバッグ

作成した J2EE アプリケーションをデバッグします。デバッグ方法については，「6.7　J2EE アプリケーションのデバッグと実行」を参照してください。

8.3　SEI を起点とした Web サービスの開発

SEI を起点とした Web サービスの開発方法について説明します。

8.3.1　SEI を起点とした Web サービスの開発の流れ

Eclipse を利用して，SEI を起点とした，POJO の Web サービスおよび EJB の Web サービスを開発できます。POJO の Web サービスおよび EJB の Web サービスの開発の流れは基本的に同じですが，一部異なる個所があります。SEI を起点とした Web サービスの開発の流れを，次の図に示します。

図 8-2　SEI を起点とした Web サービスの開発の流れ

それぞれの作業の概要を次に示します。

1. プロジェクトの作成

   POJO の Web サービスを開発する場合，Web サービスの開発に使用する動的 Web プロジェクトを作成します。詳細は，「8.3.2　プロジェクトの作成」を参照してください。

   EJB の Web サービスを開発する場合，EJB の Web サービスは EJB-JAR ファイルに含まれるため，EJB プロジェクトを作成します。詳細は，「4.4.2　EJB プロジェクトの作成」を参照してください。

2. Web サービス実装クラスの作成

   Web サービスの処理を記述した Web サービス実装クラスを作成します。詳細は，「8.3.3　Web サービス実装クラスの作成」を参照してください。

3. Java ソース・WSDL・XSD の生成

Web サービスの開発に必要な Java ソース，WSDL，および XSD を生成します。詳細は，「8.3.4 Java ソース・WSDL・XSD の生成」を参照してください。

4. web.xml の編集

   POJO の Web サービスを開発する場合，Web サービス用の設定を追加します。詳細は，「8.3.5 web.xml の編集」を参照してください。

   EJB の Web サービスを開発する場合，EJB の Web サービスは EJB-JAR ファイルに含まれるため，web.xml の編集は不要です。

5. エンタープライズアプリケーションプロジェクトの作成とモジュールの追加

   エンタープライズアプリケーションプロジェクトを作成して，動的 Web プロジェクトまたは EJB プロジェクトを追加します。詳細は，「8.3.6　エンタープライズアプリケーションプロジェクトの作成とモジュールの追加」を参照してください。

6. J2EE アプリケーションのデプロイとデバッグ

   J2EE アプリケーションを J2EE サーバにデプロイして，デバッグします。詳細は，「8.3.7　J2EE アプリケーションのデプロイとデバッグ」を参照してください。

以降の項では，この流れに沿って SEI を起点とした Web サービスの開発の手順を説明します。

8.3.2　プロジェクトの作成

POJO の Web サービスを開発する場合，SEI を起点とした Web サービスの開発に使用する動的 Web プロジェクトを作成します。動的 Web プロジェクトの作成方法については，「4.4.1　動的 Web プロジェクトの作成」を参照してください。

EJB の Web サービスを開発する場合，EJB の Web サービスは EJB-JAR ファイルに含まれるため，EJB プロジェクトを作成します。詳細は，「4.4.2　EJB プロジェクトの作成」を参照してください。

注意事項

Web サービスの開発に使用するプロジェクトを作成する場合，次の点に注意してください。

- Web サービスを開発するためのプロジェクトを Eclipse で作成する場合

  次に示す値を設定してください。次に示した値以外を設定すると，Web サービスを実行できないことがあります。

  動的 Web プロジェクト作成時

  ［動的 web モジュールバージョン］を［2.5］または「3.0」に設定。

  EJB プロジェクト作成時

  ［EJB モジュールバージョン］を［3.0］または「3.1」に設定。

  EAR プロジェクト作成時

［EAR バージョン］を［5.0］または「6.0」に設定。

8.3.3　Web サービス実装クラスの作成

Web サービスの処理を記述した Web サービス実装クラスの Java ソースファイルを作成します。Eclipse で［プロジェクト］－［自動的にビルド］を設定している場合，Web サービス実装クラスの Java ソースファイルを保存したときに，Eclipse によって自動的にコンパイルされ，クラスファイル（*.class）が作成されます。EJB の Web サービスの場合，Web サービス実装クラスに必ず@javax.ejb.Stateless を付与してください。Web サービス実装クラスの作成例については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「5.3.1　Web サービス実装クラスを作成する」を参照してください。

8.3.4　Java ソース・WSDL・XSD の生成

作成した Web サービス実装クラスを基に，Web サービスの開発や実行に必要な Java ソース（Java Bean（スタブ）），WSDL ファイル，および XSD ファイルを生成します。Java ソース，WSDL ファイル，および XSD ファイルは，Eclipse から生成できます。

参考

hwsgen コマンドを実行する場合，Java ソース，WSDL ファイル，および XSD ファイルは生成不要です。Java ソース，WSDL ファイル，および XSD ファイルは，必要に応じて任意で生成してください。

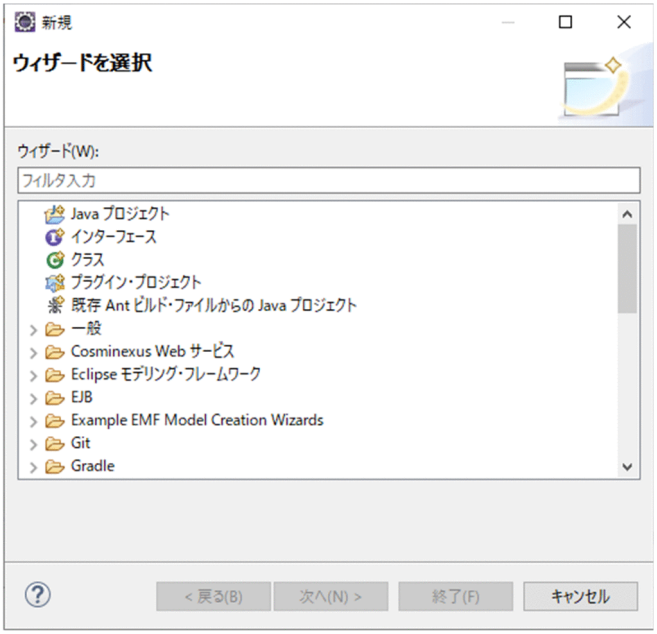
Java ソース，WSDL ファイル，および XSD ファイルを生成する手順を次に示します。

1. **［プロジェクト・エクスプローラー］ビューで，Web サービスの開発に使用するプロジェクトを選択します。**

   POJO の Web サービスを開発する場合は動的 Web プロジェクトを，EJB の Web サービスを開発する場合は EJB プロジェクトを選択します。

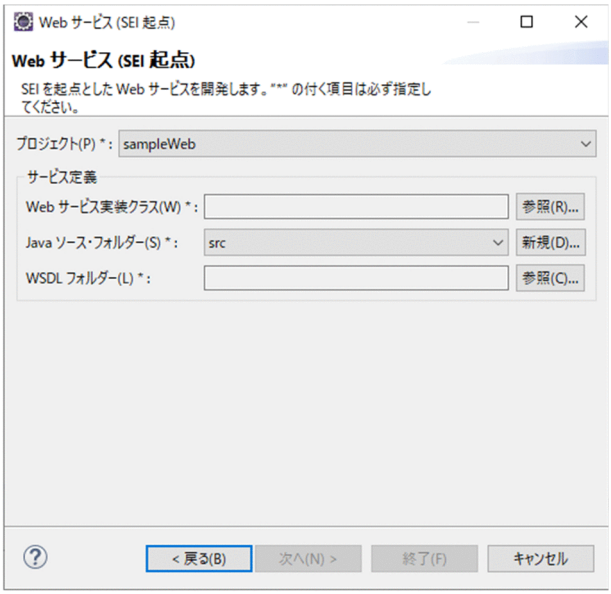
2. **Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。**

   ［新規］ダイアログが表示されます。



3. [Cosminexus Web サービス] − [Web サービス (SEI 起点)] を選択して，[次へ] ボタンをクリックします。

   [Web サービス (SEI 起点)] ダイアログの [Web サービス (SEI 起点)] ページが表示されます。



4. 次の項目を指定します。

項目名	指定値
プロジェクト	ワークスペースの動的 Web プロジェクト名または EJB プロジェクト名を選択します。
Web サービス実装クラス	Web サービス実装クラスを指定します。
Java ソース・フォルダー	Java ソースを生成するフォルダを指定します。
WSDL フォルダー※	WSDL ファイルを生成するフォルダを指定します。

注※ ［WSDL フォルダー］の［参照］ボタンをクリックして［フォルダの参照］ダイアログを表示し，ツリービューで［マイコンピュータ］などの仮想フォルダを選択した状態で，フォルダ名を［フォルダ］に入力して選択することはできません。

次に示す設定をした場合，［Web サービス (SEI 起点)］ダイアログを表示すると，［Java ソース・フォルダー］が空欄となり，以降の操作ができなくなります。そのため，［Java ソース・フォルダー］には，プロジェクトルート以外を設定してください。

- Web サービスの開発に使用するプロジェクトの［プロパティ］ダイアログの［Java のビルド・パス］ページで，［ソース］タブの［ビルド・パス上のソース・フォルダ］にプロジェクトルートを設定した場合

また，［プロジェクト］および［Java ソース・フォルダー］に入力した値を変更すると，次に示す項目の値が初期状態に戻ります。

- ［プロジェクト］の値を変更した場合

  ［Web サービス実装クラス］，［Java ソース・フォルダー］，および［WSDL フォルダー］の値が初期状態に戻ります。

- ［Java ソース・フォルダー］の値を変更した場合

  ［Web サービス実装クラス］，および［WSDL フォルダー］の値が初期状態に戻ります。

各項目の詳細については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「14.1.2　hwsgen コマンド」も合わせて参照してください。

5. ［終了］ボタンをクリックします。

処理が実行されたことを示すメッセージが表示され，Java ソース，WSDL ファイル，および XSD ファイルが生成されます。

注意事項

Web サービスのバージョンアップなどの理由で，Web サービス実装クラスを修正して，［Web サービス (SEI 起点)］ダイアログで Java ソースを再生成する場合，必ず［Java ソース・フォルダー］に指定するフォルダに出力ファイルと同名のファイルがないか確認してください。

また，WSDL ファイルを生成する場合も，同様に［WSDL フォルダー］に指定するフォルダに出力ファイルと同名のファイルがないか確認してください。

出力ファイルと同名の Java Bean（スタブ），WSDL，および XSD がある場合，上書きされます。

8.3.5　web.xml の編集

POJO の Web サービスを開発する場合，Web サービスをデプロイするために，web.xml に Web サービスとして動作するための設定を追加します。EJB の Web サービスを開発する場合，web.xml の編集は不要です。

追加する要素と指定する値を次の表に示します。

表 8-2　web.xml で追加する要素と指定する値

web.xml の要素名	指定する値
web-app	−
├ listener	−
│　└ listener-class │	com.cosminexus.xml.ws.transport.http.servlet.WSServletContextListener
├ servlet	−
│　├ servlet-name	WSServlet
│　├ servlet-class │　│	com.cosminexus.xml.ws.transport.http.servlet.WSServlet
│　└ load-on-startup	1
└ servlet-mapping	−
├ servlet-name	WSServlet
└ url-pattern	マッピングする URL※

（凡例）　−：指定する値はありません。

注※

Web サービス実装クラスの WebService アノテーションで serviceName 属性を記述しているかどうかによって，指定する値が異なります。

- serviceName 属性を記述している場合

  「"/" + serviceName 属性値」を指定してください（「"/" + serviceName 属性値」は，serviceName 属性値の先頭に「/」（スラッシュ）が付くことを示します）。

- serviceName 属性を記述していない場合

  「"/" + Web サービス実装クラスの単純名 + Service」を指定してください（「"/" + Web サービス実装クラスの単純名 + Service」は，Web サービス実装クラスの単純名の先頭に「/」(スラッシュ)，末尾に「Service」が付くことを示します）。

指定例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <listener>
        <listener-class>com.cosminexus.xml.ws.transport.http.servlet.WSServletContextListener</listener-class>
    </listener>
```

```
    <servlet>
        <servlet-name>WSServlet</servlet-name>
        <servlet-class>com.cosminexus.xml.ws.transport.http.servlet.WSServlet</servlet-class
>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>WSServlet</servlet-name>
        <url-pattern>/TestJaxWsService</url-pattern>
    </servlet-mapping>

</web-app>
```

web.xml の編集の詳細は，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「3.4 web.xml の作成」を参照してください。

8.3.6 エンタープライズアプリケーションプロジェクトの作成とモジュールの追加

(1) エンタープライズアプリケーションプロジェクトの作成

エンタープライズアプリケーションプロジェクトを作成します。エンタープライズアプリケーションプロジェクトの作成手順については，「4.4.4 エンタープライズアプリケーションプロジェクトの作成」を参照してください。

(2) エンタープライズアプリケーションプロジェクトへのモジュールの追加

エンタープライズアプリケーションプロジェクトに，「8.3.2 プロジェクトの作成」で作成した SEI を起点とした Web サービス用の動的 Web プロジェクトまたは EJB プロジェクトを追加します。動的 Web プロジェクトまたは EJB プロジェクトの追加手順を次に示します。

1. [New EAR Application Project] ダイアログの［EAR Application Project］ページの［次へ］ボタンをクリックします。

   ［Enterprise Application］ページが表示されます。

2. 追加するモジュールをチェックして，［終了］ボタンをクリックします。

8.3.7 J2EE アプリケーションのデプロイとデバッグ

作成したプロジェクトをデプロイしてデバッグします。

(1)　J2EE アプリケーションのデプロイ

エンタープライズアプリケーションプロジェクトから作成された J2EE アプリケーションを J2EE サーバにデプロイします。デプロイ方法については，「6.6　J2EE サーバへのプロジェクトの公開」を参照してください。

(2)　J2EE アプリケーションのデバッグ

作成した J2EE アプリケーションをデバッグします。デバッグ方法については，「6.7　J2EE アプリケーションのデバッグと実行」を参照してください。

8.4　Web サービスクライアントの開発

Web サービスクライアントを開発するには，次に示す方法があります。

- Java アプリケーションを使用する方法
- Web アプリケーションを使用する方法
- EJB を使用する方法
- 既存の Web サービスを使用する方法

POJO の Web サービスにアクセスするための Web サービスクライアントの開発方法と，EJB の Web サービスにアクセスするための Web サービスクライアントの開発方法は同じです。

以降の項では，これらの Web サービスクライアントの開発方法について説明します。

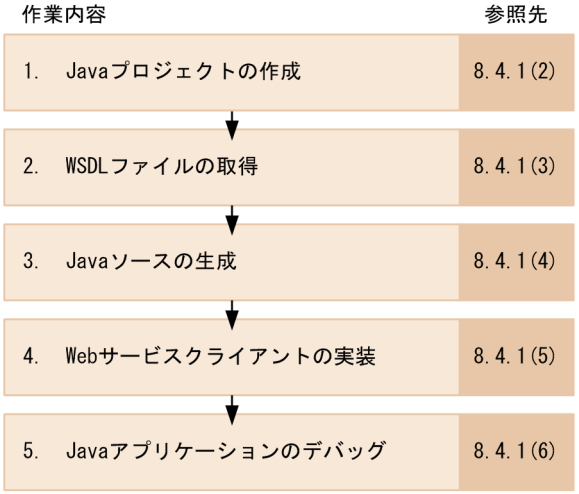
8.4.1　Java アプリケーションを使用した Web サービスクライアントの開発

Java アプリケーションを使用した Web サービスクライアントの開発方法について説明します。

(1)　Java アプリケーションを使用した Web サービスクライアントの開発の流れ

Java アプリケーションを使用した Web サービスクライアントの開発の流れを，次の図に示します。

図 8-3　Java アプリケーションを使用した Web サービスクライアントの開発の流れ



それぞれの作業の概要を次に示します。

1. Java プロジェクトの作成

   Web サービスクライアントの開発に使用する Java プロジェクトを作成します。詳細は，「8.4.1(2) Java プロジェクトの作成」を参照してください。

2. WSDL ファイルの取得

WSDL ファイルを取得します。または，公開されている WSDL ファイルの URL を取得します。詳細は，「8.4.1(3)　WSDL ファイルの取得」を参照してください。

3. Java ソースの生成

Web サービスクライアントを実装するための Java ソースを生成します。詳細は，「8.4.1(4)　Java ソースの生成」を参照してください。

4. Web サービスクライアントの実装

生成された Java ソースを利用して，Web サービスクライアントを実装します。詳細は，「8.4.1(5)　Web サービスクライアントの実装」を参照してください。

5. Java アプリケーションのデバッグ

Java アプリケーションをデバッグします。詳細は，「8.4.1(6)　Java アプリケーションのデバッグ」を参照してください。

以降，この流れに沿って Java アプリケーションを使用した Web サービスクライアントの開発の手順を説明します。

(2)　Java プロジェクトの作成

Java アプリケーションを使用した Web サービスクライアントの開発に使用する Java プロジェクトを作成します。Java プロジェクトを作成する手順を次に示します。

1. Eclipse のメニューから［ファイル］－［新規］－［プロジェクト］を選択します。

［新規プロジェクト］ダイアログが表示されます。

2. ［Java プロジェクト］を選択して，［次へ］ボタンをクリックします。

［Java プロジェクトの作成］ページが表示されます。

3. 必要な項目を指定して，［終了］ボタンをクリックします。

Java プロジェクトが作成されます。

なお，JAX-WS エンジンを使用し，Java アプリケーションで Web サービスを実行するためには，プロジェクトのビルドパスに外部 Jar を設定したり，Java アプリケーションの実行構成，またはデバッグ構成に VM 引数を設定したりする必要があります。それぞれの設定手順を次に示します。

- プロジェクトのビルドパスへの外部 Jar の設定

1. Java プロジェクトを選択して，右クリックのコンテキストメニューから［ビルド・パス］－［ビルド・パスの構成］を選択します。
2. ［プロパティ］ダイアログの［Java のビルド・パス］ページの［ライブラリー］タブを選択します。
3. ［外部 Jar の追加］ボタンをクリックします。
4. 外部 Jar を追加します。

J2EE サーバの J2EE サーバモードによって，設定する外部 Jar が異なります。

推奨モードの場合

<Developerのインストールディレクトリ>¥CC¥javaee¥1100¥lib¥javaee-api.jar

<Developerのインストールディレクトリ>¥CC¥client¥lib¥HiEJBClientStatic.jar

<Developerのインストールディレクトリ>¥jdk¥lib¥hcompatlib¥hjdk.tpb.jar

<Developerのインストールディレクトリ>¥jdk¥lib¥hcompatlib¥hjdk.act.jar

<HNTRLib2インストールディレクトリ>※¥classes¥hntrlib2j64.jar

<HNTRLib2インストールディレクトリ>※¥classes¥hntrlibMj64.jar

V9 互換モードの場合

<Developerのインストールディレクトリ>¥jaxp¥lib¥csmjaxb.jar

<Developerのインストールディレクトリ>¥jaxp¥lib¥csmjaxp.jar

<Developerのインストールディレクトリ>¥jaxws¥lib¥cjjaxws.jar

<Developerのインストールディレクトリ>¥CC¥client¥lib¥j2ee-javax.jar

<Developerのインストールディレクトリ>¥CC¥client¥lib¥HiEJBClientStatic.jar

<Developerのインストールディレクトリ>¥jdk¥lib¥hcompatlib¥hjdk.tpb.jar

<Developerのインストールディレクトリ>¥jdk¥lib¥hcompatlib¥hjdk.act.jar

<HNTRLib2インストールディレクトリ>※¥classes¥hntrlib2j64.jar

<HNTRLib2インストールディレクトリ>※¥classes¥hntrlibMj64.jar

注※ <HNTRLib2インストールディレクトリ>の部分は環境変数COSMINEXUS_HNTRLIB_HOMEの値を指定します。

- Java アプリケーションの実行構成またはデバッグ構成の VM 引数の設定

  1. Eclipse のメニューから［実行］－［実行構成]，または［実行］－［デバッグの構成］を選択します。
  2. [Java アプリケーション］をダブルクリックします。
  3. [メイン］タブの［プロジェクト］に Java プロジェクトの名称を設定します。
  4. [メイン］タブの［メイン・クラス］に Java プロジェクトを実行するメインクラスの完全修飾名を設定します。
  5. [引数］タブを選択して，次に示す VM 引数を設定します。

     -Dcosminexus.home="<Developerのインストールディレクトリ>"

(3) WSDL ファイルの取得

呼び出そうとしている Web サービスのメタ情報が記述された WSDL ファイルを取得します。または WSDL ファイルの URL が公開されている場合はその URL を取得します。

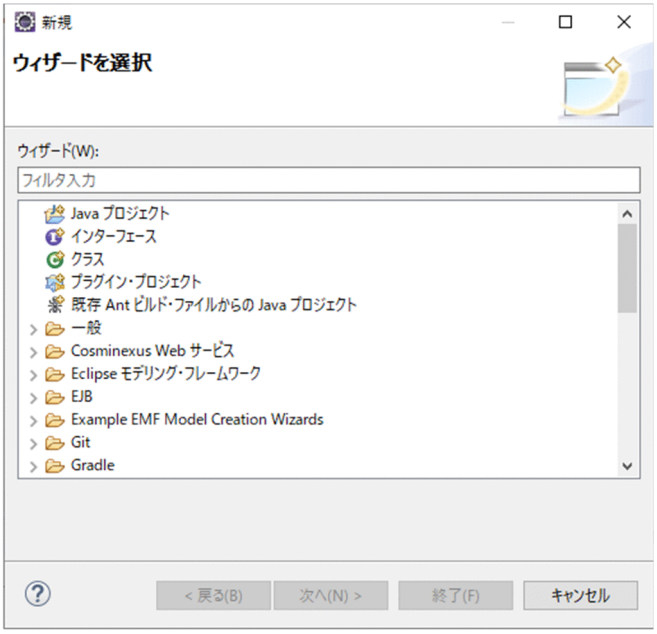
(4) Java ソースの生成

取得した WSDL ファイルを基に，Web サービスクライアントの開発や実行に必要な Java ソース（サービスクラス，SEI，および Java Bean（スタブ））を生成します。Java ソースは，Eclipse を使って生成できます。Java ソースを生成する手順を次に示します。

1. [プロジェクト・エクスプローラー] ビューで，Web サービスクライアントの開発に使用するプロジェクトを選択します。

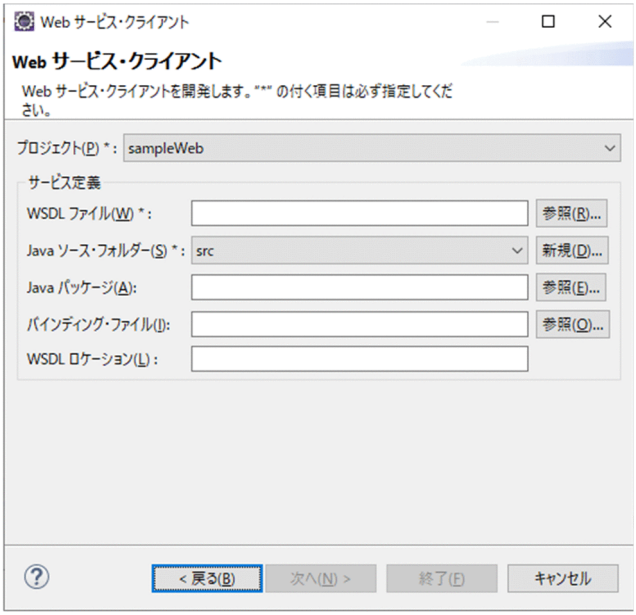
2. Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。

   ［新規］ダイアログが表示されます。



3. [Cosminexus Web サービス] － [Web サービス・クライアント] を選択して，［次へ］ボタンをクリックします。

   [Web サービス・クライアント] ダイアログの［Web サービス・クライアント］ページが表示されます。



4. 次の項目を指定します。

項目名	指定値
プロジェクト	ワークスペースの Java プロジェクト名を選択します。
WSDL ファイル	WSDL ファイル，または WSDL ファイルの URL を指定します。
Java ソース・フォルダー	Java ソースを生成するフォルダを指定します。

次に示す操作や設定をした場合，［Web サービス・クライアント］ダイアログを表示すると，［Java ソース・フォルダー］が空欄となり，以降の操作ができなくなります。そのため，［Java ソース・フォルダー］には，プロジェクトルート以外を設定してください。

- Web サービスクライアント開発に Java プロジェクトを使用する際，Java プロジェクト作成時に［新規 Java プロジェクト］ダイアログの［プロジェクト・レイアウト］に［プロジェクト・フォルダをソースおよびクラス・ファイルのルートとして使用］を選択した場合
- Web サービスクライアントの開発に使用するプロジェクトの［プロパティ］ダイアログの［Java のビルド・パス］ページで，［ソース］タブの［ビルド・パス上のソース・フォルダ］にプロジェクトルートを設定した場合

また，［プロジェクト］および［Java ソース・フォルダー］に入力した値を変更すると，次に示す項目の値が初期状態に戻ります。

- ［プロジェクト］の値を変更した場合

  ［WSDL ファイル］，［Java ソース・フォルダー］，［Java パッケージ］，［バインディング・ファイル］，および［WSDL ロケーション］の値が初期状態に戻ります。
- ［Java ソース・フォルダー］の値を変更した場合

  ［WSDL ファイル］，［Java パッケージ］，［バインディング・ファイル］，および［WSDL ロケーション］の値が初期状態に戻ります。

なお，必要に応じて，次の項目を指定してください。

項目名	指定値
Java パッケージ	生成する Java ソースのパッケージ名を指定します。
バインディング・ファイル	バインディングファイルを指定します。
WSDL ロケーション	javax.xml.ws.WebServiceClient アノテーションの，wsdlLocation 要素に設定する値を指定します。

各項目の詳細については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「14.1.1　cjwsimport コマンド」も合わせて参照してください。

5. [終了] ボタンをクリックします。

処理が実行されたことを示すメッセージが表示され，Java ソースが生成されます。

注意事項

Web サービスのバージョンアップなどの理由で，呼び出される Web サービスの WSDL ファイルやバインディングファイルが修正され，[Web サービス・クライアント] ダイアログで Java ソースを再生成する場合，必ず [Java ソース・フォルダー] に指定するフォルダに出力ファイルと同名のファイルがないか確認してください。

出力ファイルと同名のサービスクラス，SEI または Java Bean（スタブ）がある場合，上書きされます。

(5)　Web サービスクライアントの実装

生成された Java ソースを利用して，Web サービスクライアントを実装します。Web サービスクライアントの実装例については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「3.6 Web サービスクライアントの実装」を参照してください。

(6)　Java アプリケーションのデバッグ

作成した Java アプリケーションをデバッグします。Java アプリケーションのデバッグ構成に VM 引数を設定し，[デバッグ] ボタンをクリックします。Java アプリケーションのデバッグ構成の VM 引数の設定については，「8.4.1(2)　Java プロジェクトの作成」を参照してください。

8.4.2　Web アプリケーションを使用した Web サービスクライアントの開発

Web アプリケーションを使用した Web サービスクライアントの開発方法について説明します。

(1)　Web アプリケーションを使用した Web サービスクライアントの開発の流れ

Web アプリケーションを使用した Web サービスクライアントの開発の流れを，次の図に示します。

図 8-4　Web アプリケーションを使用した Web サービスクライアントの開発の流れ

それぞれの作業の概要を次に示します。

1. 動的 Web プロジェクトの作成

Web サービスクライアントの開発に使用する動的 Web プロジェクトを作成します。詳細は，「8.4.2(2)　動的 Web プロジェクトの作成」を参照してください。

2. WSDL ファイルの取得

WSDL ファイルを取得します。または，公開されている WSDL ファイルの URL を取得します。詳細は，「8.4.2(3)　WSDL ファイルの取得」を参照してください。

3. Java ソースの生成

Web サービスクライアントを実装するための Java ソースを生成します。詳細は，「8.4.2(4)　Java ソースの生成」を参照してください。

4. Web サービスクライアントの実装

生成された Java ソースを利用して，Web サービスクライアントを実装します。詳細は，「8.4.2(5)　Web サービスクライアントの実装」を参照してください。

5. web.xml の編集

Web サービスクライアントを実装した JSP およびサーブレットの定義を追加します。詳細は，「8.4.2(6)　web.xml の編集」を参照してください。

6. エンタープライズアプリケーションプロジェクトの作成とモジュールの追加

エンタープライズアプリケーションプロジェクトを作成して，動的 Web プロジェクトを追加します。詳細は，「8.4.2(7)　エンタープライズアプリケーションプロジェクトの作成とモジュールの追加」を参照してください。

7. J2EE アプリケーションのデプロイとデバッグ

J2EE アプリケーションを J2EE サーバにデプロイして，デバッグします。詳細は，「8.4.2(8)　J2EE アプリケーションのデプロイとデバッグ」を参照してください。

以降，この流れに沿って Web アプリケーションを使用した Web サービスクライアントの開発の手順を説明します。

(2)　動的 Web プロジェクトの作成

Web アプリケーションを使用した Web サービスクライアントの開発に使用する動的 Web プロジェクトを作成します。動的 Web プロジェクトの作成方法については，「4.4.1　動的 Web プロジェクトの作成」を参照してください。

(3)　WSDL ファイルの取得

呼び出そうとしている Web サービスのメタ情報が記述された WSDL ファイルを取得します。または WSDL ファイルの URL が公開されている場合はその URL を取得します。

(4)　Java ソースの生成

取得した WSDL ファイルを基に，Web サービスクライアントの開発や実行に必要な Java ソース（サービスクラス，SEI，および Java Bean（スタブ））を生成します。Java ソースは，Eclipse を使って生成できます。Java ソースを生成する手順は，「8.4.1(4)　Java ソースの生成」と同様ですが，［Web サービス・クライアント］ダイアログの［プロジェクト］には，ワークスペースの動的 Web プロジェクト名を選択します。

また，次に示す操作や設定をした場合，［Web サービス・クライアント］ダイアログを表示すると，［Java ソース・フォルダー］が空欄となり，以降の操作ができなくなります。そのため，［Java ソース・フォルダー］には，プロジェクトルート以外を設定してください。

- Web サービスクライアントの開発に使用するプロジェクトの［プロパティ］ダイアログの［Java のビルド・パス］ページで，［ソース］タブの［ビルド・パス上のソース・フォルダ］にプロジェクトルートを設定した場合

(5)　Web サービスクライアントの実装

生成された Java ソースを利用して，Web サービスクライアントを実装します。Web サービスクライアントの実装例については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「3.6 Web サービスクライアントの実装」を参照してください。

(6)　web.xml の編集

Web サービスクライアントを実装した JSP およびサーブレットの定義を web.xml に追加します。

(7)　エンタープライズアプリケーションプロジェクトの作成とモジュールの追加

(a)　エンタープライズアプリケーションプロジェクトの作成

エンタープライズアプリケーションプロジェクトを作成します。エンタープライズアプリケーションプロジェクトの作成手順については，「4.4.4　エンタープライズアプリケーションプロジェクトの作成」を参照してください。

(b)　エンタープライズアプリケーションプロジェクトへのモジュールの追加

エンタープライズアプリケーションプロジェクトに，「8.4.2(2)　動的 Web プロジェクトの作成」で作成した Web サービスクライアント用の動的 Web プロジェクトを追加します。動的 Web プロジェクトの追加手順を次に示します。

1. [New EAR Application Project] ダイアログの［EAR Application Project］ページの［次へ］ボタンをクリックします。

   ［Enterprise Application］ページが表示されます。

2. 追加するモジュールをチェックして，［終了］ボタンをクリックします。

(8)　J2EE アプリケーションのデプロイとデバッグ

作成したプロジェクトをデプロイしてデバッグします。

(a)　J2EE アプリケーションのデプロイ

エンタープライズアプリケーションプロジェクトから作成された J2EE アプリケーションを J2EE サーバにデプロイします。デプロイ方法については，「6.6　J2EE サーバへのプロジェクトの公開」を参照してください。

(b)　J2EE アプリケーションのデバッグ

作成した J2EE アプリケーションをデバッグします。デバッグ方法については，「6.7　J2EE アプリケーションのデバッグと実行」を参照してください。

8.4.3　EJB を使用した Web サービスクライアントの開発

EJB を使用した Web サービスクライアントの開発方法について説明します。

(1)　EJB を使用した Web サービスクライアントの開発の流れ

EJB を使用した Web サービスクライアントの開発の流れを，次の図に示します。

図 8-5　EJB を使用した Web サービスクライアントの開発の流れ

それぞれの作業の概要を次に示します。

1. EJB プロジェクトの作成

Web サービスクライアントの開発に使用する EJB プロジェクトを作成します。詳細は，「8.4.3(2) EJB プロジェクトの作成」を参照してください。

2. WSDL ファイルの取得

WSDL ファイルを取得します。または，公開されている WSDL ファイルの URL を取得します。詳細は，「8.4.3(3)　WSDL ファイルの取得」を参照してください。

3. Java ソースの生成

Web サービスクライアントを実装するための Java ソースを生成します。詳細は，「8.4.3(4)　Java ソースの生成」を参照してください。

4. Web サービスクライアントの実装

生成された Java ソースを利用して，Web サービスクライアントを実装します。詳細は，「8.4.3(5) Web サービスクライアントの実装」を参照してください。

5. ejb-jar.xml の編集

Web サービスクライアントを実装した EJB の定義を追加します。詳細は，「8.4.3(6)　ejb-jar.xml の編集」を参照してください。

6. エンタープライズアプリケーションプロジェクトの作成とモジュールの追加

エンタープライズアプリケーションプロジェクトを作成して，EJB プロジェクトを追加します。詳細は，「8.4.3(7)　エンタープライズアプリケーションプロジェクトの作成とモジュールの追加」を参照してください。

7. J2EE アプリケーションのデプロイとデバッグ

J2EE アプリケーションを J2EE サーバにデプロイして，デバッグします。詳細は，「8.4.3(8)　J2EE アプリケーションのデプロイとデバッグ」を参照してください。

以降，この流れに沿って EJB を使用した Web サービスクライアントの開発の手順を説明します。

(2)　EJB プロジェクトの作成

EJB を使用した Web サービスクライアントの開発に使用する EJB プロジェクトを作成します。EJB プロジェクトの作成方法については，「4.4.2　EJB プロジェクトの作成」を参照してください。

(3)　WSDL ファイルの取得

呼び出そうとしている Web サービスのメタ情報が記述された WSDL ファイルを取得します。または WSDL ファイルの URL が公開されている場合はその URL を取得します。

(4)　Java ソースの生成

取得した WSDL ファイルを基に，Web サービスクライアントの開発や実行に必要な Java ソース（サービスクラス，SEI，および Java Bean（スタブ））を生成します。Java ソースは，Eclipse を使って生成できます。Java ソースを生成する手順は，「8.4.1(4)　Java ソースの生成」と同様ですが，［Web サービス・クライアント］ダイアログの［プロジェクト］には，ワークスペースの EJB プロジェクト名を選択します。

また，次に示す操作や設定をした場合，［Web サービス・クライアント］ダイアログを表示すると，［Java ソース・フォルダー］が空欄となり，以降の操作ができなくなります。そのため，［Java ソース・フォルダー］には，プロジェクトルート以外を設定してください。

- Web サービスクライアントの開発に使用するプロジェクトの［プロパティ］ダイアログの［Java のビルド・パス］ページで，［ソース］タブの［ビルド・パス上のソース・フォルダ］にプロジェクトルートを設定した場合

(5)　Web サービスクライアントの実装

生成された Java ソースを利用して，Web サービスクライアントを実装します。Web サービスクライアントの実装例については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「3.6 Web サービスクライアントの実装」を参照してください。

(6)　ejb-jar.xml の編集

Web サービスクライアントを実装した EJB の定義を ejb-jar.xml に追加します。

(7)　エンタープライズアプリケーションプロジェクトの作成とモジュールの追加

(a)　エンタープライズアプリケーションプロジェクトの作成

エンタープライズアプリケーションプロジェクトを作成します。エンタープライズアプリケーションプロジェクトの作成手順については，「4.4.4　エンタープライズアプリケーションプロジェクトの作成」を参照してください。

(b)　エンタープライズアプリケーションプロジェクトへのモジュールの追加

エンタープライズアプリケーションプロジェクトに，「8.4.3(2)　EJB プロジェクトの作成」で作成した Web サービスクライアント用の EJB プロジェクトを追加します。EJB プロジェクトの追加手順を次に示します。

1. [New EAR Application Project] ダイアログの［EAR Application Project］ページの［次へ］ボタンをクリックします。

   ［Enterprise Application］ページが表示されます。

2. 追加するモジュールをチェックして，［終了］ボタンをクリックします。

(8)　J2EE アプリケーションのデプロイとデバッグ

作成したプロジェクトをデプロイしてデバッグします。

(a)　J2EE アプリケーションのデプロイ

エンタープライズアプリケーションプロジェクトから作成された J2EE アプリケーションを J2EE サーバにデプロイします。デプロイ方法については，「6.6　J2EE サーバへのプロジェクトの公開」を参照してください。

(b)　J2EE アプリケーションのデバッグ

作成した J2EE アプリケーションをデバッグします。デバッグ方法については，「6.7　J2EE アプリケーションのデバッグと実行」を参照してください。

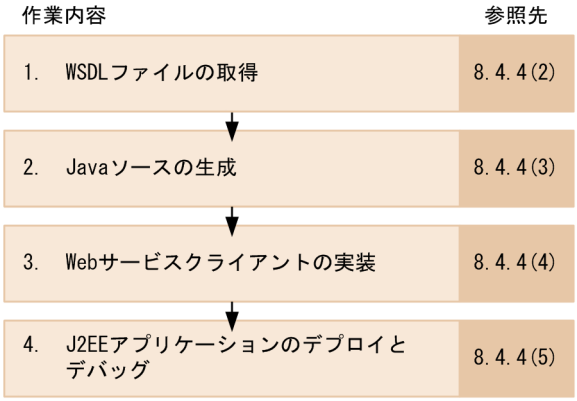
8.4.4　既存の Web サービスを使用した Web サービスクライアントの開発

既存の Web サービスを使用した Web サービスクライアントの開発方法について説明します。なお，ここでは，Web サービス用の動的 Web プロジェクトがすでに存在するものとして説明します。

(1) 既存の Web サービスを使用した Web サービスクライアントの開発の流れ

既存の Web サービスを実装した既存の動的 Web プロジェクトに，Web サービスクライアントの機能を実装して Web サービスクライアントを開発できます。既存の Web サービスを使用した Web サービスクライアントの開発の流れを，次の図に示します。

図 8-6　既存の Web サービスを使用した Web サービスクライアントの開発の流れ



それぞれの作業の概要を次に示します。

1. WSDL ファイルの取得

   WSDL ファイルを取得します。または，公開されている WSDL ファイルの URL を取得します。詳細は，「8.4.4(2)　WSDL ファイルの取得」を参照してください。

2. Java ソースの生成

   Web サービスクライアントを実装するための Java ソースを生成します。詳細は，「8.4.4(3)　Java ソースの生成」を参照してください。

3. Web サービスクライアントの実装

   生成された Java ソースを利用して，Web サービスクライアントを実装します。詳細は，「8.4.4(4) Web サービスクライアントの実装」を参照してください。

4. J2EE アプリケーションのデプロイとデバッグ

   J2EE アプリケーションを J2EE サーバにデプロイして，デバッグします。詳細は，「8.4.4(5)　J2EE アプリケーションのデプロイとデバッグ」を参照してください。

以降，この流れに沿って既存の Web サービスを使用した Web サービスクライアントの開発の手順を説明します。

(2) WSDL ファイルの取得

呼び出そうとしている Web サービスのメタ情報が記述された WSDL ファイルを取得します。または WSDL ファイルの URL が公開されている場合はその URL を取得します。

(3)　Java ソースの生成

取得した WSDL ファイルを基に，Web サービスクライアントの開発や実行に必要な Java ソース（サービスクラス，SEI，および Java Bean（スタブ））を生成します。Java ソースは，Eclipse を使って生成できます。Java ソースを生成する手順は，「8.4.1(4)　Java ソースの生成」と同様ですが，［Web サービス・クライアント］ダイアログの［プロジェクト］には，ワークスペースの動的 Web プロジェクト名を選択します。

また，次に示す操作や設定をした場合，［Web サービス・クライアント］ダイアログを表示すると，［Java ソース・フォルダー］が空欄となり，以降の操作ができなくなります。そのため，［Java ソース・フォルダー］には，プロジェクトルート以外を設定してください。

- Web サービスクライアントの開発に使用するプロジェクトの［プロパティ］ダイアログの［Java のビルド・パス］ページで，［ソース］タブの［ビルド・パス上のソース・フォルダ］にプロジェクトルートを設定した場合

(4)　Web サービスクライアントの実装

生成された Java ソースを利用して，Web サービスクライアントを実装します。Web サービスクライアントの実装例については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「3.6 Web サービスクライアントの実装」を参照してください。

(5)　J2EE アプリケーションのデプロイとデバッグ

作成したプロジェクトをデプロイして，デバッグします。

(a)　J2EE アプリケーションのデプロイ

エンタープライズアプリケーションプロジェクトから作成された J2EE アプリケーションを J2EE サーバにデプロイします。デプロイ方法については，「6.6　J2EE サーバへのプロジェクトの公開」を参照してください。

(b)　J2EE アプリケーションのデバッグ

作成した J2EE アプリケーションをデバッグします。デバッグ方法については，「6.7　J2EE アプリケーションのデバッグと実行」を参照してください。

9 バッチアプリケーションの開発

バッチアプリケーションは，バッチサーバで実行する Java アプリケーションです。バッチアプリケーションでは，リソースアクセス，EJB アクセス，トランザクション管理といった Java EE 機能の一部を使用できます。

この章では，Eclipse プラグインを使用したバッチアプリケーションの開発として，バッチアプリケーションの作成から実行までの手順，および注意事項について説明します。

9.1　バッチアプリケーションの開発の流れ

Eclipse を使用したバッチアプリケーションの開発の流れを，次の図に示します。

図 9-1　Eclipse を使用したバッチアプリケーションの開発の流れ

それぞれの作業の概要を説明します。

1. バッチサーバの環境構築

運用管理ポータルや Smart Composer 機能を使用して，バッチアプリケーションを実行するバッチサーバを構築します。詳細は，「9.2　バッチサーバの環境構築」を参照してください。

2. Java プロジェクトの作成

Eclipse の機能を使用して，バッチアプリケーションを開発するための Java プロジェクトを作成します。詳細は，「9.3　Java プロジェクトの作成」を参照してください。

3. Java プロジェクトへのバッチライブラリの追加

バッチアプリケーションの開発で使う Java プロジェクトにバッチライブラリを追加します。詳細は，「9.4　Java プロジェクトへのバッチライブラリの追加と削除」を参照してください。

4. バッチアプリケーションの作成

Eclipse の機能を使用して，バッチアプリケーションを作成します。Java EE 機能など外部のライブラリを使用する場合は，コンパイル時のオプションに，クラスパス（ビルドパス）を設定します。詳細は，「9.5　バッチアプリケーションの作成」を参照してください。

5. バッチアプリケーションのデバッグ

バッチアプリケーションに対して，デバッグを実行します。詳細は，「9.6　バッチアプリケーションのデバッグ」を参照してください。

6. バッチアプリケーションの実行

バッチアプリケーションを実行します。詳細は，「9.7.1　バッチアプリケーションの実行」を参照してください。

7. バッチアプリケーションの強制停止

必要に応じてバッチアプリケーションを強制的に停止します。詳細は，「9.7.2　バッチアプリケーションの強制停止」を参照してください。

8. バッチサーバの停止

バッチアプリケーションを実行したバッチサーバを停止します。詳細は，「9.8　バッチサーバの停止」を参照してください。

以降の節では，この流れに沿ってバッチアプリケーションの開発手順を説明します。

9.1.1　バッチアプリケーション開発時の注意事項

- Eclipse は必ず管理者権限で起動してください。
- UAC（ユーザアカウント制御）が有効な Windows で，バッチアプリケーション開発をする場合は，<Eclipse のインストールディレクトリ>¥eclipse¥eclipse.exe を右クリックして［管理者として実行］を選択する，などの方法で，権限を管理者に昇格させて Eclipse を起動してください。

  管理者のパスワードまたは確認を求められた場合は，画面の指示に従って，パスワードを入力してください。

9.2　バッチサーバの環境構築

バッチアプリケーションを実行する前にバッチサーバを構築します。バッチサーバの構築には，運用管理ポータルまたは Smart Composer 機能を使用します。

バッチサーバは，実行するバッチアプリケーションがあるマシンに構築してください。また，バッチサーバは複数のバッチアプリケーションを同時に実行またはデバッグできません。複数のバッチアプリケーションを同時に実行またはデバッグしたい場合は，それぞれのバッチアプリケーションに対して一つのバッチサーバを用意してください。

また，あらかじめ Management Server および運用管理エージェントを起動しておいてください。

運用管理ポータルでのバッチサーバの構築については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「5.　バッチアプリケーションを実行するシステムの構築と削除」を参照してください。Smart Composer 機能でのバッチサーバの構築については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「4.6　バッチアプリケーションを実行するシステムの構築」を参照してください。

注意事項

バッチアプリケーション開発プラグインで操作するバッチサーバを運用管理ポータルで構築する場合，運用管理ポータルの［J2EE サーバの基本設定］画面の［基本設定］タブで，［起動オプションの設定］－［セキュリティマネージャの使用］を［する］に設定してください。

［セキュリティマネージャの使用］を［しない］に設定すると，バッチアプリケーション開発プラグインでセキュリティマネージャを使用してバッチサーバを起動できません。

9.3　Java プロジェクトの作成

Java プロジェクトを作成する手順については，「8.4.1(2)　Java プロジェクトの作成」を参照してください。

9.4　Java プロジェクトへのバッチライブラリの追加と削除

バッチアプリケーションを開発する場合，Java プロジェクトにバッチライブラリを追加する必要があります。ここでは，バッチライブラリの追加と削除の手順を説明します。

9.4.1　Java プロジェクトへのバッチライブラリの追加

バッチライブラリは，Eclipse の［ライブラリーの追加］ダイアログで追加します。

(1)　［ライブラリーの追加］ダイアログの表示方法

［ライブラリーの追加］ダイアログの表示方法を説明します。

［ライブラリーの追加］ダイアログを表示するには，次に示す方法があります。

- コンテキストメニューから表示する方法
- プロジェクトのプロパティダイアログから表示する方法

それぞれの手順を示します。

(a)　コンテキストメニューから表示する方法

1. ［J2EE］パースペクティブの［プロジェクト・エクスプローラー］ビューで，バッチアプリケーションを開発している Java プロジェクトを選択します。
2. コンテキストメニューから［ビルド・パス］－［ライブラリーの追加］を選択します。
   ［ライブラリーの追加］ダイアログが表示されます。

(b)　プロジェクトのプロパティダイアログから表示する方法

1. ［J2EE］パースペクティブの［プロジェクト・エクスプローラー］ビューで，バッチアプリケーションを開発している Java プロジェクトを選択します。
2. Eclipse のメニューから［プロジェクト］－［プロパティ］を選択します。
   ［プロパティ： <プロジェクト名>］ダイアログが表示されます。
3. ［プロパティ： <プロジェクト名>］ダイアログの左ペインで［Java のビルド・パス］を選択します。
   ［Java のビルド・パス］ページが表示されます。
4. ［ライブラリー］タブを選択し，［ライブラリーの追加］ボタンをクリックします。
   ［ライブラリーの追加］ダイアログが表示されます。

(2) バッチライブラリの追加方法

バッチライブラリの追加手順を次に示します。

1. [ライブラリーの追加] ダイアログの［ライブラリーの追加］ページで，ライブラリのリストから［Cosminexus バッチ・ライブラリー］を選択して，［次へ］ボタンをクリックします。
   ［Cosminexus バッチ・ライブラリー］ページが表示されます。

2. [終了] ボタンをクリックします。
   Java プロジェクトのビルドパスに，バッチアプリケーションの開発で使用するバッチライブラリが追加されます。

注意事項

追加したバッチライブラリは編集できません。［プロパティ： <プロジェクト名>］ダイアログの［Java のビルド・パス］ページの［ビルド・パス上の JAR およびクラス・フォルダー］で，［Cosminexus バッチ・ライブラリー］を選択して，［編集］ボタンをクリックすると，［ライブラリーの編集］ダイアログが表示されますが，このダイアログで［終了］ボタン，または［キャンセル］ボタンをクリックしても，変更なく［ライブラリーの編集］ダイアログが閉じられます。

9.4.2 Java プロジェクトからのバッチライブラリの削除

ここでは，Java プロジェクトのビルドパスに追加したバッチライブラリを削除する方法を説明します。

バッチライブラリを削除するには，次に示す方法があります。

- コンテキストメニューから削除する方法
- プロジェクトのプロパティダイアログから削除する方法

それぞれの手順を示します。

(1) コンテキストメニューから削除する方法

1. [J2EE] パースペクティブの［プロジェクト・エクスプローラー］ビューで，［Cosminexus バッチ・ライブラリー］を選択します。

2. コンテキストメニューから［ビルド・パス］－［ビルド・パスから削除］を選択します。
   バッチライブラリが削除されます。

(2) プロジェクトのプロパティダイアログから削除する方法

1. [J2EE] パースペクティブの［プロジェクト・エクスプローラー］ビューで，バッチアプリケーションを開発している Java プロジェクトを選択します。

2. Eclipse のメニューバーから［プロジェクト］－［プロパティ］を選択します。
   ［プロパティ： <プロジェクト名>］ダイアログが表示されます。

3. ［プロパティ： <プロジェクト名>］ダイアログの左ペインで［Java のビルド・パス］を選択します。
   ［Java のビルド・パス］ページが表示されます。

4. ［ライブラリー］タブを選択し，［ビルド・パス上の JAR およびクラス・フォルダー］で［Cosminexus バッチ・ライブラリー］を選択します。

5. ［除去］ボタンをクリックします。
   バッチライブラリが削除されます。

9.5　バッチアプリケーションの作成

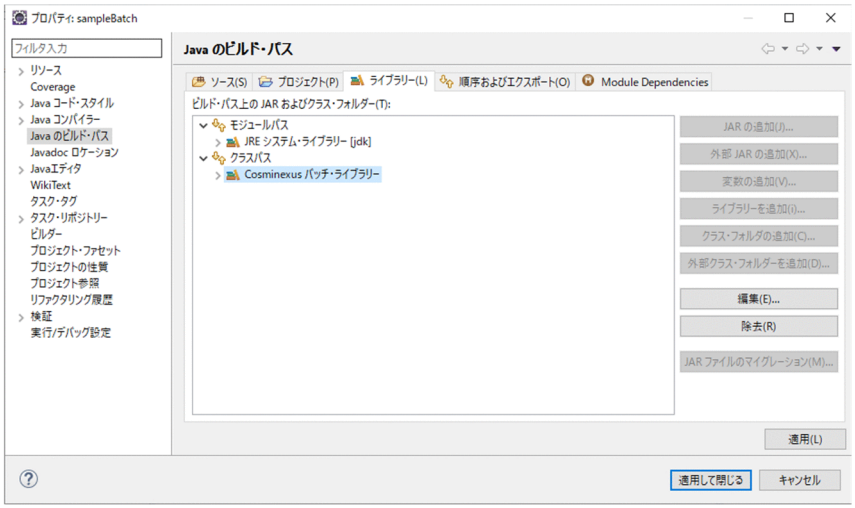
バッチアプリケーションは，バッチライブラリを追加した Java プロジェクトに作成します。バッチアプリケーション作成時の注意事項については，マニュアル「アプリケーションサーバ 機能解説 拡張編」の「2.3.11　バッチアプリケーション作成時の注意」を参照してください。

また，J2EE アプリケーションと同様に外部のライブラリを使用する場合は，ビルドパス（コンパイル時のオプションとなるクラスパス）を追加する必要があります。ここでは，ejbserver.jar を追加する手順を例として，ビルドパスの追加手順を説明します。

1. [J2EE] パースペクティブの［プロジェクト・エクスプローラー］ビューで，バッチアプリケーションを開発している Java プロジェクトを選択します。

2. Eclipse のメニューバーから［プロジェクト］－［プロパティ］を選択します。
   ［プロパティ： <プロジェクト名>］ダイアログが表示されます。

3. [プロパティ： <プロジェクト名>］ダイアログの左ペインで［Java のビルド・パス］を選択します。
   [Java のビルド・パス］ページが表示されます。



4. [ライブラリー］タブを選択し，［外部 JAR の追加］ボタンをクリックし，次に示すライブラリを選択します。
   <Developerのインストールディレクトリ>¥CC¥lib¥ejbserver.jar

5. [プロパティ： <プロジェクト名>］ダイアログの［OK］ボタンをクリックします。
   ビルドパスが追加されます。

ほかのビルドパスも同様の手順で追加できます。

なお，バッチサーバのクラスパスにライブラリを追加するには，運用管理ポータルまたは Smart Composer 機能で，コンテナ拡張ライブラリの設定をしてください。運用管理ポータルを使用する場合は，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.8.2　J2EE コンテナの設定」を参照

してください。Smart Composer 機能を使用する場合は，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「19.　コンテナ拡張ライブラリ」を参照してください。

9.6　バッチアプリケーションのデバッグ

作成したバッチアプリケーションをデバッグします。バッチアプリケーションのデバッグは，Eclipse で実行できます。

9.6.1　デバッグ環境の設定

デバッグ環境としてバッチサーバの起動構成を作成します。起動構成では，バッチアプリケーションについて次の内容を指定します。

- メインクラスおよびバッチサーバ
- 起動時の引数および Java VM の起動オプション
- クラスパス
- ソースパス

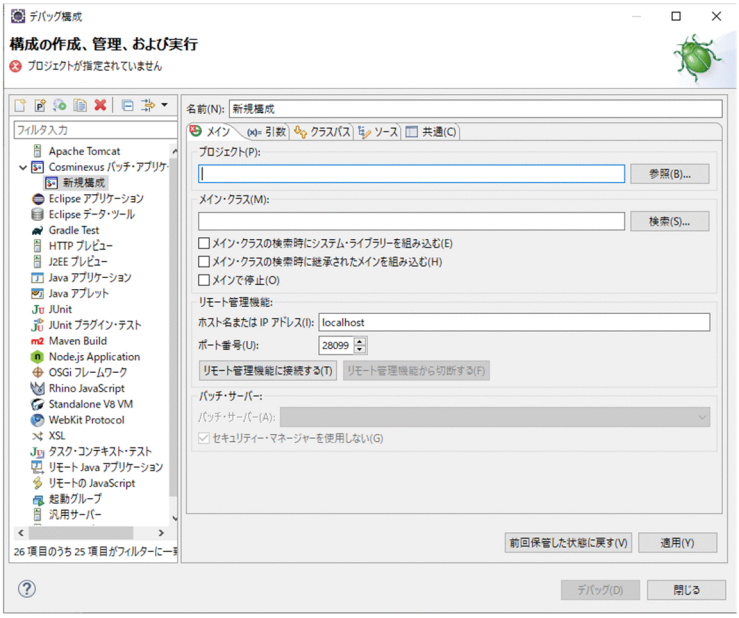
起動構成は，［デバッグ構成］ダイアログで作成します。［デバッグ構成］ダイアログで起動構成を作成する手順を次に示します。

1. Eclipse のメニューバーから，［実行］－［デバッグの構成］を選択します。
   ［デバッグ構成］ダイアログが表示されます。
2. ［デバッグ構成］ダイアログの左ペインで［Cosminexus バッチ・アプリケーション］を選択し，ツールバーの［ （新規の起動構成）］をクリックします。
   ［構成の作成、管理、および実行］ページが表示されます。
3. 右ペインの［名前］に，起動構成の名前を指定します。
4. バッチアプリケーションの起動方法を構成する基本的な項目，引数，クラスパスなどの項目をタブごとに指定します。
5. ［閉じる］ボタンをクリックします。
   ［デバッグ構成］ダイアログで指定した項目が起動構成に保管されます。
   ［変更を保管します］ダイアログが表示された場合は，［はい］ボタンをクリックしてください。［デバッグ構成］ダイアログが閉じて，指定した項目が起動構成に保管されます。

バッチアプリケーションの起動構成に必要な内容は，右ペインに表示されるタブで指定します。

(1)　メインクラスおよびバッチサーバの指定

デバッグするために必要なバッチアプリケーションのメインクラスや実行するバッチサーバ，およびバッチサーバを指定するために必要な Management Server リモート管理機能への接続は，［デバッグ構成］ダイアログの［メイン］タブで指定します。



［メイン］タブでは，次の項目を指定します。

項目名	指定値
名前	起動構成の名前を指定します。
プロジェクト	デバッグするバッチアプリケーションを開発している Java プロジェクトを指定します。
メイン・クラス	デバッグするバッチアプリケーションのメインクラスを指定します。
ホスト名または IP アドレス※1	Management Server リモート管理機能の接続先のホスト名として「localhost」を指定します。
ポート番号※1	Management Server リモート管理機能の接続先のポート番号を指定します。
リモート管理機能に接続する	リモート管理機能に接続します。接続すると，バッチサーバを指定※2 できます。 Management Server の管理ユーザの設定で管理ユーザの認証を［認証あり］に設定している場合は，［ログイン - リモート管理］ダイアログが表示されます。
バッチ・サーバー※3	バッチアプリケーションを実行するサーバを選択します。

注※1

リモート管理機能の接続情報は，起動構成単位では保存できません。バッチアプリケーションの起動構成が複数ある場合は，［リモート管理機能］グループの設定が共有されます。そのため，一つの起動構成の接続情報を変更すると，ほかの起動構成の接続情報も変更されます。

また，バッチアプリケーションの起動構成をすべて削除しても，リモート管理機能の設定は残ります。

また，［ホスト名または IP アドレス］を指定していない場合は，リモート管理機能に接続できません。

注※2

バッチサーバの選択は，起動構成単位で保存できます。

注※3

リモート管理機能に接続するホストを切り替えたり，運用管理ポータルでバッチサーバを破棄したりしたあと，［リモート管理機能に接続する］ボタンをクリックしてリモート管理機能に接続した場合，［バッチ・サーバー］コンボボックスに前回選択していたバッチサーバが存在しないときは，ユーザの選択していないバッチサーバが［バッチ・サーバー］コンボボックスで選択されます。この場合は，［バッチ・サーバー］コンボボックスで適切なバッチサーバを再度選択し保管してください。

また，［バッチ・サーバー］コンボボックスには，運用管理ポータルで追加しただけのセットアップしていないバッチサーバも表示されます。セットアップしていないバッチサーバを選択すると，バッチアプリケーションは実行できないため，Management Server のエラーになります。

必要に応じて，次の項目を指定してください。

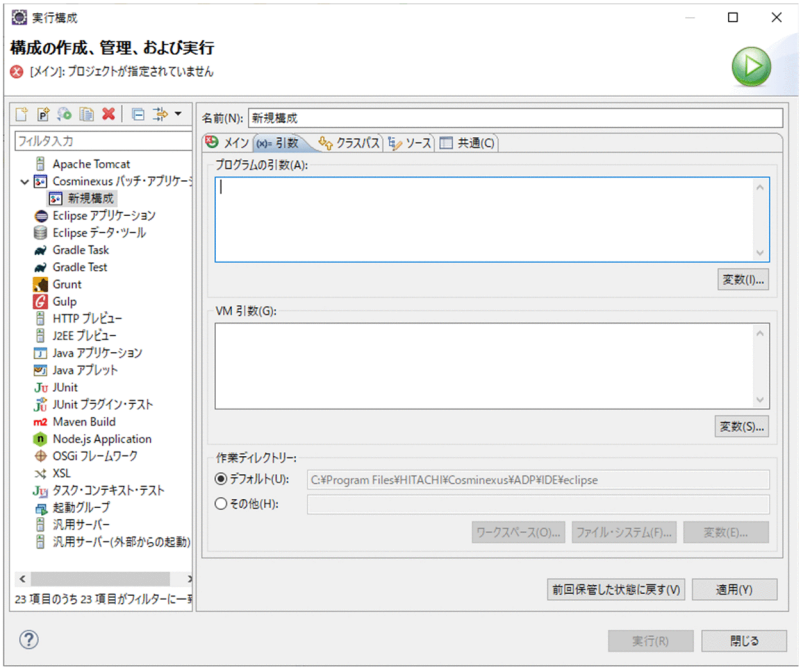
項目名	指定値
メイン・クラスの検索時にシステム・ライブラリーを組み込む	［メイン・クラス］の［検索］ボタンをクリックしてメインクラスを検索する場合で，システムライブラリも検索範囲に含めるときにチェックします。
メイン・クラスの検索時に継承されたメインを組み込む	［メイン・クラス］の［検索］ボタンをクリックしてメインクラスを検索する場合で，メインクラスを継承したクラスも検索範囲に含めるときにチェックします。
メインで停止	デバッグ時，main メソッドにブレークポイントを設定していなくても，自動的にメインで中断するときにチェックします。
リモート管理機能から切断する	リモート管理機能から切断します。切断すると，バッチサーバを選択できません。Management Server の管理ユーザの設定で管理ユーザの認証を［認証あり］に設定している場合は，［ログイン - リモート管理］ダイアログが表示されます。
セキュリティー・マネージャーを使用しない	バッチサーバを起動するときに，セキュリティマネージャを使用するかどうかを指定します。 • セキュリティマネージャを使用する場合 ［セキュリティー・マネージャーを使用しない］をチェックしません。 • セキュリティマネージャを使用しない場合 ［セキュリティー・マネージャーを使用しない］をチェックします。

注意事項

［リモート管理機能］グループの設定は，［リモート管理機能に接続する］ボタンをクリックした場合にだけ適用されます。ほかのフィールドを変更し［適用］ボタンが活性化された場合に［適用］ボタンをクリックしても［リモート管理機能］グループの設定は適用されません。同様に，［前回保管した状態に戻す］ボタンが活性化された場合に［前回保管した状態に戻す］ボタンをクリックしても［リモート管理機能］グループの設定は戻りません。

(2)　起動時の引数および Java VM の起動オプションの指定

デバッグするバッチアプリケーションの起動時の引数，および Java VM の起動オプションは，［デバッグ構成］ダイアログの［引数］タブで指定します。

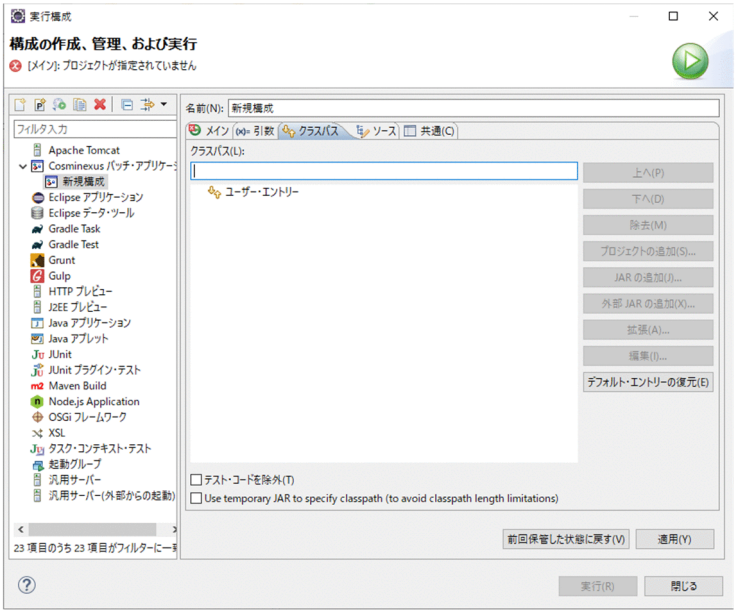


[引数] タブでは，次の項目を指定します。

項目名	指定値
プログラムの引数	バッチアプリケーションの起動時の引数を指定します。
VM 引数	バッチアプリケーションを実行する Java VM の起動オプション，およびシステムプロパティを指定します。
作業ディレクトリー	バッチアプリケーションを実行するフォルダを指定します。

(3) クラスパスの指定

デバッグするバッチアプリケーションのクラスパスは，[デバッグ構成] ダイアログの [クラスパス] タブで指定します。通常，プロジェクトのビルドパスに必要なライブラリを設定しておけば，クラスパスの指定は不要です。実行時にだけ参照するクラスパスがあれば，そのクラスパスを追加します。



[クラスパス] タブでは，次の項目を指定します。

項目名	指定値
クラスパス	バッチアプリケーションを実行するときに参照するクラスパスを指定します。

(4)　ソースパスの指定

デバッグで中断した場合，中断個所のソースを検索するパスは，[デバッグ構成] ダイアログの [ソース] タブで指定します。通常，プロジェクト内にソースが存在するため，指定する必要はありませんが，外部ライブラリを使用する場合などに，そのライブラリのソースコードが存在するパスを指定します。

(5)　共通項目の指定

起動構成のファイルへの保存，起動構成のメニューバーへの表示，起動構成のコンソール出力のファイルへの出力など，すべての起動構成で共通に設定できるオプションは，[デバッグ構成] ダイアログの [共通] タブで指定します。

9.6.2　デバッグの実行

バッチアプリケーションのデバッグは，Eclipse で実行します。ここでは「9.6.1　デバッグ環境の設定」でバッチアプリケーションの Management Server リモート管理機能への接続を設定していることを前提としています。なお，バッチアプリケーションの実行については，「9.7.1　バッチアプリケーションの実行」を参照してください。

バッチアプリケーションをデバッグするには，次に示す方法があります。

- ショートカットからデバッグする方法

- ［デバッグ構成］ダイアログを表示してデバッグする方法

それぞれの手順を示します。

(1) ショートカットからデバッグする方法

1. ［プロジェクト・エクスプローラー］ビューでデバッグするプロジェクトの Java ソースファイルを開いて，ブレークポイントを設定します。

2. ［プロジェクト・エクスプローラー］ビューで，デバッグするプロジェクトを選択するか，または Java ファイルをエディタで開きます。

3. Eclipse のメニューから［実行］－［デバッグ］－［Cosminexus バッチ・アプリケーション］を選択します。

   デバッグが開始されます。バッチサーバが起動されていない場合は，バッチサーバが起動します。

   Management Server の管理ユーザの設定で管理ユーザの認証を［認証あり］に設定している場合で Management Server リモート管理機能に未接続のときは，［ログイン - リモート管理］ダイアログが表示されるので，管理ユーザ ID とパスワードを入力して，［OK］ボタンをクリックします。

   ブレークポイントに達すると，処理が中断され，［デバッグ］パースペクティブを開くかどうかのメッセージが表示されます。［はい］ボタンをクリックすると，［デバッグ］パースペクティブが表示されます。

   なお，ブレークポイントで中断された処理を再開したい場合は，Eclipse のメニューから［実行］－［再開］を選択してください。

(2) ［デバッグ構成］ダイアログを表示してデバッグする方法

1. ［プロジェクト・エクスプローラー］ビューでデバッグするプロジェクトの Java ソースファイルを開いて，ブレークポイントを設定します。

2. Eclipse のメニューから［実行］－［デバッグの構成］を選択します。

   ［デバッグ構成］ダイアログが表示されます。

3. 作成した起動構成を選択して，［デバッグ］ボタンをクリックします。

   デバッグが開始されます。バッチサーバが起動されていない場合は，バッチサーバが起動します。

   Management Server の管理ユーザの設定で管理ユーザの認証を［認証あり］に設定している場合で Management Server リモート管理機能に未接続のときは，［ログイン - リモート管理］ダイアログが表示されるので，管理ユーザ ID とパスワードを入力して，［OK］ボタンをクリックします。

   ブレークポイントに達すると，処理が中断され，［デバッグ］パースペクティブを開くかどうかのメッセージが表示されます。［はい］ボタンをクリックすると，［デバッグ］パースペクティブが表示されます。

   なお，ブレークポイントで中断された処理を再開したい場合は，Eclipse のメニューから［実行］－［再開］を選択してください。

(3) デバッグ時の注意事項

- バッチサーバが一つも構築されていない場合は，エラーになります。
- デバッグ開始時に，デバッグの対象となるメインクラス，またはデバッグを実行するバッチサーバが複数ある場合は，［バッチ・アプリケーションの選択］ダイアログが表示されるので，デバッグの対象となるメインクラス，またはデバッグを実行するバッチサーバを指定してください。
- デバッグ対象となる同一の起動構成が複数ある場合は，［バッチ・アプリケーションの選択］ダイアログが表示されるので，デバッグを実行する起動構成を選択してください。
- 起動済みのバッチサーバの設定を，運用管理ポータルや Smart Composer 機能で変更した場合，バッチサーバを再起動してください。
- バッチアプリケーションの初回デバッグ時には，バッチサーバを起動するため，バッチアプリケーションのデバッグまでに時間が掛かります。
- Management Server の設定で，バッチサーバの自動再起動が有効になっている場合，バッチアプリケーションの強制停止に失敗すると，バッチサーバが自動的に再起動されます。バッチサーバの再起動中は，バッチサーバを操作できません。また，デバッグモードで起動していたバッチサーバの再起動が完了すると，自動的にバッチサーバへデバッグ接続（再接続）されます。
- バッチアプリケーションがブレークポイントで中断している状態で，バッチアプリケーションの強制停止，バッチサーバの停止，または Eclipse を終了した場合，バッチアプリケーションはブレークポイントの位置で停止されないで，ブレークポイント以降の数ステップの処理が実行されたあと，停止するときがあります。

9.7　バッチアプリケーションの実行と停止

Eclipse でバッチアプリケーションを実行および強制停止する方法について説明します。

9.7.1　バッチアプリケーションの実行

バッチアプリケーションを実行する場合もデバッグ時と同様に，Eclipse から実行できます。ここでは「9.6.1　デバッグ環境の設定」でバッチアプリケーションの Management Server リモート管理機能への接続を設定していることを前提としています。

バッチアプリケーションを実行するには，次に示す方法があります。

- ショートカットからバッチアプリケーションを実行する方法
- ［実行構成］ダイアログを表示してバッチアプリケーションを実行する方法

それぞれの手順を示します。

(1)　ショートカットからバッチアプリケーションを実行する方法

1. ［プロジェクト・エクスプローラー］ビューで，実行するプロジェクトを選択する，または Java ファイルをエディタで開きます。
2. Eclipse のメニューから［実行］－［実行］－［Cosminexus バッチ・アプリケーション］を選択します。

   バッチアプリケーションの実行が開始されます。バッチサーバが起動されていない場合は，バッチサーバが起動します。

   Management Server の管理ユーザの設定で管理ユーザの認証を［認証あり］に設定している場合で Management Server リモート管理機能に未接続のときは，［ログイン - リモート管理］ダイアログが表示されるので，管理ユーザ ID とパスワードを入力して，［OK］ボタンをクリックします。

(2)　［実行構成］ダイアログを表示してバッチアプリケーションを実行する方法

1. Eclipse のメニューから［実行］－［実行構成］を選択します。

   ［実行構成］ダイアログが表示されます。
2. 作成した起動構成を選択して，［実行］ボタンをクリックします。

   バッチアプリケーションの実行が開始されます。バッチサーバが起動されていない場合は，バッチサーバが起動します。

   Management Server の管理ユーザの設定で管理ユーザの認証を［認証あり］に設定している場合で Management Server リモート管理機能に未接続のときは，［ログイン - リモート管理］ダイアログが表示されるので，管理ユーザ ID とパスワードを入力して，［OK］ボタンをクリックします。

(3) 実行時の注意事項

- バッチアプリケーションを実行する場合は，実行を中断したり，変数や式を調査したりすることはできません。
- バッチサーバが一つも構築されていない場合は，エラーになります。
- 実行開始時に，実行の対象となるメインクラス，または実行するバッチサーバが複数ある場合は，［バッチ・アプリケーションの選択］ダイアログが表示されるので，実行の対象となるメインクラスまたは実行するバッチサーバを指定してください。
- 実行の対象となる同一の起動構成が複数ある場合は，［バッチ・アプリケーションの選択］ダイアログが表示されるので，実行する起動構成を選択してください。
- 起動済みのバッチサーバの設定を，運用管理ポータルや Smart Composer 機能で変更した場合，バッチサーバを再起動してください。
- バッチアプリケーションの初回実行時には，バッチサーバを起動するため，バッチアプリケーションの実行までに時間が掛かります。
- Management Server の設定で，バッチサーバの自動再起動が有効になっている場合，バッチアプリケーションの強制停止に失敗すると，バッチサーバが自動的に再起動されます。バッチサーバの再起動中は，バッチサーバを操作できません。

9.7.2 バッチアプリケーションの強制停止

バッチ処理の途中で，デバッグ中または実行中のバッチアプリケーションを強制的に停止できます。デバッグ中または実行中のバッチアプリケーションを強制的に停止する手順を次に示します。

1. ［デバッグ］ビューで［<起動構成名> [Cosminexus バッチ・アプリケーション]］を選択します。
2. ［デバッグ］ビューのツールバーの［ 🟥 （終了)］をクリックします。
   バッチアプリケーションが終了します。

注意事項

バッチアプリケーションを強制停止する際の注意を次に示します。

- ［デバッグ］ビューに表示された cjkilljob.exe プロセスの終了値が 2 の場合，［コンソール］ビューにある cjkilljob.exe プロセスのコンソールを参照して，異常終了または警告終了の原因を取り除いたあとで，バッチサーバを停止してください。
- ［デバッグ］ビューで［終了］ボタンを連続でクリックした場合，バッチサーバへのデバッグ接続が切断されるときがあります。バッチサーバへのデバッグ接続が切断された場合，バッチサーバを停止したあとで，バッチアプリケーションをデバッグ，または実行してください。

- バッチアプリケーションの強制停止が失敗すると，Management Server の設定で，バッチサーバが自動的に再起動されます。バッチサーバを再起動中に Eclipse を終了しても，バッチサーバの再起動は中断されません。バッチサーバの再起動が完了してから Eclipse を起動すると，再起動されたバッチサーバは，外部で起動したバッチサーバとなり，バッチアプリケーションのデバッグ，または実行ができません。この場合，Management Server 運用管理ポータルなどでバッチサーバを停止してください。

9.8　バッチサーバの停止

ここでは，バッチサーバを停止する手順を説明します。

バッチ処理の終了したバッチサーバやバッチ処理の途中で実行中のバッチサーバを停止できます。実行中のバッチサーバを停止する手順を次に示します。

1. [デバッグ] ビューで［<起動構成名> [Cosminexus バッチ・サーバー]］を選択します。
2. [デバッグ] ビューのツールバーの［ (終了)］をクリックします。
   バッチサーバが停止します。
3. [デバッグ構成] ダイアログ，または［実行構成］ダイアログの［メイン］タブの［リモート管理機能から切断する］ボタンをクリックします。
   リモート管理機能との接続が切断されます。

注意事項

バッチサーバを停止すると，バッチアプリケーションの起動構成が停止するため，バッチサーバ上で実行中のバッチアプリケーションがある場合は，そのバッチアプリケーションも終了します。

付録

付録 A　Eclipse の動作確認用サンプルプロジェクト

Developer では Eclipse の動作確認用のサンプルプロジェクトとして，組み込みデータベースを使用するサンプルプロジェクト（Bank）を提供しています。サンプルプロジェクト（Bank）は，Eclipse が提供する開発支援機能，組み込みデータベースの構築および接続のチュートリアルとして使用できます。また，サンプルプロジェクト（Bank）を使用することで，Eclipse によるデータベースを使用した J2EE アプリケーション開発手順の理解を深められます。

ここでは，サンプルプロジェクトの概要およびサンプルプロジェクトの実行手順を説明します。

参考

なお，このほかにも，アプリケーションサーバではサンプルプログラムが提供されています。アプリケーションサーバで提供するサンプルプログラムについては，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「付録 L　アプリケーションサーバが提供するサンプルプログラム」を参照してください。

付録 A.1　サンプルプロジェクトの概要

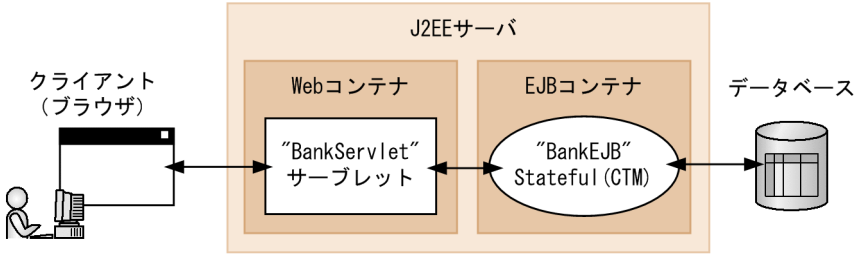
ここでは，Developer が提供するサンプルプロジェクト（Bank）の概要を説明します。

(1)　サンプルプロジェクトの内容

Developer が提供しているサンプルプロジェクト（Bank）は，Web ブラウザから入力されたユーザ ID（User ID）を持つユーザの資金を当座預金口座（Checking Account）から普通預金口座（Savings Account）へ移動させるプログラムです。表示された画面上でユーザ ID と取引額（Transaction amount）を入力し，［Transfer］ボタンをクリックすると，指定したユーザの当座預金口座から指定した金額が普通預金口座に送金されます。資金はデータベースで管理され，資金移動時に当座預金口座テーブルと普通預金口座テーブルが更新されます。

このサンプルプロジェクト（Bank）の構成を次に示します。

図 A-1　サンプルプロジェクト（Bank）の概要



このサンプルプロジェクト（Bank）では，アノテーションを使用してデータベースに接続します。

指定したユーザ ID，金額に対して処理を実行します。「Bank」で指定できるユーザ ID とユーザごとの初期残高は次の表に示すとおりです。

表 A-1　サンプルプロジェクト（Bank）の初期設定

User ID	Checking Account※	Savings Account※
001	10,000	500
002	20,000	1,000
003	30,000	1,500

注※

表示される金額の単位は円です。

サンプルプロジェクトの EJB では，@Resource アノテーションでリソースアダプタを取得して，データベースにアクセスします。@Resource アノテーションの指定内容を次に示します。

- BankEJB.java

```
@Resource(mappedName="DB_Connector_for_Cosminexus_Driver")
private DataSource dataSource;
```

また，サーブレット（Web コンテナ）では，@EJB アノテーションで EJB を取得して，EJB にアクセスします。@EJB アノテーションの指定内容を次に示します。

- BankServlet.java

```
@EJB(name="BankEJB")
private BankIF bank ;
```

なお，@EJB アノテーションに指定している参照名「BankEJB」は，BankEJB のコードで次のように定義しています。

- BankEJB.java

```
@Stateful(name = "BankEJB")
```

このアノテーションは，EJB が Stateful Session Bean であることを表します。

(2)　サンプルプロジェクトの前提環境

付録 A では，次の環境を前提として，サンプルプロジェクト（Bank）の使用手順を説明します。

- **Eclipse がセットアップされている環境**

  付録 A では，Eclipse のセットアップが正しく行われている環境を前提としています。Eclipse のセットアップについては「2.4　Eclipse セットアップ機能を使用したセットアップ」を参照してください。

- **開発環境インスタントセットアップ機能で構築した環境**

付録 A では，開発環境インスタントセットアップ機能で構築した環境を前提としています。このため，次の条件に当てはまる論理サーバを使用します。

- アプリケーションサーバがローカルにインストールされている。
- 環境構築が正しく実施されている。
- J2EE サーバを開始する前に，J2EE サーバに関係する論理サーバをあらかじめ開始している。
- J2EE サーバモードが推奨モードである。

  V9 互換モードで実行したい場合は，サーバランタイムを Cosminexus J2EE V9 互換モードで登録し，Bank サンプルの各プロジェクト（Bank，Bank_EJB，Bank_Web）のサーバランタイムを Cosminexus J2EE V9 互換モードに変更してください。

- **JRE の設定**

  次に示す JDK を使用してください。

  - Developer の JDK（<Developer のインストールディレクトリ>¥jdk）

    JDK は，Eclipse の［設定］ダイアログの［インストール済みの JRE］ページで設定できます。JDK の確認方法については，「2.5.1　JDK の確認」を参照してください。

- **Management Server リモート管理機能の接続先**

  次に示すホストを指定してください。

  - localhost

    Management Server リモート管理機能の接続先は，Eclipse（WTP コネクタ）で設定します。設定方法は「4.2　リモート管理機能の設定」を参照してください。

- **使用するパースペクティブ**

  次に示すパースペクティブを使用します。ただし，パースペクティブの設定は，デフォルトから変更していないことを前提としています。

  - デバッグを行う場合：[デバッグ] パースペクティブ
  - デバッグ以外の場合：[J2EE] パースペクティブ

- **プロキシの設定**

  インターネットの接続にプロキシを使用している場合は，Eclipse のプロキシを設定してください。Eclipse のプロキシは，［設定］ダイアログの［ネットワーク接続］ページで設定できます。プロキシの設定方法については，「2.6.1　Eclipse のプロキシの設定」を参照してください。

- **リソースアダプタ**

  サンプルプロジェクト（Bank）では，開発環境インスタントセットアップ機能がデプロイしたリソースアダプタを使用します。なお，このリソースアダプタは，J2EE リソースアダプタとしてデプロイされます。

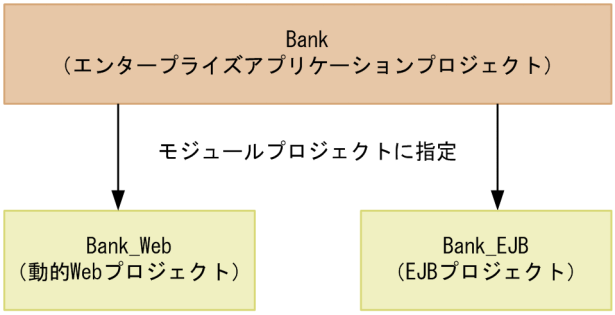
(3)　サンプルプロジェクトの構成

サンプルプロジェクト（Bank）では，次のプロジェクトを提供しています。

表 A-2　サンプルプロジェクト（Bank）が提供するプロジェクト

プロジェクト名	概要
Bank	エンタープライズアプリケーションプロジェクトです。
Bank_EJB	EJB プロジェクトです。サンプルプロジェクト（Bank）のモジュールプロジェクトに指定します。
Bank_Web	動的 Web プロジェクトです。サンプルプロジェクト（Bank）のモジュールプロジェクトに指定します。
BankDBBatch	シンプルプロジェクトです。サンプルプロジェクト（Bank）で使用するテーブルを作成するためのファイルが格納されています。

サンプルプロジェクト（Bank）が提供するプロジェクトの構成は，次のとおりです。



(4)　サンプルプロジェクトの提供形態と格納先

Developer は，サンプルプロジェクト（Bank）を zip フォーマットのアーカイブファイル「Bank.zip」として提供しています。Bank.zip は，次のディレクトリに格納されます。

```
<Developerのインストールディレクトリ>¥ADP¥samples
```

(5)　サンプルプロジェクトのディレクトリ構成

Bank.zip を任意の場所に解凍すると，次のディレクトリ構成になります。

表 A-3　サンプルプロジェクト（Bank）のディレクトリ構成

ディレクトリー／ファイル	説明
Bank	エンタープライズアプリケーションプロジェクトのルート
┣EarContent	コンテンツ・フォルダ
┃ ┣META-INF	管理情報を格納するディレクトリ
┃ ┗application.xml ┃	DD ファイル （J2EE アプリケーション（EAR）の配備記述子）
┗.project	プロジェクト記述ファイル （Eclipse のプロジェクトの情報を保存）

ディレクトリー／ファイル	説明
Bank_EJB	EJB プロジェクトのルート
├ejbModule	ソースファイル格納用ディレクトリ
│ ├bank	パッケージ・フォルダ
│ │ └ejb	パッケージ・フォルダ
│ │ ├BankEJB.java │ │ │	Java ソースファイル (EJB プロジェクト用のソースコード)
│ │ ├BankIF.java │ │ │	Java ソースファイル (EJB プロジェクト用のソースコード)
│ │ └BankItem.java │ │	Java ソースファイル (EJB プロジェクト用のソースコード)
│ └META-INF	管理情報を格納するディレクトリ
│ └MANIFEST.MF	マニフェストファイル
├.classpath │	クラスパスファイル (プロジェクトのクラスパスを保存)
└.project	プロジェクト記述ファイル (Eclipse のプロジェクトの情報を保存)
Bank_Web	Web プロジェクトのルート
├src	ソースファイル格納用のディレクトリ
│ └bank	パッケージ・フォルダ
│ └servlet	パッケージ・フォルダ
│ ├BankServlet.java	Java ソースファイル
│ └BankServletException.java	Java ソースファイル
├WebContent	Web ルートフォルダ
│ ├META-INF	管理情報を格納するディレクトリ
│ │ └MANIFEST.MF	マニフェストファイル
│ ├WEB-INF │ │ │	Web クライアントから直接アクセスできないファイルを格納するディレクトリ
│ │ ├lib	ライブラリ格納用ディレクトリ
│ │ └web.xml │ │	DD ファイル (Web アプリケーションの配備記述子)
│ ├bank.jsp │ │	JSP ファイル（ID と数値を入力して，Servlet に送信する）
│ ├error_500.jsp	JSP ファイル（エラーを表示する）

ディレクトリー／ファイル	説明
┃ ┗index.jsp ┃	JSP ファイル（サンプルプロジェクトの入り口となり，Servlet を呼び出す）
┣.classpath ┃	クラスパスファイル （プロジェクトのクラスパスを保存）
┗.project	プロジェクト記述ファイル （Eclipse のプロジェクトの情報を保存）
BankDBBatch ┃	サンプルプロジェクトのルート
┣bank_tblcreate.bat ┃	バッチファイル （テーブル作成バッチファイル）
┣Insert_BankTable_ ┃ Control_checking	コントロールファイル （checking テーブルのコントロールファイル）
┣Insert_BankTable_ ┃ Control_saving	コントロールファイル （saving テーブルのコントロールファイル）
┣Insert_BankTable_ ┃ Input_checking	データファイル （checking テーブルのデータファイル）
┣Insert_BankTable_ ┃ Input_saving	データファイル （saving テーブルのデータファイル）
┣tablecreate ┃	SQL ファイル （テーブルのスキーマ定義 SQL ファイル）
┗.project	プロジェクト記述ファイル （Eclipse のプロジェクトの情報を保存）

付録 A.2　サンプルプロジェクトの実行手順

Developer が提供するサンプルプロジェクトは，Eclipse の J2EE アプリケーション開発のチュートリアルとして使用できます。チュートリアルが対象とする機能は，次のとおりです。

1. Eclipse の起動
2. サーバランタイムの作成※
3. サンプルプロジェクト（Bank）のインポート
4. テーブルの作成
5. J2EE サーバの作成※
6. サンプルプロジェクト（Bank）の実行

注※

Eclipse を起動した際，［サーバー］ビューに［InstantJ2EEServer at localhost］が表示されている場合，作業は不要です。

(1) Eclipse の起動

Eclipse のインストールディレクトリにある eclipse.exe を実行して，Eclipse を起動します。

注意事項

管理者特権で Eclipse を起動してください。

(2) サーバランタイムの作成

Eclipse で J2EE サーバを操作するためにサーバランタイムを作成します。

なお，すでに［サーバー］ビューに［InstantJ2EEServer at localhost］が表示されている場合，作業は不要です。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます

2. ［設定］ダイアログの左ペインで，［サーバー］－［ランタイム環境］を選択します。
   ［サーバー・ランタイム環境］ページが表示されます。

3. ［追加］ボタンをクリックします。
   ［新規サーバー・ランタイム］ダイアログが表示されます。

4. ［新規サーバー・ランタイム］ページで，［Cosminexus］－［Cosminexus J2EE］を選択し，［終了］ボタンをクリックします。
   Eclipse で J2EE サーバを操作するためのサーバランタイムが作成されます。

(3) サンプルプロジェクト（Bank）のインポート

サンプルプロジェクト（Bank）を Eclipse のワークスペースにインポートします。

1. Eclipse のメニューから［ファイル］－［インポート］を選択します。
   ［インポート］ダイアログが表示されます。

2. ［インポート］ダイアログの［インポート・ソースの選択］で，［一般］－［既存プロジェクトをワークスペースへ］を選択し，［次へ］ボタンをクリックします。
   ［プロジェクトのインポート］ページが表示されます。

3. ［アーカイブ・ファイルの選択］をチェックし，［参照］ボタンをクリックします。

［インポートするプロジェクトを含むアーカイブの選択］ダイアログが表示されます。

4. **［インポートするプロジェクトを含むアーカイブの選択］ダイアログで，次のファイルを選択して，［開く］ボタンをクリックします。**

```
<Developerのインストールディレクトリ>¥ADP¥samples¥Bank.zip
```

5. **［プロジェクトのインポート］ページの［プロジェクト］エリア内にある項目をすべてチェックして，［終了］ボタンをクリックします。**

サンプルプロジェクト（Bank）が Eclipse のワークスペースにインポートされます。

注意事項

サーバランタイムの作成で，J2EE サーバー・ランタイム名をデフォルトから変更した場合，プロジェクトのターゲット・ランタイムを変更する必要があります。

1. **［プロジェクト・エクスプローラー］ビューで Bank プロジェクトを選択します。**
2. **Eclipse のメニューから［プロジェクト］－［プロパティ］を選択します。**
   ［プロパティ： <プロジェクト名>］ダイアログが表示されます。
3. **［プロパティ： <プロジェクト名>］ダイアログの左ペインで［ターゲット・ランタイム］を選択します。**
   ［ターゲット・ランタイム］ページが表示されます。
4. **作成したターゲット・ランタイムのチェックボックスを選択し，［OK］ボタンをクリックします。**
   ターゲット・ランタイムが変更されます。

(4) テーブルの作成

開発環境インスタントセットアップ機能で構築した組み込みデータベースに対して，サンプルプロジェクト（Bank）で使用するテーブルを作成します。次の手順で作成してください。

なお，テーブルを作成する前に，次の状態であることを確認してください。

- **組み込みデータベースを開始していること。**
  組み込みデータベースの操作方法については，HiRDB のマニュアルを参照してください。
- **Cosminexus サーバが停止状態であること。**
  ［サーバー］ビューで［InstantJ2EEServer at localhost］が「停止」になっていることを確認してください。［InstantJ2EEServer at localhost］の状態が実行中の場合は，［InstantJ2EEServer at localhost］を選択してビューツールバーの［ （停止）］をクリックして，Cosminexus サーバを停止してください。

1. **［プロジェクト・エクスプローラー］ビューで，［BankDBBatch］－［bank_tblcreate.bat］を選択し，コンテキストメニューから［アプリケーションから開く］－［テキストエディタ］を選択します。**

bank_tblcreate.bat ファイルが Eclipse のテキストエディタで開かれます。

2. bank_tblcreate.bat ファイルに指定されている認可識別子およびパスワードを開発環境インスタントセットアップ機能で設定した値に修正し，保存します。

   認可識別子およびパスワードは，bank_tblcreate.bat ファイルの 5，6 行目に指定されています。次に示すとおり修正してください。

行番号	修正前	修正後
5	set USER="USER1"	set USER="<開発環境インスタントセットアップ機能で指定した認可識別子>"
6	set PSWD=	set PSWD="<開発環境インスタントセットアップ機能で指定したパスワード>"

3. 保存した bank_tblcreate.bat ファイルを［プロジェクト・エクスプローラー］ビューで選択し，コンテキストメニューから［アプリケーションから開く］－［デフォルト・エディター］を選択します。

   bank_tblcreate.bat ファイルの内容が実行され，サンプルプロジェクト（Bank）用のテーブルが作成されます。

> **参考**
>
> テーブルの削除方法について
>
> テーブルが不要になった場合は，HiRDB SQL Executer などを使用して，手動でテーブルを削除してください。

(5)　J2EE サーバの作成

Eclipse で J2EE サーバを作成します。

なお，すでに［サーバー］ビューに［InstantJ2EEServer at localhost］が表示されている場合，作業は不要です。

1. Eclipse のメニューから［ファイル］－［新規］－［その他］を選択します。
2. ［新規］ダイアログで，［サーバー］－［サーバー］を選択し，［次へ］ボタンをクリックします。
3. ［新規サーバー］ダイアログで，［Cosminexus］－［J2EE サーバー］を選択し，［次へ］ボタンをクリックします。
4. ［J2EE サーバー］ページで，［リモート管理機能］の［接続ホスト］に接続先のホストを選択します。ログインしていない場合は，［ログイン］ボタンをクリックしてログインします。
5. ［InstantJ2EEServer］をチェックして，［終了］ボタンをクリックします。

   J2EE サーバが作成されます。

(6) サンプルプロジェクト（Bank）の実行

サンプルプロジェクト（Bank）を実行します。

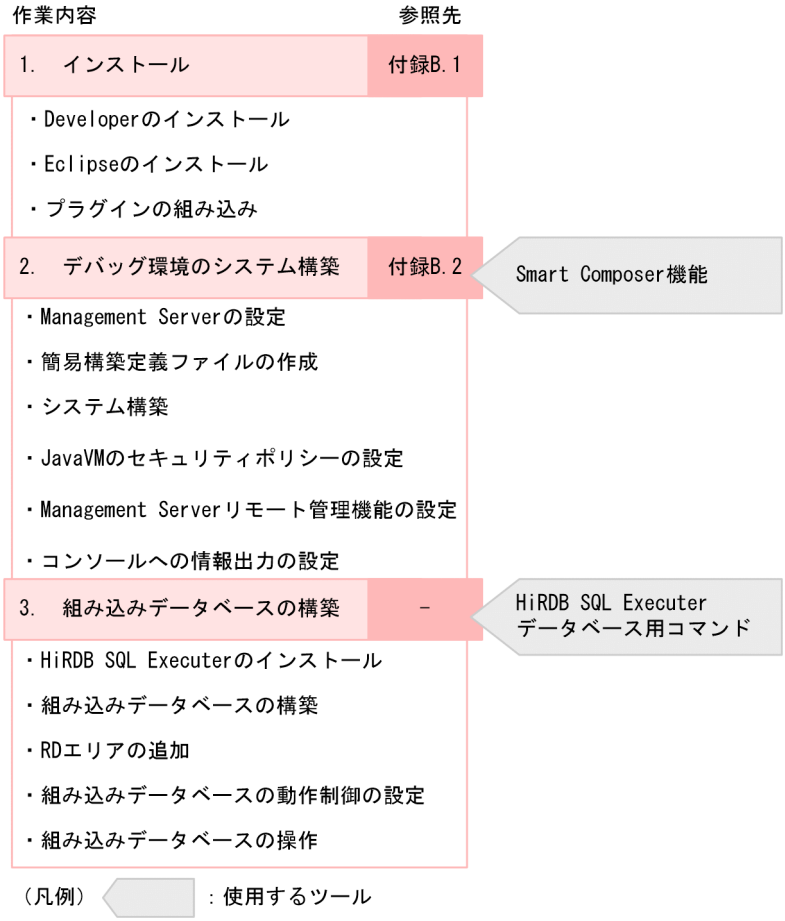
1. [プロジェクト・エクスプローラー] ビューで，Bank プロジェクトを選択して，コンテキストメニューから［実行］－［サーバーで実行］または［デバッグ］－［サーバーでデバッグ］を選択します。

2. [サーバーで実行] または［サーバーでデバッグ］ダイアログで，[localhost] － [InstantJ2EEServer at localhost] を選択し，［終了］ボタンをクリックします。
   J2EE サーバの起動とプロジェクトの公開が完了し，Web ブラウザが表示されます。

付録 B　開発環境インスタントセットアップ機能および Eclipse セットアップ機能を使用しないデバッグ環境の構築手順

ここでは，Developer が提供する開発環境インスタントセットアップ機能および Eclipse セットアップ機能を使用しないで，Eclipse を使用するときのデバッグ環境を構築する手順を説明します。構築の流れを次の図に示します。

図 B-1　デバッグ環境の構築手順（開発環境インスタントセットアップ機能および Eclipse セットアップ機能を使用しない場合）



図中の参照先を示します。

作業内容	参照先マニュアル	参照個所
1.　インストール	このマニュアル	付録 B.1
2.　デバッグ環境のシステム構築		付録 B.2
3.　組み込みデータベースの構築	HiRDB のマニュアル	-

環境構築後の作業の流れについては，「1.5.2　J2EE アプリケーションの開発手順」を参照してください。なお，J2EE アプリケーションにリソースアダプタを含めている場合は，リソースアダプタのプロパティを

設定します。リソースアダプタのプロパティの設定については，「付録 B.3　リソースアダプタのプロパティ設定」を参照してください。

付録 B.1　インストール

J2EE アプリケーション開発環境で使用するプログラムをインストールします。プログラムのインストール順序を次に示します。

1. Developer のインストール
   インストーラを使用して，Developer をインストールします。インストール手順については，「2.2.1 Developer のインストール」を参照してください。

2. Eclipse のインストール
   Eclipse のプログラムをダウンロードして，インストールします。Eclipse のインストール手順については，「(1)　Eclipse のインストール」を参照してください。

3. Developer が提供する Eclipse プラグインのセットアップ
   Eclipse プラグインをセットアップします。Eclipse プラグインのセットアップ手順については，「(2) Developer が提供する Eclipse プラグインのセットアップ」を参照してください。

ここでは，Eclipse のインストール手順，およびプラグインの組み込み方法について説明します。なお，プログラムのアンインストールについても説明します。アンインストールの手順については，「付録 B.4 アンインストール」を参照してください。

(1)　Eclipse のインストール

ここでは，Eclipse のインストールについて説明します。なお，Developer をアップグレードインストールした場合で，以前のバージョンとは異なる Eclipse を使用するときの手順についても説明します。

(a)　Eclipse のインストールの手順

J2EE アプリケーションの開発，実行時に使用する Eclipse をインストールします。インストール手順を次に示します。

1. Eclipse をインストールするためのディレクトリを作成します。
   Eclipse をインストールするディレクトリ，および Eclipse のインストールのために一時的に使用するディレクトリ（作業用ディレクトリ）を作成します。

2. Eclipse のアーカイブファイルを入手します。

Developer に同梱されている添付品または Eclipse.org のダウンロードサイトから，サポートしているバージョンを入手してください。

注意事項

サポートする Eclipse のバージョンは，Developer でインストールした JDK のバージョンによって異なります。

参考

アーカイブファイルは，Developer に同梱されている添付品のアーカイブファイルを使用することもできます。これを使用する場合は，ダウンロードサイトからの入手は不要です。

3. Eclipse のアーカイブファイルを解凍します。

入手した Eclipse のアーカイブファイルを手順 1.で作成した作業用ディレクトリで解凍します。

4. 解凍した Eclipse のファイルをコピーします。

手順 3.で解凍した eclipse フォルダを手順 1.で作成した Eclipse のインストールディレクトリにコピーします。

5. 作業用ディレクトリを削除します。

参考

ランゲージパックを適用する場合は，ダウンロードしたバージョンに合わせて準備してください。

(b)　旧バージョンで使用していた Eclipse とは異なるバージョンの Eclipse を使用する場合

バージョンアップ時に，旧バージョンで使用していた Eclipse とは異なるバージョンの Eclipse を使用する場合，Windows のエクスプローラから Eclipse のインストールディレクトリを削除してから，Eclipse をインストールします。

(2)　Developer が提供する Eclipse プラグインのセットアップ

Developer が提供する Eclipse プラグインのセットアップ手順を次に示します。

1. リンクファイルをコピーして貼り付けます。

コピー元のファイルは次の 2 種類です。

- <Developerのインストールディレクトリ>¥common¥dropins¥com.cosminexus.common.plugin.link
- <Developerのインストールディレクトリ>¥plugins¥dropins¥com.cosminexus.plugin.link

コピー先のフォルダを次に示します。

<Eclipseのインストールディレクトリ>¥eclipse¥dropins

2. <Eclipse のインストールディレクトリ>¥eclipse 直下にある eclipse.ini をコピーして，任意の場所に退避します。

   Eclipse を Developer インストール前の状態に戻すために使用します。

3. <Eclipse のインストールディレクトリ>¥eclipse 直下の eclipse.ini を編集します。

   eclipse.ini の編集例を次に示します。背景色付きの太字部分を追加してください。

```
-startup
plugins/org.eclipse.equinox.launcher_1.6.300.v20210813-1054.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.2.300.v20210828-0802
-product
org.eclipse.epp.package.jee.product
-showsplash
org.eclipse.epp.package.common
--launcher.defaultAction
openFile
--launcher.defaultAction
openFile
--launcher.appendVmargs
-vm
<Developerのインストールディレクトリ>¥jdk¥bin¥javaw.exe
-vmargs
-Dosgi.requiredJavaVersion=11
-Dosgi.instance.area.default=@user.home/eclipse-workspace
-Dsun.java.command=Eclipse
-XX:+UseG1GC
-XX:+UseStringDeduplication
-Dosgi.configuration.area=@user.home/ADP/eclipse/configuration※
--add-modules=ALL-SYSTEM
-Dosgi.requiredJavaVersion=11
-Dosgi.dataAreaRequiresExplicitInit=true
-Dorg.eclipse.swt.graphics.Resource.reportNonDisposed=true
-Xms256m
-Xmx2048m
--add-modules=ALL-SYSTEM
```

   注※

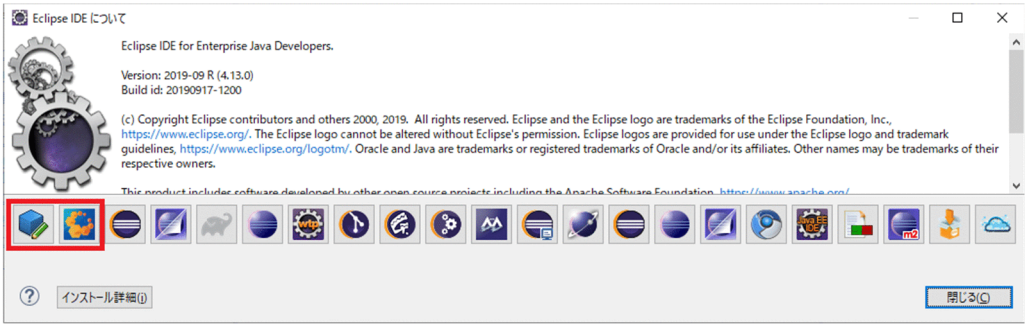
   Eclipse を共有使用する場合に追加してください。

4. -clean オプションを指定して Eclipse を起動します。

```
<Eclipseのインストールディレクトリ>¥eclipse¥eclipse.exe -clean
```

5. Eclipse のメニューから［ヘルプ］－［Eclipse について］を選択します。

   ［Eclipse について］ダイアログに，枠で示すアイコンが表示されていることを確認してください。



6. 確認できたら［OK］ボタンをクリックします。

参考

- セットアップした Eclipse を起動し，［Eclipse について］ダイアログを表示すると，枠で示すアイコンが表示されない場合があります。この場合は，Eclipse を再起動すると表示されます。
- この節の手順によって，次の Eclipse プラグインがセットアップされます。

  Application Development Plug-in

  Sharing Library Plug-in

付録 B.2　デバッグ環境のシステム構築

J2EE アプリケーションのテストで使用するシステムを構築します。

システムの構築には，Smart Composer 機能を使用します。Smart Composer 機能とは，システムの構築を支援するための機能です。システムの情報を定義したファイルを作成して Smart Composer 機能のコマンドを実行すると，簡単にシステムを構築できます。

Smart Composer 機能を使用したシステム構築の流れを次に示します。

1. Management Server の設定

   Management Server とは，アプリケーションサーバを一括で管理および運用するための運用管理プロセスです。

   Management Server をセットアップして，Management Server の管理ユーザを設定します。また，Management Server および運用管理エージェントを自動起動するための設定もします。設定方法については，マニュアル「アプリケーションサーバ システム構築・運用ガイド」の「4.1.14　運用管理機能を構築する」を参照してください。

   なお，Management Server および運用管理エージェントの概要については，マニュアル「アプリケーションサーバ システム設計ガイド」の「2.3.1　プロセス構成」を参照してください。

2. 簡易構築定義ファイルの作成

構築するシステムの情報を，簡易構築定義ファイルに定義します。簡易構築定義ファイルには，構築するシステムの名称やシステムに配置する論理サーバの種類などを定義します。また，J2EE アプリケーションのデバッグ環境ではリロード機能を使用します。リロード機能を使用するための設定も簡易構築定義ファイルに定義します。作成方法については，「(1)　簡易構築定義ファイルの作成」を参照してください。

3. システムの構築

Smart Composer 機能で提供するコマンドを使用して，システムを構築します。Smart Composer 機能で使用するコマンドについては，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「8.　Smart Composer 機能で使用するコマンド」を参照してください。

4. JavaVM のセキュリティポリシーの設定

セキュリティマネージャを使用する場合は，JavaVM のセキュリティポリシーを指定できます。JavaVM のセキュリティポリシーを指定する場合は，あらかじめ設定しておきます。セキュリティマネージャを使用しない場合はこの設定は不要です。設定方法については，「(2)　JavaVM のセキュリティポリシーの設定」を参照してください。

5. Management Server リモート管理機能の設定

WTP コネクタで Management Server リモート管理機能を使用するため，事前に Management Server リモート管理機能の設定が必要です。Management Server リモート管理機能の設定については，「(3)　Management Server リモート管理機能の設定」を参照してください。

6. コンソールへの情報出力の設定

コンソールに情報を出力する場合は，運用管理エージェントの設定が必要です。コンソールに情報出力をするための設定については，「(4)　コンソールへの情報出力の設定」を参照してください。

(1)　簡易構築定義ファイルの作成

構築するシステムの情報を定義した簡易構築定義ファイルを作成します。

ここでは，Developer で提供しているテンプレートファイル（cmxdefcombinedmodel.xml）を使用して，簡易構築定義ファイルを作成します。簡易構築定義ファイルのテンプレートは次の場所に格納されています。テンプレートファイルを任意の場所にコピーして，簡易構築定義ファイルを作成してください。

テンプレートファイル（cmxdefcombinedmodel.xml）の格納先

```
<Developerのインストールディレクトリ>¥manager¥config¥templates¥
```

簡易構築定義ファイルで定義する内容については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「4.3　簡易構築定義ファイル」を参照してください。

なお，デバッグ環境をプログラミング環境とは別のマシンに構築している場合は，Management Server にアクセスするホストを許可する設定が必要になります。簡易構築定義ファイルの<param>タグの下に次のタグを設定します。

- <param-name>タグ
  「webserver.connector.http.permitted.hosts」を指定します。
- <param-value>タグ
  Management Server へのアクセスを許可するホスト名または IP アドレス（Developer がインストールされている開発環境のホスト名または IP アドレス）を指定します。

また，JSP デバッグを実施する場合は，簡易構築定義ファイルの<param>タグの下に次のタグを設定します。

- <param-name>タグ
  webserver.jsp.debugging.enabled
- <param-value>タグ
  true

(2) JavaVM のセキュリティポリシーの設定

JavaVM のセキュリティポリシーを設定するには，server.policy を編集します。server.policy の格納先を次に示します。

server.policy の格納先

<Developer のインストールディレクトリ>¥CC¥server¥usrconf¥ejb¥<サーバ名称>¥server.policy

server.policy に設定する内容については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「2.2.4　server.policy（J2EE サーバ用セキュリティポリシーファイル)」を参照してください。

ポイント

server.policy の格納先である<サーバ名称>フォルダ，および server.policy は，cmx_build_system コマンドを実行すると作成されます。

(3) Management Server リモート管理機能の設定

WTP コネクタを使用するためには，Management Server リモート管理機能の設定が必要です。Management Server リモート管理機能を使用するためには，mserver.properties（Management Server 環境設定ファイル）にプロパティを設定します。mserver.properties の格納先と追加するプロパティを次に示します。

- **格納先**
  <Developer のインストールディレクトリ>¥manager¥config¥mserver.properties
- **追加するプロパティ**

```
com.cosminexus.mngsvr.management.enabled=true
com.cosminexus.mngsvr.management.connector.enabled=true
com.cosminexus.mngsvr.management.port=28099※
```

注※

任意のポート番号を指定してください。デフォルトは 28099 です。

なお，Management Server 実行中にプロパティを設定しても反映されません。設定を反映するには，Management Server を再起動する必要があります。

(4) コンソールへの情報出力の設定

コンソールに情報を出力する場合は，adminagent.properties（運用管理エージェントプロパティファイル）に次の設定を追加してください。

- 格納先

  <Developer のインストールディレクトリ>¥manager¥config¥adminagent.properties

- 追加するプロパティ

```
adminagent.j2ee.process.console_event.enabled=true
```

注

adminagent.properties には，上記のプロパティがすでに記載されています。
adminagent.properties に記載されているプロパティを使用する場合は，次の内容を実施してください。

・コメントアウトを解除します（#を削除します）。

・プロパティの値を，false から true に変更します。

なお，Management Server 実行中にプロパティを設定しても反映されません。設定を反映するには，Management Server および運用管理エージェントを再起動する必要があります。

ポイント

ここで設定しているのは，コンソールに情報を出力するかどうかです。コンソールに表示する文字数などの設定については，「2.6.3　コンソールの設定変更」を参照してください。

付録 B.3　リソースアダプタのプロパティ設定

インポートしたリソースアダプタを J2EE アプリケーションで使用するためには，リソースアダプタのプロパティを設定する必要があります。リソースアダプタのプロパティ設定には，運用管理ポータルを使用します。リソースアダプタのプロパティ設定については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「12.4.5　リソースアダプタのプロパティ設定」を参照してください。

付録 B.4　アンインストール

J2EE アプリケーション開発環境で使用したプログラムのアンインストールについて説明します。アンインストールは次の順序で実施してください。

1. Eclipse のアンセットアップ
2. Developer のアンインストール

ここでは，Eclipse をアンセットアップする方法について説明します。Developer のアンインストールについては，「2.9　Developer のアンインストール」を参照してください。

(1)　Eclipse のアンセットアップ

Eclipse のアンセットアップ手順を次に示します。

1. Eclipse を終了します。
2. 次のリンクファイルを削除します。
   - <Eclipseのインストールディレクトリ>¥eclipse¥dropins¥com.cosminexus.common.plugin.link
   - <Eclipseのインストールディレクトリ>¥eclipse¥dropins¥com.cosminexus.plugin.link
3. ほかのプラグインを同時にアンセットアップする場合は，ほかのプラグインのリンクファイルも削除します。
4. Eclipse を引き続き使用する場合は，退避しておいた eclipse.ini を<Eclipse のインストールディレクトリ>¥eclipse 直下に戻します。
5. -clean オプションを指定して Eclipse を起動します。

```
<Eclipseのインストールディレクトリ>¥eclipse¥eclipse.exe -clean
```

付録 C　開発環境インスタントセットアップ実行時の設定値

開発環境インスタントセットアップ機能を使用してデバッグ環境のカスタムセットアップをすると，表 C-1 から表 C-6 に示すように，デフォルト値ではない値が設定されるキーがあります。

表 C-1　adminagent.properties（運用管理エージェントプロパティファイル）の設定値

項番	キー名称	設定される値	設定内容
1	adminagent.hws.sys_cmd.abnormal_end.traceinfo	false	障害検知時コマンドの実行の際に Web サーバの内部トレースを採取するかどうかが設定されます。
2	adminagent.j2ee.process.console_event.enabled	true	J2EE サーバのコンソール出力情報を ManagementServer を使用する Eclipse プラグインで表示するかどうかが設定されます。
3	adminagent.j2ee.sys_cmd.abnormal_end.javatrace	false	障害検知時コマンドの実行の際に J2EE サーバのスタックトレースを取得するかどうかが設定されます。
4	adminagent.j2ee.sys_cmd.abnormal_end.threaddump	false	障害検知時コマンド実行の際に J2EE サーバのスレッドダンプを収集するかどうかが設定されます。
5	adminagent.j2ee.watch.start_time	7	J2EE サーバの起動で，起動コマンドを実行してから動作確認を開始するまでの時間（単位：秒）が設定されます。
6	adminagent.sys_cmd.abnormal_end.prftrace	false	障害検知時コマンド実行の際に性能解析トレースファイルを収集するかどうかが設定されます。

表 C-2　mserver.properties（Management Server 環境設定ファイル）の設定値

項番	キー名称	設定される値	設定内容
1	adminagent.connector.comm.state.cache_max_time	-1	運用管理エージェントとの通信状態をキャッシュする最大時間（単位：秒）が設定されます。

項番	キー名称	設定される値	設定内容
2	com.cosminexus.mngsvr.management.connector.enabled	true	Management Server リモート管理機能への外部接続を有効にするかどうかが設定されます。
3	com.cosminexus.mngsvr.management.enabled	true	Management Server リモート管理機能を有効にするかどうかが設定されます。
4	com.cosminexus.mngsvr.management_user_account.enabled	false	Management Server のユーザアカウントを有効にするかどうかが設定されます。
5	com.cosminexus.mngsvr.snapshot.auto_collect.enabled	false	障害発生時または一括再起動時に snapshot ログを収集するかどうかが設定されます。
6	com.cosminexus.mngsvr.sys_cmd.abnormal_end.enabled	false	システムによる障害検知時コマンド実行機能を利用するかどうかが設定されます。
7	webserver.connector.http.permitted.hosts	localhost	Management Server へのアクセスを許可するホストおよび運用管理エージェント稼働ホストの IP アドレスまたはホスト名が設定されます。

表 C-3　mserver.cfg（Management Server 用オプション定義ファイル）の設定値

項番	キー名称	設定される値	設定内容
1	add.jvm.arg	-Xms256m -Xmx512m -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=128m -XX:CompressedClassSpaceSize=128m	指定されたオプションを使って JavaVM を起動します。

表 C-4　usrconf.cfg（Java アプリケーション用オプション定義ファイル）の設定値

項番	キー名称	設定される値	設定内容
1	add.class.path	<Developer のインストールディレクトリ>¥DB¥CLIENT¥UTL¥<HiRDB Type4 JDBC Driver の JAR ファイル>	クラスパスに追加する値が設定されます。
2	add.jvm.arg	-agentlib:jdwp=transport=dt_socket,server=y,address=*:3999,suspend=n -Xms256m -Xmx512m	指定されたオプションを使って JavaVM を起動します。

表 C-5　usrconf.properties（J2EE サーバ用ユーザプロパティファイル）の設定値

項番	キー名称	設定される値	設定内容
1	ejbserver.deploy.context.check_interval	1	アプリケーション構成ファイルの更新を検知する間隔（単位：秒）が設定されます。
2	ejbserver.logger.systemlog.enabled	false	J2EE サーバの起動，停止および異常終了に関するメッセージを，イベントログへ出力するかどうかが設定されます。
3	ejbserver.management.JVM.stats_monitor.FullGCCount.enabled	false	FullGC 回数の監視を有効にするかどうかが設定されます。
4	ejbserver.management.stats_file.enabled	false	稼働情報ファイル出力機能を有効にするかどうかが設定されます。
5	ejbserver.watch.defaultRequestQueue.enabled	false	デフォルトの実行待ちキューの場合の HTTP リクエスト実行待ちキュー監視のアラート出力を有効にするかどうかが設定されます。
6	ejbserver.watch.defaultRequestQueue.writefile.enabled	false	デフォルトの実行待ちキューの場合の HTTP リクエスト実行待ちキュー監視結果をファイル出力するかどうかが設定されます。
7	ejbserver.watch.enabled	false	すべてのリソース枯渇監視を有効にするかどうかが設定されます。
8	ejbserver.watch.fileDescriptor.enabled	false	ファイルディスクリプタ監視のアラート出力を有効にするかどうかが設定されます。
9	ejbserver.watch.fileDescriptor.writefile.enabled	false	ファイルディスクリプタ監視結果をファイル出力するかどうかが設定されます。
10	ejbserver.watch.memory.enabled	false	メモリ監視のアラート出力を有効にするかどうかが設定されます。
11	ejbserver.watch.memory.writefile.enabled	false	メモリ監視結果をファイル出力するかどうかが設定されます。
12	ejbserver.watch.thread.enabled	false	スレッド監視のアラート出力を有効にするかどうかが設定されます。

項番	キー名称	設定される値	設定内容
13	ejbserver.watch.thread.writefile.enabled	false	スレッド監視結果をファイル出力するかどうかが設定されます。
14	ejbserver.watch.threaddump.enabled	false	スレッドダンプ監視のアラート出力を有効にするかどうかが設定されます。
15	ejbserver.watch.threaddump.writefile.enabled	false	スレッドダンプ監視結果をファイル出力するかどうかが設定されます。
16	webserver.connector.http.permitted.hosts	*	Management Server へのアクセスを許可するホストの IP アドレスまたはホスト名が設定されます。
17	webserver.connector.inprocess_http.enabled	true	「J2EE サーバモード」が「V9 互換モード」の場合だけ設定します。インプロセス HTTP サーバを使用するかどうかが設定されます。
18	webserver.connector.inprocess_http.init_threads	2	「J2EE サーバモード」が「V9 互換モード」の場合だけ設定します。サーバ起動時に生成するインプロセス HTTP サーバのリクエスト処理スレッド数（1～1024）が設定されます。
19	webserver.errorpage.stack_trace.enabled	true	例外発生時，デフォルトのエラーページにスタックトレースを出力するかどうかが設定されます。
20	webserver.jsp.debugging.enabled	true	JSP デバッグ機能を有効にするかどうかが設定されます。
21	webserver.static_content.cache.filesize.threshold	512	静的コンテンツキャッシュ機能が有効な場合，キャッシュできるファイルサイズ（単位：秒）が設定されます。
22	webserver.static_content.cache.size	1024	静的コンテンツキャッシュ機能が有効な場合，メモリにキャッシュできるサイズ（単位：秒）の上限が設定されます。

表 C-6　論理サーバ共通パラメタの設定値

項番	キー名称	設定される値	設定内容
1	additional.startcmd	-nosecurity	起動コマンドに追加するオプションが設定されます。

注　論理サーバ起動時に SecurityManager を解除して起動します。

付録 D　デバッグ環境のシステムのチューニング

構築したテスト用のシステムのパラメタをチューニングする場合は，運用管理ポータルという Management Server を操作するための GUI を使用できます。運用管理ポータルを使用するためには，Management Server のセットアップ，および Management Server および運用管理エージェントの起動が必要です。

ただし，これらの設定は Management Server 設定時に実施済みのため，ここで特に設定する必要はありません。

- **運用管理ポータルへのログイン**

  運用管理ポータルへは，Web ブラウザからログインします。Management Server を起動しているホスト上，または Management Server を起動しているホストとネットワークで接続されているホスト上で Web ブラウザを起動して，URL に「http://<ホスト名>:<ポート番号>/mngsvr/」を指定します。インストール初期状態での URL は「http://localhost:28080/mngsvr/」です。

  運用管理ポータルを起動すると，ログイン画面が表示されます。ログイン画面で入力する管理ユーザ ID とパスワードには，次の値を入力してください。

  - 管理ユーザ ID

    Management Server 設定時に指定した「管理ユーザ ID」を入力します。

  - パスワード

    Management Server 設定時に指定した「管理ユーザパスワード」を入力します。

    ［ログイン］ボタンをクリックすると，運用管理ポータル画面が表示されます。

- **運用管理ポータルでの設定変更**

  運用管理ポータルにログインすると，［運用管理ポータル］画面が表示されます。システムのプロパティは，［運用管理ポータル］画面の［論理サーバの環境設定］アンカーをクリックして表示される「論理サーバの環境設定」の各画面で変更します。論理サーバごとに設定する画面が異なります。

  なお，運用管理ポータルの画面と各画面で設定できる内容については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「7.2　運用管理ポータルの画面構成」を参照してください。

付録 E　障害時に必要となる情報の採取方法

Eclipse 使用時に発生した障害を解決するために必要な情報の採取方法を説明します。また，開発環境インスタントセットアップ機能を実行したときに出力されるログの確認方法についても説明します。

付録 E.1　構成情報の採取

Eclipse の構成情報の採取方法を説明します。次の手順で採取できます。

1. Eclipse のメニューから［ヘルプ］－［Eclipse について］を選択します。
   ［Eclipse について］ダイアログが表示されます。
2. ［インストール詳細］ボタンをクリックします。
   ［Eclipse のインストール詳細］ダイアログが表示されます。
3. ［構成］タブを選択します。
   構成情報が表示されます。
4. ［クリップボードにコピー］ボタンをクリックします。
   構成情報がクリップボードにコピーされます。コピーされた内容をテキストエディタなどに貼り付けて，内容を保存してください。

付録 E.2　エラーログの採取

Eclipse のエラーログの採取方法には，次の 2 とおりの方法があります。

1. ［Eclipse のインストール詳細］ダイアログの［構成］タブから採取する
2. エラーログファイルを直接採取する

それぞれの方法について，次に説明します。

(1)　［Eclipse のインストール詳細］ダイアログの［構成］タブから採取する

次の手順でエラーログを採取できます。

1. Eclipse のメニューから［ヘルプ］－［Eclipse について］を選択します。
   ［Eclipse について］ダイアログが表示されます。
2. ［インストール詳細］ボタンをクリックします。
   ［Eclipse のインストール詳細］ダイアログが表示されます。
3. ［構成］タブを選択します。

構成情報が表示されます。

4. [エラー・ログの表示] ボタンをクリックします。

エラーログの内容が Web ブラウザに表示されるので，内容を保存してください。

(2) エラーログファイルを直接採取する

エラーログファイルは，次に示すファイルに記録されます。このファイルを直接採取してください。なお，バックアップされたエラーログファイルがある場合は，それらも採取してください。

- エラーログファイル

  <Eclipseワークスペース・ディレクトリ>¥.metadata¥.log

- エラーログファイルのバックアップファイル

  <Eclipseワークスペース・ディレクトリ>¥.metadata¥.bak_N※.log

  注※

  N は，任意の数字です。

付録 E.3 トレースログの採取

Application Development Plug-in が提供する Eclipse プラグインのトレースログは，Eclipse ワークスペースディレクトリ下に出力されます。次の表に示すファイルを直接採取してください。

表 E-1 Eclipse のトレースログ

トレースログファイルのディレクトリ※1	出力されるファイル名
.metadata	com.cosminexus.adp.common.carver.traceN※2.log

注※1

ディレクトリ名に使用される文字列です。

注※2

N には，出力されたファイルの順に 0 以上の数字が入ります。

付録 E.4 開発環境インスタントセットアップ機能および Eclipse セットアップ機能実行時の情報の採取

開発環境インスタントセットアップ機能および Eclipse セットアップ機能を実行したときに設定した内容（セットアップ内容，設定変更内容，アンセットアップ内容）は，セットアップログとしてファイルに保存されます。

セットアップログの確認方法，およびセットアップログで確認できる内容を説明します。

(1) 開発環境インスタントセットアップ機能実行時の情報

- セットアップログの確認方法

  Windows のスタートメニューから，［Cosminexus］－［デバッグ環境セットアップログ］※を選択すると，セットアップログが表示されます。

  注※

  対象 OS のスタートメニューの表示仕様によっては，［Cosminexus］－［環境構築］－［デバッグ環境アンセットアップ］と表示される場合があります。

- セットアップログで確認できる内容

  デバッグ環境のセットアップログでは，次の内容を確認できます。

  - デバッグ環境をセットアップしたときの設定内容
  - デバッグ環境の設定変更の内容
  - アンセットアップしたデバッグ環境の設定内容

  これらの情報は，実行された順番に一つのファイルに格納されます。このため，最新の情報は，ファイルの最後に格納されています。なお，エラーまたは処理の中断が発生した場合でも，これらの情報は格納されます。

  なお，開発環境インスタントセットアップ機能実行時のエラーメッセージは，開発環境インスタントセットアップ機能のコンソールに出力されます。出力されたエラーメッセージについては，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」を参照して対処してください。

注

組み込みデータベースのセットアップ時にエラーが発生した場合は，マニュアル「HiRDB メッセージ」を参照してください。

(2) Eclipse セットアップ機能実行時の情報

- セットアップログの確認方法

  Windows のスタートメニューから，［Cosminexus］－［Eclipse セットアップログ］を選択すると，セットアップログが表示されます。

- セットアップログで確認できる内容

  Eclipse 環境のセットアップログでは，次の内容を確認できます。

  - Eclipse 環境をセットアップしたときの設定内容
  - アンセットアップした Eclipse 環境の設定内容

  これらの情報は，実行された順番に一つのファイルに格納されます。このため，最新の情報は，ファイルの最後に格納されています。なお，エラーまたは処理の中断が発生した場合でも，これらの情報は格納されます。

なお，Eclipse セットアップ機能実行時のエラーメッセージは，Eclipse セットアップ機能のコンソールに出力されます。出力されたエラーメッセージについては，マニュアル「アプリケーションサーバメッセージ(構築／運用／開発用)」を参照して対処してください。

付録 F　旧バージョンからの移行（WTP からの移行の場合）

付録 F.1　プロジェクトの移行

旧バージョンの WTP で作成した WTP プロジェクトを新バージョンの WTP に移行する手順を説明します。

(1)　旧バージョンで作成したワークスペースを使用する場合に必要な作業

旧バージョンで作成したワークスペースを使用する場合は，次の作業をしてください。

- J2EE サーバランタイムをいったん削除してから，作成し直します。
- ［設定］ダイアログの［インストール済みの JRE］ページで，設定されている Developer の JDK をいったん削除してから，再び追加します。

(2)　移行前の WTP の環境で必要な作業

エクスポートの手順を次に示します。なお，WTP プロジェクトのエクスポートは，WTP の機能を使用して実施します。

1. Eclipse のメニューから［ファイル］－［エクスポート］を選択します。
   ［エクスポート］ダイアログが表示されます。
2. ［一般］－［アーカイブ・ファイル］を選択し，［次へ］ボタンをクリックします。
   ［アーカイブ・ファイル］ページが表示されます。
3. エクスポート対象のリソースを選択して，アーカイブファイルを出力します。

(3)　移行後の WTP の環境で必要な作業

インポートの手順を次に示します。

1. Eclipse のメニューから［ファイル］－［インポート］を選択します。
   ［インポート］ダイアログが表示されます。
2. ［一般］－［既存プロジェクトをワークスペースへ］を選択し，［次へ］ボタンをクリックします。
   ［プロジェクトのインポート］ページで，プロジェクトをインポートします。
3. プロジェクトのターゲット・ランタイムに，Cosminexus J2EE サーバランタイムを指定します。
   Cosminexus J2EE サーバランタイムが存在しない場合は作成してください。サーバランタイムの作成については，「4.3　サーバランタイムの作成」を参照してください。

Cosminexus J2EE サーバランタイムが存在する場合は，プロジェクトを選択し，メニューの［プロジェクト］－［プロパティ］を選択します。プロパティダイアログで［ターゲット・ランタイム］を選択し，Cosminexus J2EE サーバランタイムをチェックして，［OK］ボタンをクリックします。

付録 G　旧バージョンからの移行（MyEclipse からの移行の場合）

Developer Version 8 で作成した MyEclipse のプロジェクトを新バージョンの WTP に移行する手順を説明します。

付録 G.1　MyEclipse で作成したプロジェクトの移行

WTP のプロジェクトへの移行の可否と，対応する WTP のプロジェクトについて次の表に示します。

表 G-1　移行前後のプロジェクトの対応

移行前のプロジェクト	対応する WTP のプロジェクト
エンタープライズアプリケーションプロジェクト	エンタープライズアプリケーションプロジェクト
Web プロジェクト	動的 Web プロジェクト
EJB プロジェクト	EJB プロジェクト

付録 G.2　移行対象となるリソース

移行前の環境で開発したリソースのうち移行対象となるリソースをプロジェクトごとに示します。

(1)　エンタープライズアプリケーションプロジェクトのリソースの移行

エンタープライズアプリケーションプロジェクトのリソースの移行について説明します。エンタープライズアプリケーションプロジェクトの構成例と，構成例にあるリソースごとの WTP への移行の可否を表に示します。なお，エンタープライズアプリケーションプロジェクトの構成例にある項番は，表中の項番と対応しています。

表 G-2　EAR プロジェクトの構成例（アーカイブ形式の場合）

構成例	表 G-3 の項番
<エンタープライズアプリケーションプロジェクトロケーションディレクトリ>	－
├lib	－
│ └*.jar	1
├META-INF	－
│ ├application.xml	2
│ └MANIFEST.MF	3
├.mymetadata	4

構成例	表 G-3 の項番
└.project	5

（凡例）－：該当なし

表 G-3　EAR プロジェクト（アーカイブ形式）のリソースと WTP への移行の可否

項番	リソースの種類	WTP への移行	移行先
1	ライブラリ	○	<エンタープライズアプリケーションプロジェクトロケーションディレクトリ>
2	DD ファイル※	△	<エンタープライズアプリケーションプロジェクトロケーションディレクトリ>¥<コンテンツディレクトリ>¥META-INF
3	マニフェストファイル	×	－
4	MyEclipse プロジェクトファイル	×	－
5	プロジェクト記述ファイル	×	－

（凡例）

○：移行できる　△：任意　×：移行できない　－：該当しない

注※

デプロイに必要なプロジェクト定義だけの場合は，移行の必要はありません。ほかに定義がある場合は，直接編集して移行します。

(2)　Web プロジェクトのリソースの移行

Web プロジェクトのリソースの移行について説明します。Web プロジェクトの構成例と，構成例にあるリソースごとの WTP への移行の可否を表に示します。なお，Web プロジェクトの構成例にある項番は，表中の項番と対応しています。

表 G-4　Web プロジェクトの構成例

構成例	表 G-5 の項番
<Web プロジェクトロケーションディレクトリ>	－
├.myeclipse	－
├src	－
│ └*.java	1
├WebRoot	－
│ ├META-INF	－
│ │ └MANIFEST.MF	2
│ ├WEB-IN	－
│ │ ├classes	－

構成例	表 G-5 の項番
｜｜ ├lib	－
｜｜ ｜ └*.jar	3
｜｜ └web.xml	4
｜ └*.html，*.jsp	5
├.classpath	6
├.mymetadata	7
└.project	8

（凡例）－：該当なし

表 G-5　Web プロジェクトのリソースと WTP への移行の可否

項番	リソースの種類	WTP への移行	移行先
1	Java ソースファイル	○	<動的 Web プロジェクトロケーションディレクトリ>¥src
2	マニフェストファイル	×	－
3	ライブラリ	○	<動的 Web プロジェクトロケーションディレクトリ>¥<Web コンテンツフォルダ>¥WEB-INF¥lib
4	DD ファイル	○	<動的 Web プロジェクトロケーションディレクトリ>¥<Web コンテンツフォルダ>¥WEB-INF
5	HTML ファイル，JSP ファイル	○	<動的 Web プロジェクトロケーションディレクトリ>¥<Web コンテンツフォルダ>
6	クラスパスファイル	×	－
7	MyEclipse プロジェクトファイル	×	－
8	プロジェクト記述ファイル	×	－

（凡例）

○：移行できる　△：任意　×：移行できない　－：該当しない

(3)　EJB プロジェクトのリソースの移行

EJB プロジェクトのリソースの移行について説明します。EJB プロジェクトの構成例と，構成例にあるリソースごとの WTP への移行の可否を表に示します。なお，EJB プロジェクトの構成例にある項番は，表中の項番と対応しています。

表 G-6　EJB プロジェクトの構成例

構成例	表 G-7 の項番
<プロジェクトディレクトリ>	－

構成例	表 G-7 の項番
├ .myeclipse	−
├ classes	−
│　├ *.class	1
│　└ META-INF	−
│　└ MANIFEST.MF	2
├ src	−
│ ├ META-INF	−
│ │　└ MANIFEST.MF	3
│ └ *.java	4
├ .classpath	5
├ .mymetadata	6
└ .project	7

（凡例）−：該当なし

表 G-7　EJB プロジェクトのリソースと WTP への移行の可否

項番	リソースの種類	WTP への移行	移行先
1	クラスファイル	×	−
2	マニフェストファイル	×	−
3	マニフェストファイル	×	−
4	Java ソースファイル	○	<EJB プロジェクトロケーションディレクトリ>¥<ソースフォルダ>
5	クラスパスファイル	×	−
6	MyEclipse プロジェクトファイル	×	−
7	プロジェクト記述ファイル	×	−

（凡例）

○：移行できる　×：移行できない　−：該当しない

付録 G.3　WTP プロジェクトへの移行手順

プロジェクトを移行する流れを説明します。

1. EJB プロジェクトを移行します。

WTP で EJB プロジェクトを作成します。プロジェクト作成後，MyEclipse の EJB プロジェクトのリソースを移行します。

2. Web プロジェクトを移行します。
   WTP で動的 Web プロジェクトを作成します。プロジェクト作成後，MyEclipse の Web プロジェクトのリソースを移行します。

3. エンタープライズアプリケーションプロジェクトを移行します。
   WTP でエンタープライズアプリケーションプロジェクトを作成します。プロジェクト作成後，MyEclipse のエンタープライズアプリケーションプロジェクトのリソースを移行します。

付録 G.4　EJB プロジェクトの移行

MyEclipse の EJB プロジェクトを WTP に移行するには，WTP でプロジェクトを作成してから，移行したい EJB プロジェクトのリソースを移行します。移行手順について次に説明します。

(1)　EJB プロジェクトの作成

WTP で EJB プロジェクトを作成します。EJB プロジェクトの作成手順については，「4.4.2　EJB プロジェクトの作成」を参照してください。

WTP に移行するための，EJB プロジェクト作成時の注意を次に示します。

- EJB プロジェクトが複数ある場合は，プロジェクトの数だけ EJB プロジェクトを作成します。
- WTP の［EJB Project］ページで指定する［プロジェクト名］には，EJB プロジェクトのプロジェクト名を指定してください。

(2)　EJB プロジェクトのリソースの移行

EJB プロジェクトのリソースを移行します。移行できるリソースについては，「付録 G.2(3)　EJB プロジェクトのリソースの移行」を参照してください。

EJB プロジェクトのリソースを移行するには，移行の対象となるリソースを WTP で作成した EJB プロジェクトにコピーします。リソースのコピーには，WTP のインポート機能を使用できます。インポートの手順を次に示します。

1. WTP の［プロジェクト・エクスプローラー］ビューで，WTP で作成したプロジェクトを選択します。

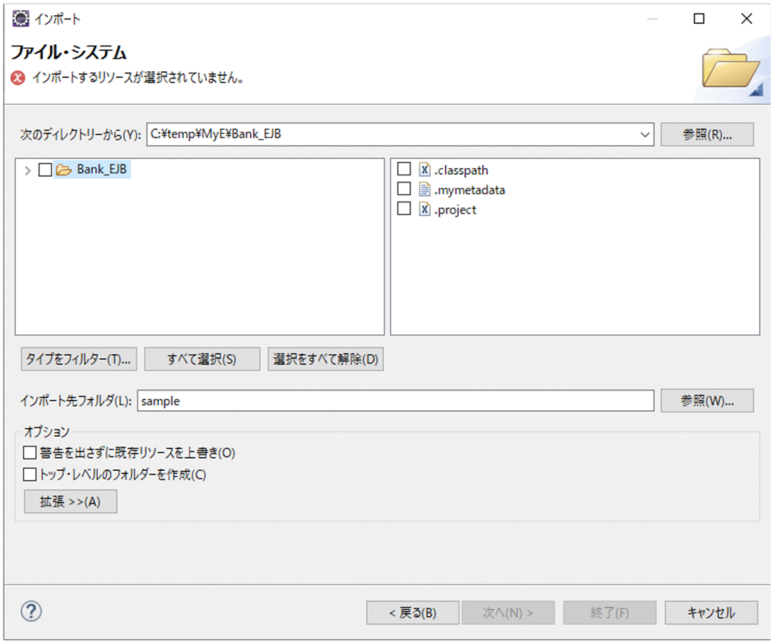
2. Eclipse のメニューから，［ファイル］－［インポート］を選択します。
   ［インポート］ダイアログが表示されます。

3. ［インポート］ダイアログで，［一般］－［ファイル・システム］を選択して，［次へ］ボタンをクリックします。
   ［インポート］ダイアログの［ファイル・システム］ページが表示されます。

4. [次のディレクトリーから] に，移行するプロジェクトロケーションディレクトリを指定します。

［参照］ボタンからも指定できます。

指定されたディレクトリ内のフォルダおよびファイルが表示されます。左ペインには，［次のディレクトリーから］で指定したディレクトリ内のフォルダとファイルがツリー形式で表示されます。右ペインには，左ペインで選択したフォルダ内のファイルが表示されます。



5. 移行するリソースを選択します。

- フォルダ内のすべてのリソースを移行する場合

  左ペインでフォルダのチェックボックスをチェックします。
- フォルダ内の一部のリソースだけ移行する場合

  右ペインで移行するファイルのチェックボックスをチェックします。

6. [オプション] で「トップ・レベルのフォルダーを作成」を指定します。

7. [終了] ボタンをクリックします。

選択したリソースがコピーされます。

注意事項

- EJB プロジェクトが複数ある場合は，すべての EJB プロジェクトのリソースを，対応する EJB プロジェクトに移行してください。移行対象となるリソースについては，「付録 G.2　移行対象となるリソース」を参照してください。
- Windows のエクスプローラを使用してファイルをコピーすることもできます。ただし，エクスプローラを使用した場合，［プロジェクト・エクスプローラー］などのビューにコピーしたフォルダやファイルが表示されません。コピーしたフォルダやファイルを表示するには，［プロジェクト・エクスプローラー］ビューでプロジェクトを選択し，コンテキストメニューから［更新］を選択してください。

付録 G.5　Web プロジェクトの移行

MyEclipse の Web プロジェクトを WTP に移行するには，WTP で動的 Web プロジェクトを作成してから，移行したい Web プロジェクトのリソースを移行します。移行手順について次に説明します。

(1)　動的 Web プロジェクトの作成

WTP で動的 Web プロジェクトを作成します。動的 Web プロジェクトの作成手順については，「4.4.1 動的 Web プロジェクトの作成」を参照してください。

WTP に移行するための，動的 Web プロジェクト作成時の注意を次に示します。

- Web プロジェクトが複数ある場合は，プロジェクトの数だけ動的 Web プロジェクトを作成します。
- WTP の［Dynamic Web Project］ページの［Project name］，［Web Module］ページの［Content directory］および［Context root］には，次の値を指定してください。

表 G-8　動的 Web プロジェクト作成時に指定する値

項目	値
Project name	Web プロジェクトのプロジェクト名を指定します。
Content directory	Web プロジェクトの［Web のルート・フォルダー］に指定した値と同じ値を指定します。
Context root	EAR プロジェクトの DD（application.xml）の<context-root>タグの値を指定します。

(2)　Web プロジェクトのリソースの移行

Web プロジェクトのリソースを移行します。移行できるリソースについては，「付録 G.2(2)　Web プロジェクトのリソースの移行」を参照してください。

Web プロジェクトのリソースを移行するには，移行の対象となるリソースを WTP で作成した動的 Web プロジェクトにコピーします。リソースのコピーには，WTP のインポート機能を使用できます。インポートの手順は，「付録 G.4(2)　EJB プロジェクトのリソースの移行」を参照してください。

注意事項

- Web プロジェクトが複数ある場合は，すべての Web プロジェクトのリソースを，対応する動的 Web プロジェクトに移行してください。移行対象となるリソースについては，「付録 G.2　移行対象となるリソース」を参照してください。
- Windows のエクスプローラを使用してファイルをコピーすることもできます。ただし，エクスプローラを使用した場合，［プロジェクト・エクスプローラー］などのビューにコピーしたフォルダやファイルが表示されません。コピーしたフォルダやファイルを表示するには，［プロジェクト・エクスプローラー］ビューでプロジェクトを選択し，コンテキストメニューから［更新］を選択してください。

付録 G.6　エンタープライズアプリケーションプロジェクトの移行

エンタープライズアプリケーションプロジェクトを WTP に移行するには，WTP でエンタープライズアプリケーションプロジェクトを作成してから，移行したいエンタープライズアプリケーションプロジェクトのリソースを移行します。移行手順について次に説明します。

(1)　エンタープライズアプリケーションプロジェクトの作成

WTP でエンタープライズアプリケーションプロジェクトを作成します。エンタープライズアプリケーションプロジェクトの作成手順については，「4.4.4　エンタープライズアプリケーションプロジェクトの作成」を参照してください。

エンタープライズアプリケーションプロジェクトを WTP のエンタープライズアプリケーションプロジェクトに移行する場合は，プロジェクト名にエンタープライズアプリケーションプロジェクトのプロジェクト名を指定してください。

また，次の表に応じて対応するモジュール・プロジェクトを作成してください。

表 G-9　作成するモジュール・プロジェクトの対応

項目	作成するモジュール・プロジェクト
エンタープライズアプリケーションプロジェクトに Web プロジェクトだけが組み込まれていた場合	動的 Web モジュール・プロジェクト
エンタープライズアプリケーションプロジェクトに EJB プロジェクトだけが組み込まれていた場合	EJB モジュール・プロジェクト
エンタープライズアプリケーションプロジェクトに Web プロジェクトおよび EJB プロジェクトが組み込まれていた場合	動的 Web モジュール・プロジェクト，および EJB モジュール・プロジェクト

(2)　エンタープライズアプリケーションプロジェクトのリソースの移行

エンタープライズアプリケーションプロジェクトのリソースを移行します。移行できるリソースについては，「付録 G.2(1)　エンタープライズアプリケーションプロジェクトのリソースの移行」を参照してください。

エンタープライズアプリケーションプロジェクトのリソースを移行するには，移行の対象となるリソースを WTP で作成したエンタープライズアプリケーションプロジェクトにコピーします。ただし，application.xml は WTP のプロジェクトに移行できないため，定義だけをコピーする必要があります。ここでは，リソースのコピー方法と，application.xml の移行方法について説明します。

(a)　リソースのコピー方法

リソースのコピーには，WTP のインポート機能を使用できます。インポートの手順は，「付録 G.4(2) EJB プロジェクトのリソースの移行」を参照してください。

注意事項

- 移行対象となるリソースについては，「付録 G.2　移行対象となるリソース」を参照してください。
- Windows のエクスプローラを使用してファイルをコピーすることもできます。ただし，エクスプローラを使用した場合，［プロジェクト・エクスプローラー］などのビューにコピーしたフォルダやファイルが表示されません。コピーしたフォルダやファイルを表示するには，［プロジェクト・エクスプローラー］ビューでプロジェクトを選択し，コンテキストメニューから［更新］を選択してください。

(b)　application.xml の移行方法

WTP でエンタープライズアプリケーションプロジェクトを作成する際に，application.xml のひな型を生成できます。生成した application.xml に，移行前の環境で使用していた application.xml の定義内容をコピーします。エンタープライズアプリケーションプロジェクトの作成については，「4.4.4　エンタープライズアプリケーションプロジェクトの作成」を参照してください。

次に，WTP で自動生成された application.xml の例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:application="http://java.sun.com/xml/ns/javaee/application_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/application_5.xsd" id="Application_ID" version="5">
  <display-name>HelloEAR</display-name>
  <module>
    <web>
      <web-uri>HelloWeb.war</web-uri>
      <context-root>HelloWeb</context-root>
    </web>
  </module>
  <module>
    <ejb>HelloEJB.jar</ejb>
  </module>
</application>
```

application.xml は，エンタープライズアプリケーションプロジェクト作成時に設定した情報を基に自動生成されます。次に，application.xml のそれぞれのタグについて説明します。

- <display-name>タグ

  エンタープライズアプリケーションプロジェクトの作成時に，［EAR アプリケーション・プロジェクト］ページで指定した［プロジェクト名］が自動設定されます。
- <module>タグの中の<ejb>タグおよび<web-uri>タグ

  エンタープライズアプリケーションプロジェクトの作成時に，［エンタープライズ・アプリケーション］ページで指定したプロジェクトの［<プロジェクト名>.war］または［<プロジェクト名>.jar］が設定されます。

  WAR ファイル名および EJB-JAR ファイル名を変更すると，J2EE アプリケーションのデプロイに失敗するおそれがあるので，変更しないでください。

- <module>タグの中の<context-root>タグ
  Web プロジェクトの作成時に，［Web モジュール］ページの［Context root］に指定した値が設定されます。

付録 H　Developer Version 9 以前のバージョンからのバージョンアップ時の注意事項

Developer Version 11 以降，標準で提供していた開発環境で使用できる組み込みデータベースを HiRDB Embedded Server から HiRDB/Single Server へ変更しています。

バージョンアップする場合は，必ず旧バージョンで構築した環境を削除してからバージョンアップしてください。

旧バージョンの開発環境インスタントセットアップ機能で構築した環境は，新しいバージョンの開発環境インスタントセットアップ機能では削除できません。

旧バージョンで構築した環境を削除する前に Developer をバージョンアップしてしまった場合は，次の手順で旧バージョンの環境を削除してください。なお，開発環境インスタントセットアップ機能を使用しないで構築した旧バージョンの環境がある場合も，次に示す手順でデバッグ環境をアンセットアップしてから，新しいバージョンの環境を構築してください。

1. 次に示す Smart Composer 機能のコマンドおよびサーバ管理コマンド（CUI）を実行して，J2EE サーバおよびパフォーマンストレーサをアンセットアップします。
   - <Developerのインストールディレクトリ>¥manager¥bin¥cmx_stop_target -m localhost:<開発環境インスタントセットアップ機能で指定した接続HTTPポート番号> -u <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのID> -p <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのパスワード> -mode ALL -s InstantWebSystem※
   - <Developerのインストールディレクトリ>¥CC¥server¥bin¥cjsetup -d cmx_InstantWebSystem_unit1_J2EE_01
   - <Developerのインストールディレクトリ>¥manager¥bin¥cmx_delete_system -m localhost:<開発環境インスタントセットアップ機能で指定した接続HTTPポート番号> -u <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのID> -p <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのパスワード> -s InstantWebSystem※
2. 次に示すファイルをコピーし，cjsetup コマンド（J2EE サーバの削除）を実行して，ManagementServer をアンセットアップします。

   コピー元のファイル

   ・<Developer のインストールディレクトリ>¥manager¥config¥templates¥adminagent.properties

   ・<Developer のインストールディレクトリ>¥manager¥config¥templates¥mserver.properties

   ・<Developer のインストールディレクトリ>¥manager¥config¥templates¥mserver.xml

   ・<Developer のインストールディレクトリ>¥manager¥config¥templates¥mserver.cfg

   コピー先のディレクトリ

   <Developer のインストールディレクトリ>¥manager¥config

実行するコマンド（cjsetup コマンド）

```
<Developerのインストールディレクトリ>¥CC¥server¥bin¥cjsetup -d cosmi_m
```

3. 次に示すバッチファイルを実行して，組み込みデータベースをアンセットアップします。

```
cddbdelete.bat
```

4. 次に示すディレクトリを削除します。

- <組み込みデータベース構築ディレクトリ>¥area
- <組み込みデータベース構築ディレクトリ>¥bats
- <組み込みデータベース構築ディレクトリ>¥conf
- <組み込みデータベース構築ディレクトリ>¥ini

注※

- 引数-u は，Management Server リモート管理機能で管理ユーザを設定しない場合，省略してください。
- 引数-p は，Management Server リモート管理機能で管理ユーザを設定しない場合，またはパスワードが設定されていない場合，省略してください。

付録 I　Developer Version 8 以前のバージョンからのバージョンアップ時の注意事項

バージョンアップする場合は，旧バージョンで構築した環境を削除してからバージョンアップしてください。

旧バージョンの開発環境インスタントセットアップ機能と MyEclipse セットアップ機能で構築した環境がある場合は，それぞれ必ず旧バージョンの対応する機能を使用して環境を削除してください。新しいバージョンの開発環境インスタントセットアップ機能では削除できません。また，WTP をセットアップしている場合は，WTP をアンセットアップしてください。

旧バージョンで構築した環境を削除する前に Developer をバージョンアップしてしまった場合は，次の手順で旧バージョンの環境を削除してください。なお，開発環境インスタントセットアップ機能と MyEclipse セットアップ機能を使用しないで構築した旧バージョンの環境がある場合も，次に示す手順でデバッグ環境をアンセットアップしてから，新しいバージョンの環境を構築してください。

1. 次に示す Smart Composer 機能のコマンドおよびサーバ管理コマンド（CUI）を実行して，J2EE サーバおよびパフォーマンストレーサをアンセットアップします。
   - <Developerのインストールディレクトリ>¥manager¥bin¥cmx_stop_target -m localhost:<開発環境インスタントセットアップ機能で指定した接続HTTPポート番号> -u <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのID> -p <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのパスワード> -mode ALL -s InstantWebSystem※
   - <Developerのインストールディレクトリ>¥CC¥server¥bin¥cjsetup -d cmx_InstantWebSystem_unit1_J2EE_01
   - <Developerのインストールディレクトリ>¥manager¥bin¥cmx_delete_system -m localhost:<開発環境インスタントセットアップ機能で指定した接続HTTPポート番号> -u <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのID> -p <開発環境インスタントセットアップ機能で指定したManagement Server管理ユーザのパスワード> -s InstantWebSystem※
2. 次に示すファイルをコピーし，cjwebsetup コマンド（Web コンテナサーバの削除）を実行して，ManagementServer をアンセットアップします。

   コピー元のファイル

   ・<Developer のインストールディレクトリ>¥manager¥config¥templates¥adminagent.properties

   ・<Developer のインストールディレクトリ>¥manager¥config¥templates¥mserver.properties

   ・<Developer のインストールディレクトリ>¥manager¥config¥templates¥mserver.xml

   ・<Developer のインストールディレクトリ>¥manager¥config¥templates¥mserver.cfg

   コピー先のディレクトリ

   <Developer のインストールディレクトリ>¥manager¥config

実行するコマンド（cjwebsetup コマンド）

```
<Developerのインストールディレクトリ>¥CC¥web¥bin¥cjwebsetup -d cosmi_m
```

3. 次に示すバッチファイルを実行して，組み込みデータベースをアンセットアップします。

```
cddbdelete.bat
```

4. 次に示すディレクトリを削除します。

- <組み込みデータベース構築ディレクトリ>¥area
- <組み込みデータベース構築ディレクトリ>¥bats
- <組み込みデータベース構築ディレクトリ>¥conf
- <組み込みデータベース構築ディレクトリ>¥ini

注※

- 引数-u は，Management Server リモート管理機能で管理ユーザを設定しない場合，省略してください。
- 引数-p は，Management Server リモート管理機能で管理ユーザを設定しない場合，またはパスワードが設定されていない場合，省略してください。

付録 J　JSF／JPA を利用する場合に必要な設定

JSF／JPA を利用して J2EE アプリケーションを開発する場合，設定が必要です。ここではそれぞれの設定手順について説明します。

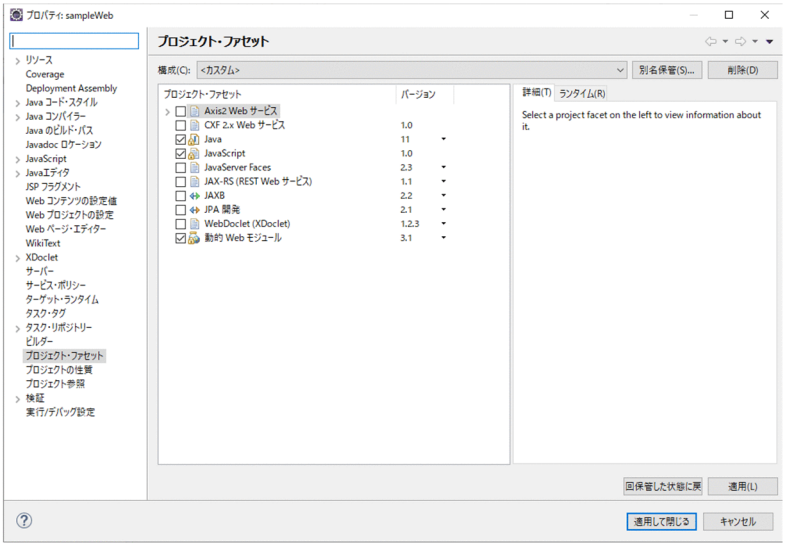
付録 J.1　推奨モードで JSF を利用する場合

JSF を利用して，J2EE アプリケーションを開発する場合，ファセットを設定する必要があります。

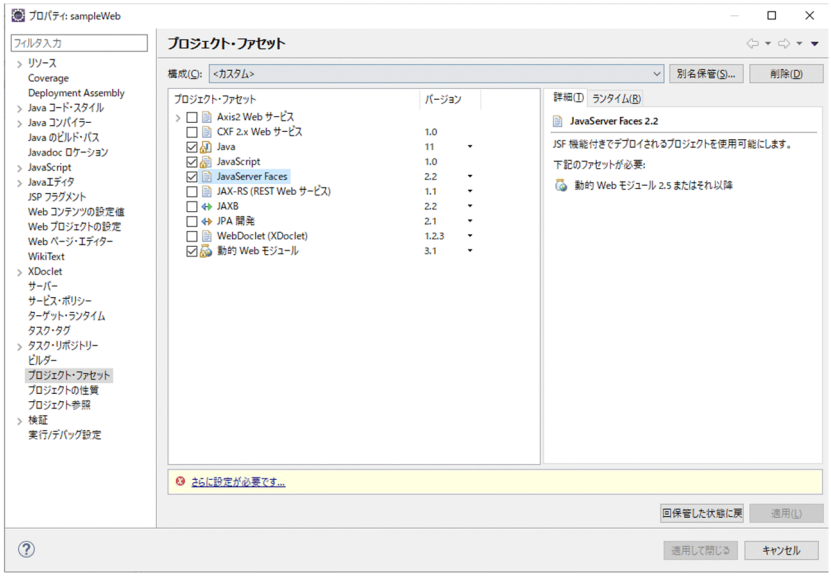
(1)　ファセットの設定

ファセットを設定する手順を次に示します。

1. [プロジェクト・エクスプローラー] ビューで，JSF を利用する動的 Web プロジェクトを選択します。
2. Eclipse のメニューから［プロジェクト］－［プロパティ］を選択します。
   ［プロパティ： <プロジェクト名>］ダイアログが表示されます。
3. [プロパティ： <プロジェクト名>] ダイアログの左ペインで［プロジェクト・ファセット］を選択します。
   ［プロジェクト・ファセット］ページが表示されます。

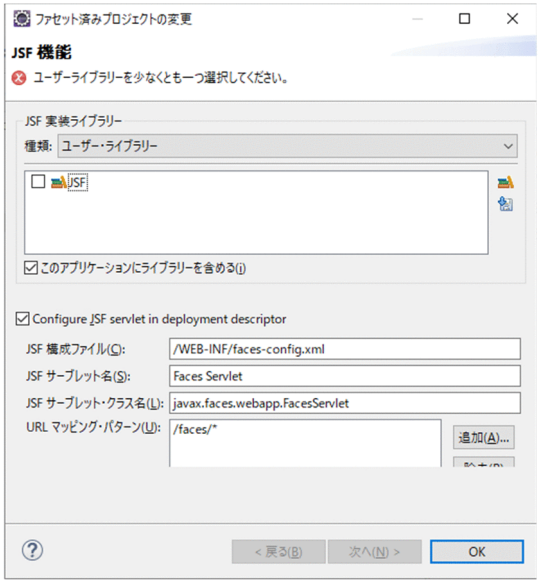


4. [プロジェクト・ファセット] で［JavaServer Faces］にチェックを入れ，バージョンに「2.2」または「2.3」を選択します（Jakarta EE 仕様である「3.0」以降のバージョンは選択できません）。
   ページの下部に「さらに設定が必要です...」のリンクが表示されます。



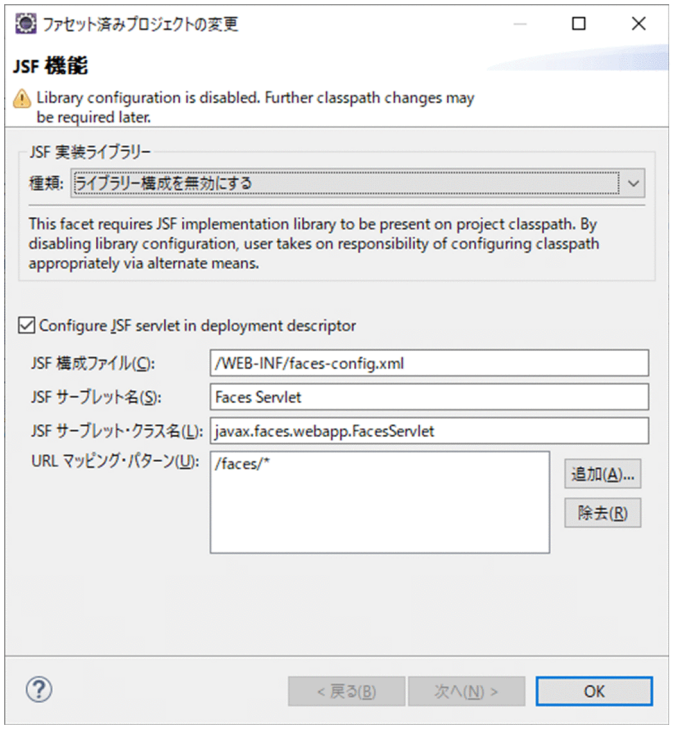
5. 「さらに設定が必要です...」のリンクをクリックします。

   ［ファセット済みプロジェクトの変更］ダイアログが表示されます。



6. ［ファセット済みプロジェクトの変更］ダイアログで，次の項目を指定します。

   - ［種類］で「ライブラリー構成を無効にする」を選択します。



7. [OK] ボタンをクリックします。

8. [プロパティ： <プロジェクト名>] ダイアログで，[OK] ボタンをクリックします。

付録 J.2　V9 互換モードで JSF を利用する場合

JSF を利用して，J2EE アプリケーションを開発する場合，ユーザー・ライブラリーやファセットを設定する必要があります。

それぞれの作業の概要を説明します。

1. ユーザー・ライブラリーの設定

   JSF を利用するためにはユーザー・ライブラリーを設定する必要があります。詳細は，「(1)　ユーザー・ライブラリーの設定」を参照してください。

2. ファセットの設定

   ファセットの設定をします。詳細は，「(2)　ファセットの設定」を参照してください。

以降で，これらの手順について説明します。

(1)　ユーザー・ライブラリーの設定

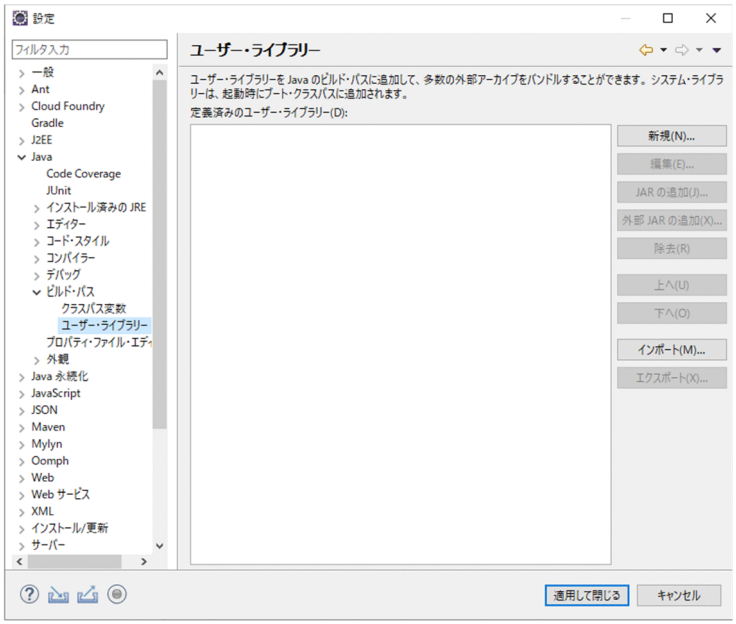
ユーザー・ライブラリーを設定する手順を次に示します。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。

   ［設定］ダイアログが表示されます。

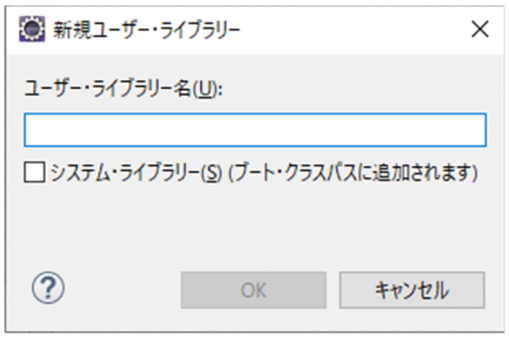
2. [設定] ダイアログの左ペインで［Java］－［ビルド・パス］－［ユーザー・ライブラリー］を選択します。

［ユーザー・ライブラリー］ページが表示されます。



3. [新規] ボタンをクリックします。

［新規ユーザー・ライブラリー］ダイアログが表示されます。



4. [ユーザー・ライブラリー名] に任意の名前を指定して，［OK］ボタンをクリックします。

［定義済みのユーザー・ライブラリー］に作成したユーザー・ライブラリーが表示されます。

5. [定義済みのユーザー・ライブラリー］で作成したユーザー・ライブラリーを選択し，［JAR の追加］ボタンをクリックします。

［JAR の選択］ダイアログが表示されます。

6. [JAR の選択］ダイアログで，［ファイル名］に追加する jar を指定して，［開く］ボタンをクリックします。

作成したユーザー・ライブラリーに指定した jar が追加されます。

ユーザー・ライブラリーに追加する jar は次のとおりです。

- <Developer のインストールディレクトリ>¥CC¥lib¥cjsf.jar

- <Developer のインストールディレクトリ>¥CC¥lib¥cjstl.jar
- <Developer のインストールディレクトリ>¥CC¥lib¥validation-api.jar

7. [設定] ダイアログで，[適用して閉じる] ボタンをクリックします。
   設定が保存されます。

注意事項

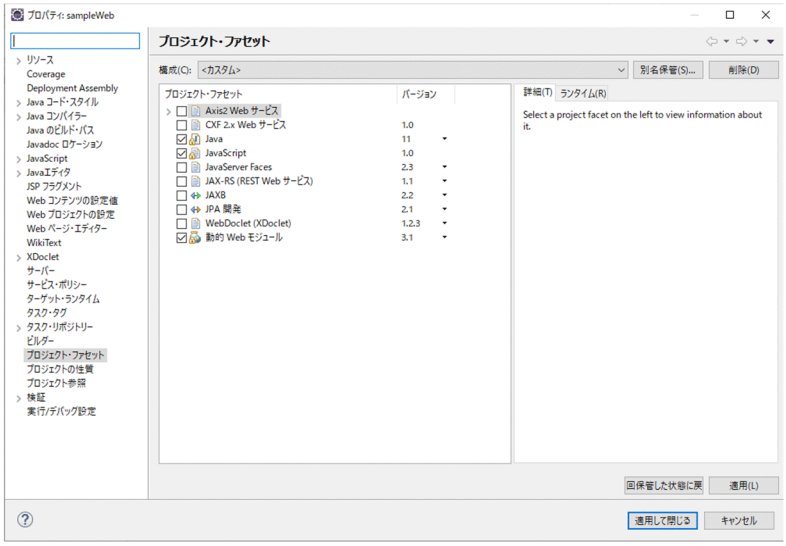
デバッグ環境で，JSF を利用する J2EE アプリケーションを実行するには，次の jar について J2EE サーバ側の設定が必要です。

- <Developer のインストールディレクトリ>¥CC¥lib¥cjsf.jar
- <Developer のインストールディレクトリ>¥CC¥lib¥cjstl.jar

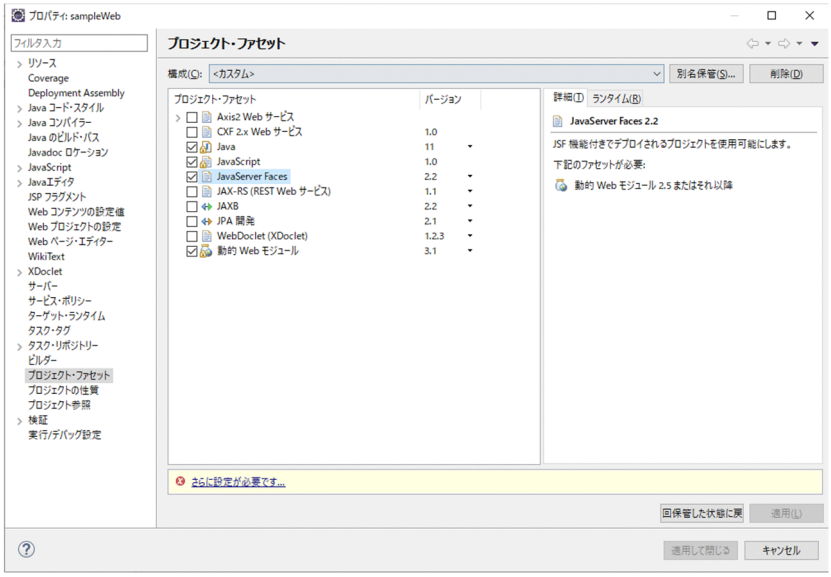
(2) ファセットの設定

ファセットを設定する手順を次に示します。

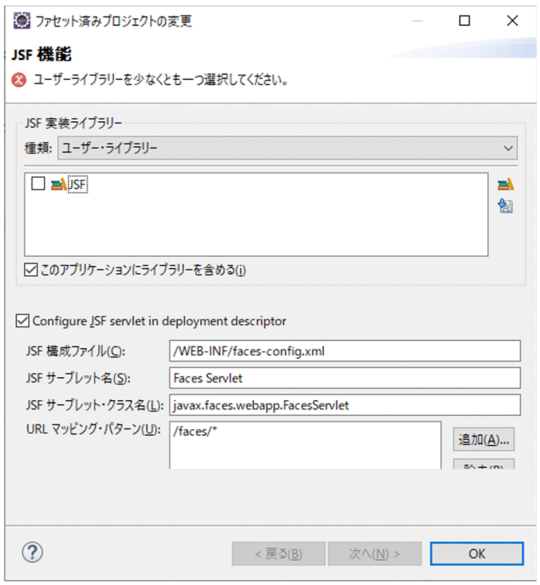
1. [プロジェクト・エクスプローラー] ビューで，JSF を利用する動的 Web プロジェクトを選択します。
2. Eclipse のメニューから［プロジェクト］－［プロパティ］を選択します。
   ［プロパティ： <プロジェクト名>］ダイアログが表示されます。
3. [プロパティ： <プロジェクト名>] ダイアログの左ペインで［プロジェクト・ファセット］を選択します。
   ［プロジェクト・ファセット］ページが表示されます。



4. [プロジェクト・ファセット] で［JavaServer Faces］にチェックを入れ，バージョンに「2.0」または「2.1」を選択します。
   ページの下部に「さらに設定が必要です...」のリンクが表示されます。



5. 「さらに設定が必要です...」のリンクをクリックします。
   ［ファセット済みプロジェクトの変更］ダイアログが表示されます。



6. ［ファセット済みプロジェクトの変更］ダイアログで，次の項目を指定します。
   - ［種類］に「ユーザー・ライブラリー」を選択します。
   - 表示されるユーザー・ライブラリーリストで，作成した JSF のユーザー・ライブラリーをチェックします。
   - ［このアプリケーションにライブラリーを含める］のチェックを外します。
7. ［OK］ボタンをクリックします。
8. ［プロパティ： <プロジェクト名>］ダイアログで，［OK］ボタンをクリックします。

(3)　web.xml 編集時の注意事項

JSF を利用するための設定をすると，web.xml に次の定義が設定されます。修正または削除してください。

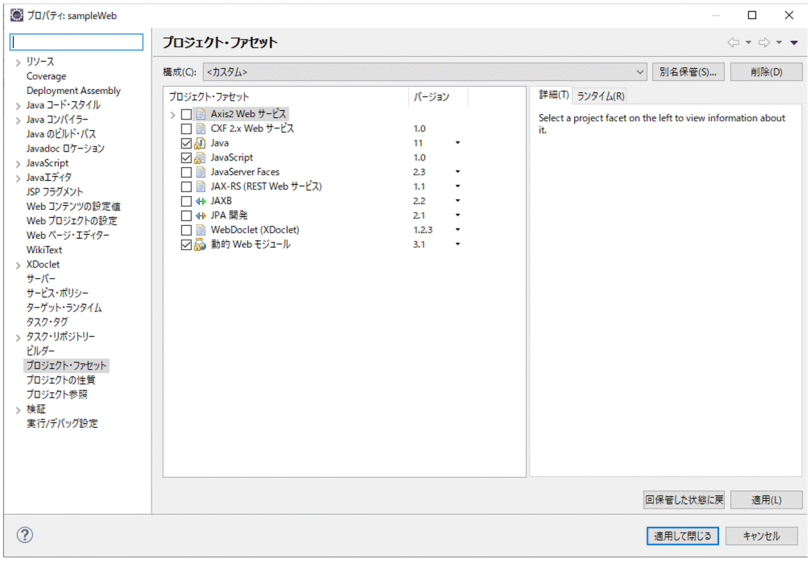
- <context-param>タグ
  - javax.faces.STATE_SAVING_METHOD
    「client」が指定されます。必要に応じて，「server」に修正してください。
  - javax.servlet.jsp.jstl.fmt.localizationContext
    必要に応じて修正または削除してください。
- <listener>タグ
  - com.sun.faces.config.ConfigureListener
    web.xml では設定する必要がないため，削除してください。

付録 J.3　JPA を利用する場合

JPA アプリケーションを開発する場合，ファセットの設定が必要です。

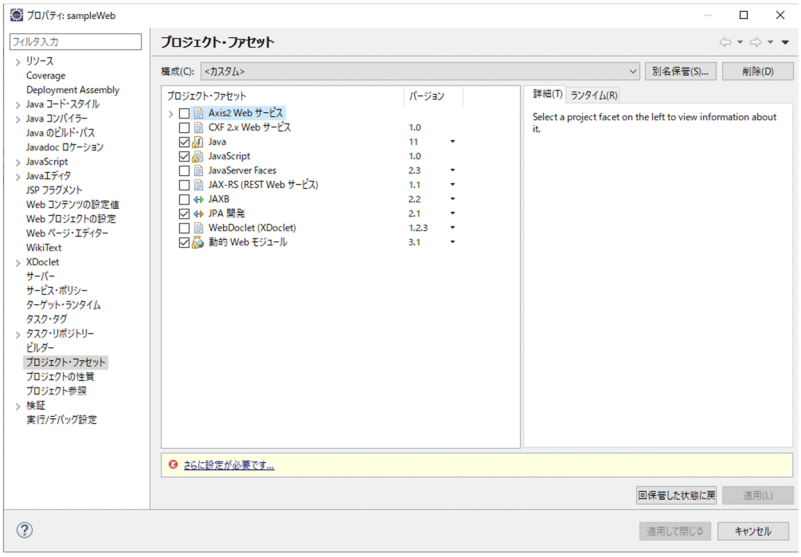
ファセットを設定する手順を次に示します。

1. [プロジェクト・エクスプローラー] ビューで，JPA を利用する動的 Web プロジェクトまたは EJB プロジェクトを選択します。
2. Eclipse メニューから［プロジェクト］－［プロパティ］を選択します。
   ［プロパティ： <プロジェクト名>］ダイアログが表示されます。
3. [プロパティ： <プロジェクト名>] ダイアログの左ペインで［プロジェクト・ファセット］を選択します。
   ［プロジェクト・ファセット］ページが表示されます。



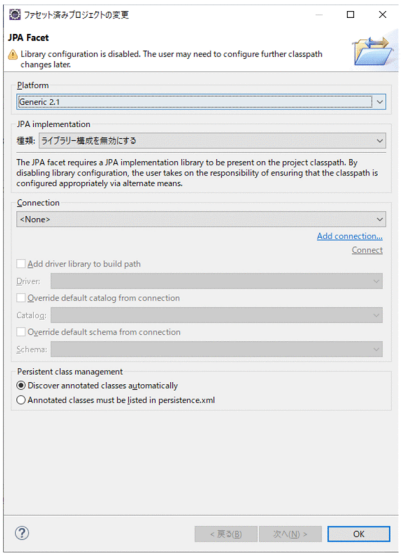
4. [プロジェクト・ファセット] で [JPA 開発] にチェックを入れ，J2EE サーバモードが「推奨モード」の場合は，バージョンに「2.1」または「2.2」を指定します（Jakarta EE 仕様である「3.0」以降のバージョンは選択できません）。J2EE サーバモードが「V9 互換モード」の場合は，バージョンに「1.0」を指定します。

   ページの下部に「さらに設定が必要です...」のリンクが表示されます。



5. 「さらに設定が必要です...」のリンクをクリックします。

   [ファセット済みプロジェクトの変更] ダイアログが表示されます。



6. [ファセット済みプロジェクトの変更] ダイアログに次の項目を入力します。

   - [Platform] に，J2EE サーバモードが「推奨モード」の場合は，バージョンに「Generic 2.1」または「Generic 2.2」を選択した JPA のバージョンに合わせて指定します。J2EE サーバモードが「V9 互換モード」の場合は，「Generic 1.0」を指定します。
   - [JPA implementation] の [種類] に「ライブラリー構成を無効にする」を指定します。

7. [OK] ボタンをクリックします。

8. [プロパティ： <プロジェクト名>] ダイアログで，[OK] ボタンをクリックします。

付録 K　JavaMail の使用例

JavaMail は，Web アプリケーション，または EJB アプリケーションで使用できます。ここでは，JavaMail の送信方法を説明します。JavaMail は，次の手順で送信します。

1. Session オブジェクトの取得
2. メッセージの作成
3. メッセージの送信

なお，以降の説明では，次のクラスのパッケージ名を省略します。

- javax.mail.NoSuchProviderException
- javax.mail.Session
- javax.mail.Transport
- javax.mail.MessagingException
- javax.mail.internet.MimeMessage
- javax.mail.internet.InternetAddress

付録 K.1　Session オブジェクトの取得

Session オブジェクトの取得方法および設定方法には，メールコンフィグレーションを使用する場合と，メールコンフィグレーションを使用しない場合があります。ここでは，それぞれの Session オブジェクトの取得方法と設定方法を説明します。

(1)　メールコンフィグレーションを使用する場合

ここでは，メールコンフィグレーションを使用する場合の Session オブジェクトの取得方法と設定方法を説明します。

(a)　メールコンフィグレーションの設定

メールコンフィグレーションを取得する前に，メールコンフィグレーションの設定とリソースのリンク解決をします。メールコンフィグレーションの設定については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「6.3　メールコンフィグレーションの設定」を参照してください。リソースのリンク解決については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「9.3.2　メールコンフィグレーションのリファレンス定義」を参照してください。

メールコンフィグレーションの設定例を示します。属性ファイルを次のように作成します。

```
<?xml version="1.0" encoding="MS932"?>
<!DOCTYPE hitachi-mail-property PUBLIC
 '-//Hitachi, Ltd.//DTD Mail Property 7.0//EN'
```

```
 'http://localhost/hitachi-mail-property_7_0.dtd'>
<hitachi-mail-property>
  <description>メール属性ファイルの例</description>
  <display-name>TheMailSession</display-name>
  <from>from_address@example.com</from>
  <server>MyMailServer</server>
  <runtime/>
</hitachi-mail-property>
```

次に，リソースの属性設定コマンドを使って，属性ファイルをリソースとして登録します。コマンドの指定例を示します。

```
cjsetresprop -type mail -resname TheMailSession -c mail.xml
```

(b) メールコンフィグレーションの取得

メールコンフィグレーションは，アノテーションまたは JNDI を使用して取得します。それぞれの取得方法を説明します。

- アノテーションを使用したメールコンフィグレーションの取得

  アノテーションを使用する場合は，次のような記述をしてメールコンフィグレーションを取得します。

  ```
  // フィールド変数
  // アノテーションによるリソースの設定
  @Resource(mappedName="TheMailSession")
  Session session;
  ```

- JNDI を使用したメールコンフィグレーションの取得

  アノテーションを使用しない場合は，リソースのリンク解決をして，JNDI を使用してメールコンフィグレーションを取得します。

  ```
  Session session = null;
  try {
      javax.naming.InitialContext initial =
      new javax.naming.InitialContext();
      session = (Session) initial.lookup("java:comp/env/TheMailSession");
  } catch (javax.naming.NamingException e) {
      // リソースが見つからなかった等
  }
  ```

(2) メールコンフィグレーションを使用しない場合

メールコンフィグレーションを使用しない場合は，Session クラスのファクトリメソッドを使用して，Session オブジェクトを取得できます。この場合，メールコンフィグレーションで設定する項目は取得できないので，プログラム内で設定する必要があります。

```
java.util.Properties prop = new java.util.Properties();
// mail.hostプロパティを設定。
// 他、mail.transport.protocol、mail.user、mail.from等必要なプロパティを設定。
```

```
prop.setProperty(”mail.host”, ”smtp.example.com”);
Session session = Session.getInstance(prop);
```

付録 K.2　メッセージの作成

送信するメッセージを作成します。

メッセージの内容に，次の項目を設定してください。

- From ヘッダフィールド

  引数を設定しない場合は，InternetAddress クラスの getLocalAddress メソッドの戻り値が From ヘッダフィールドに設定されます。メールコンフィグレーションを使用した場合は，メールコンフィグレーションで設定した From ヘッダフィールドの値がメッセージに設定されます。

- To ヘッダフィールド

  MimeMessage.RecipientType.TO の代わりに MimeMessage.RecipientType.CC，MimeMessage.RecipientType.BCC を指定することで，CC，BCC ヘッダフィールドを設定できます。

- Subject ヘッダフィールド

  charset パラメタを指定しない場合は，mail.mime.charset プロパティの値がエンコードに使用されます。

- ユーザ定義フィールド

- メール本文

  メールの内容にコンテントタイプ text/plain として本文を設定します。また，charset パラメタを指定しない場合は，mail.mime.charset プロパティの値がエンコードに使用されます。

- Date ヘッダフィールド

メッセージ内容の設定例を示します。

```
MimeMessage msg = new MimeMessage(session);
try{
    // Fromヘッダフィールドの設定
    msg.setFrom();

    // Toヘッダフィールドの設定
    msg.setRecipients(MimeMessage.RecipientType.TO, ”to_address@example.com”);

    // Subjectヘッダフィールドの設定
    msg.setSubject(”メッセージ主題”, ”ISO-2022-JP”);

    //ユーザ定義フィールドの設定
    msg.setHeader(”MyHeader”, ”MySendMailClient”);

    // メール本文の設定
    msg.setText(”メールメッセージ本文”, ”ISO-2022-JP”);

    // Dateヘッダフィールドの設定
```

```
    msg.setSentDate(new java.util.Date());

} catch (MessagingException e) {
    // メールアドレスの解析失敗等
}
```

付録 K.3　メッセージの送信

Transport クラスの send メソッドを使用して，メッセージを送信します。

```
try {
    Transport.send(msg);
} catch (MessagingException e) {
    // メッセージの送信に失敗した場合
}
```

また，複数のメッセージを送信する場合は，Transport オブジェクトを作成することで，1 接続で複数のメッセージを送信できます。

```
// Transportクラスのオブジェクトを得る
Transport transport = null;
try {
    transport = session.getTransport("smtp");
} catch (NoSuchProviderException e) {
    // プロバイダが見つからない場合
    // デフォルトの設定ではSMTPプロバイダが設定されているため発生しない
}
try {
    // SMTPサーバへの接続
    transport.connect();

    // 送信前に、ヘッダの更新を行う
    msg.saveChanges();

    // メッセージの送信
    transport.sendMessage(msg, msg.getAllRecipients());

    // 複数のメッセージを送信する場合sendMessage()メソッドを繰り返し呼び出す。
    // transport.sendMessage(msg2, msg2.getAllRecipients());
    // ...
} catch (MessagingException e) {
    // SMTPサーバへの接続、メッセージの送信失敗時
} finally {
    try{
        // SMTPサーバからの切断
        transport.close();
    }catch(MessagingException e){
        // サーバ切断失敗時
    }
}
```

付録 K.4　電子メールアドレス指定時の注意事項

次のクラスのメソッドには，電子メールアドレスは RFC822 に準拠した電子メールアドレスを指定してください。

- javax.mail.internet.MimeMessage
- javax.mail.internet.InternetAddress
- javax.mail.Transport
- 上記三つのクラスを継承したクラス

ただし，これらのメソッドに対して電子メールアドレスの検証をしない場合や，検証をする API が RFC822 に違反する電子メールアドレスをエラーとしない場合があるので注意してください。

付録 L　Developer が提供する機能を使用しない J2EE アプリケーションの開発

Developer が提供する機能を使用しない場合の J2EE アプリケーション開発の流れは，J2EE アプリケーションの形式によって異なります。

ここでは，概要や，必要な作業について説明します。

付録 L.1　Developer が提供する機能を使用しない J2EE アプリケーションの開発の概要

Developer が提供する機能を使用しない J2EE アプリケーションの開発環境，および J2EE アプリケーション開発の流れを説明します。

(1)　Developer が提供する機能を使用しない場合の開発環境

アプリケーションサーバで動作する J2EE アプリケーションは，Developer が提供する機能を使用しないで開発することもできます。J2EE アプリケーションを開発するときに必要な環境を，次の表に示します。

表 L-1　J2EE アプリケーションの開発環境（Developer が提供する機能を使用しない場合）

項番	分類	プログラム名称/構成ソフトウェア名称
1	JDK	Cosminexus Developer's Kit for Java
2	J2EE アプリケーション開発環境	• uCosminexus Application Server • uCosminexus Developer • uCosminexus Service Platform • uCosminexus Service Architect

(2)　J2EE アプリケーション開発の流れ（Developer が提供する機能を使用しない場合）

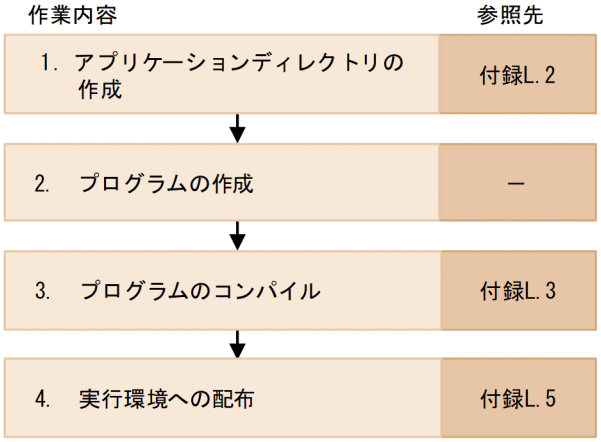
Developer が提供する機能を使用しない場合の J2EE アプリケーション開発の流れは，J2EE アプリケーションの形式によって異なります。J2EE アプリケーションの形式ごとに開発の流れを説明します。

なお，J2EE アプリケーションからほかの J2EE アプリケーションに含まれる Enterprise Bean や，ほかの J2EE サーバ上で動作する Enterprise Bean を呼び出す場合には，「付録 L.6　RMI-IIOP スタブの用意」も参照してください。

(a)　展開ディレクトリ形式の J2EE アプリケーション開発の流れ

展開ディレクトリ形式の場合の J2EE アプリケーション開発の流れを次の図に示します。

図 L-1　展開ディレクトリ形式の J2EE アプリケーション開発の流れ（Developer が提供する機能を使用しない場合）



（凡例）ー：なし

1. アプリケーションディレクトリの作成

展開ディレクトリ形式の J2EE アプリケーションを開発する場合，決められたディレクトリ構成を準備する必要があります。展開ディレクトリ形式のアプリケーションディレクトリを作成します。詳細は，「付録 L.2　アプリケーションディレクトリの作成（展開ディレクトリ）」を参照してください。

2. プログラムの作成

任意のエディタを使用して，プログラムを作成します。

3. プログラムのコンパイル

javac コマンドを使用して，プログラムをコンパイルします。詳細は，「付録 L.3　Java プログラムのコンパイル（javac コマンド）」を参照してください。

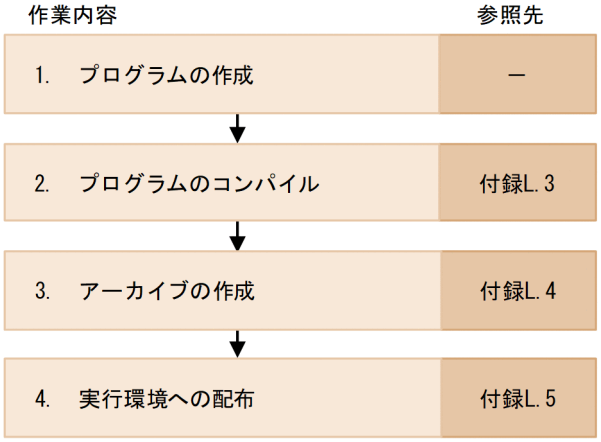
4. 実行環境への配布

作成したプログラムを実行環境へ配布します。詳細は，「付録 L.5　実行環境への配布」を参照してください。

(b)　アーカイブ形式の J2EE アプリケーション開発の流れ

アーカイブ形式の場合の J2EE アプリケーション開発の流れを次の図に示します。

図 L-2　アーカイブ形式の J2EE アプリケーション開発の流れ（Developer が提供する機能を使用しない場合）



（凡例）ー：なし

1. プログラムの作成

任意のエディタを使用して，プログラムを作成します。

2. プログラムのコンパイル

javac コマンドを使用して，プログラムをコンパイルします。詳細は，「付録 L.3　Java プログラムのコンパイル（javac コマンド）」を参照してください。

3. アーカイブの作成

jar コマンドを使用して，WAR ファイル，EJB-JAR ファイル，EAR ファイルを作成します。詳細は，「付録 L.4　アーカイブ形式の J2EE アプリケーションの作成」を参照してください。

4. 実行環境への配布

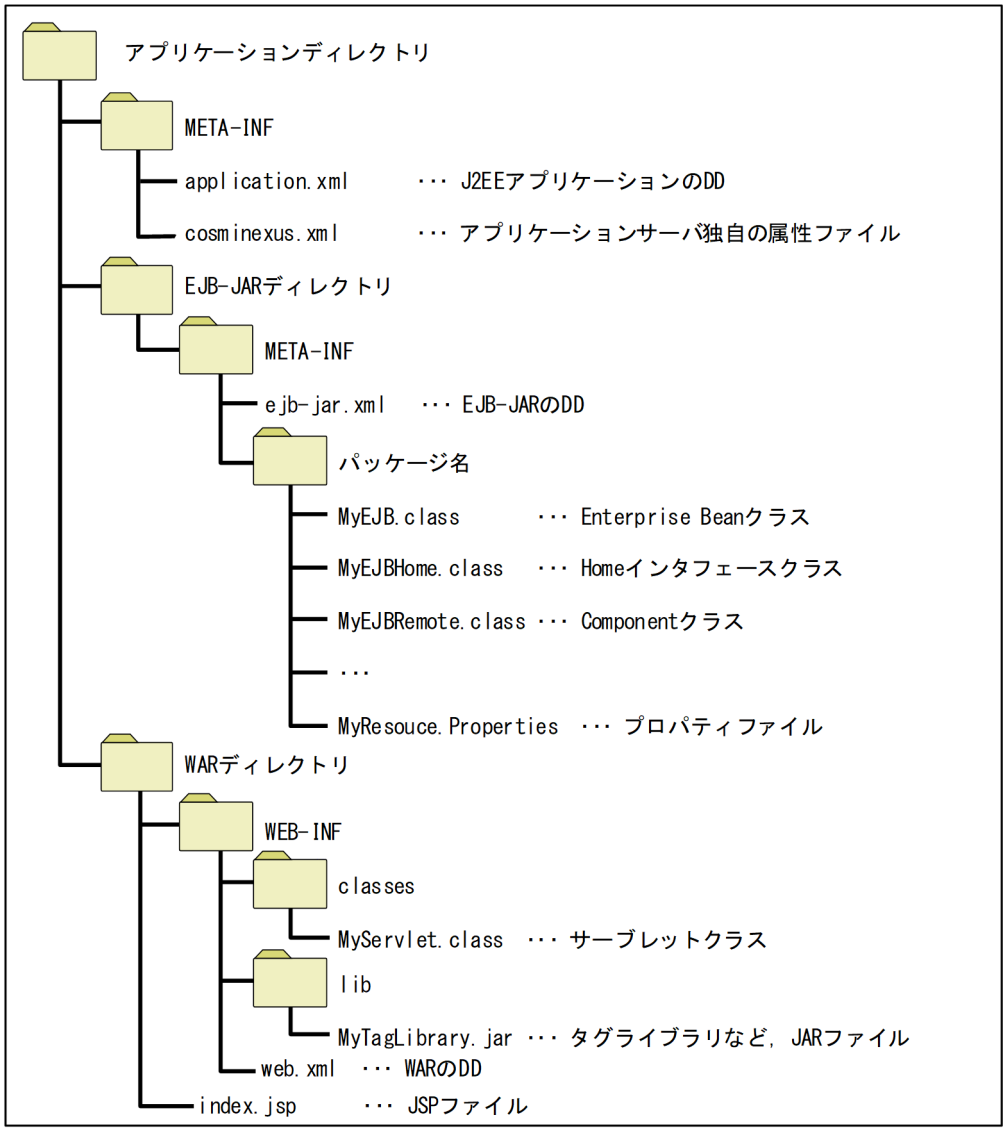
作成したプログラムを実行環境へ配布します。詳細は，「付録 L.5　実行環境への配布」を参照してください。

付録 L.2　アプリケーションディレクトリの作成（展開ディレクトリ）

Developer が提供するプラグインを使用しないで展開ディレクトリ形式の J2EE アプリケーションを開発する場合，最初にアプリケーションディレクトリを作成します。詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「18.4.2　アプリケーションディレクトリの構成」を参照してください。

作成するアプリケーションディレクトリの構成例を，次の図に示します。

図 L-3　アプリケーションディレクトリの構成例



アプリケーションディレクトリ名は任意です。EJB-JAR ディレクトリと WAR ディレクトリは，アプリケーションディレクトリの直下に作成する必要はありません。

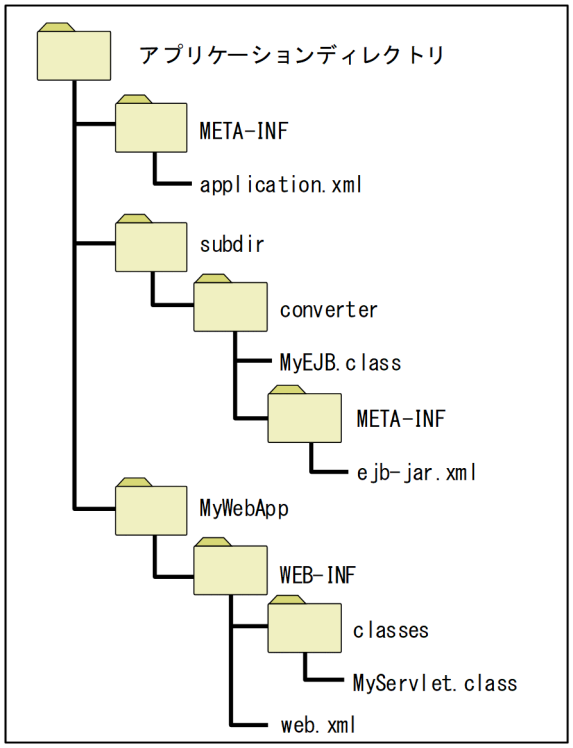
EJB-JAR ディレクトリおよび WAR ディレクトリは，それぞれ application.xml の<module>タグ以下に指定して作成します。application.xml の記述例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XML
Schema-instance" version="5" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://ja
va.sun.com/xml/ns/javaee/application_5.xsd">  <display-name>converter</display-name>
  <module>
    <ejb>subdir/converter.jar</ejb>
  </module>
  <module>
    <web>
      <web-uri>MyWebApp.war</web-uri>
      <context-root>/</context-root>
    </web>
```

```
  </module>
</application>
```

この application.xml に対応するアプリケーションディレクトリの構成を，次に示します。

図 L-4　アプリケーションディレクトリの構成



application.xml の定義内容とアプリケーションディレクトリの対応

- EJB-JAR ディレクトリおよび WAR ディレクトリは，それぞれ<module>タグ以下に指定した EAR ファイル上の相対パスに作成します。
- <module>タグ以下に定義する EJB-JAR および WAR のモジュールは，拡張子（.jar または.war）を付けて宣言します。
- 作成される EJB-JAR ディレクトリおよび WAR ディレクトリの名称は，拡張子（.jar または.war）を取り除いた名称となります。
- アプリケーションディレクトリ以下に置かれている拡張子が小文字の JAR ファイル（.jar）のうち，<module>タグ以下に定義していない JAR ファイルは，ライブラリ JAR として扱われます。

注意

- アプリケーションディレクトリとして UNC パスは指定できません。指定した場合は，サーバ管理コマンドでインポートするときにエラーとなります。
- J2EE サーバ起動処理中やサーバ管理コマンド実行中に，アプリケーションディレクトリ以下のファイルおよびディレクトリを追加，削除，上書きしないでください。
- application.xml に<alt-dd>タグが指定されている J2EE アプリケーションは，展開ディレクトリ形式で使用できません。展開ディレクトリ形式の J2EE アプリケーションをインポートする場合に，

J2EE アプリケーションの application.xml に<alt-dd>タグが指定されているときは，コマンド実行時にエラーとなります。

- <module>タグ以下に定義する EJB-JAR および WAR のモジュールで，異なる拡張子を付けた場合，J2EE アプリケーションのインポートに失敗します。
- アプリケーションディレクトリ以下に Java ソースファイルを置くような構成は推奨しません。同じディレクトリに置かれているクラスファイルと Java ソースファイルの同期が取れていない場合，J2EE アプリケーションの開始やリロードに失敗する場合があります。
- 半角記号「!」「#」「%」「+」はエスケープ文字として扱われるため，アプリケーションディレクトリ名およびモジュール名に指定しないでください。
- アプリケーションディレクトリ以下に置かれている JAR ファイルはライブラリ JAR として扱われるため，アプリケーションディレクトリ以下には，cjgetstubsjar コマンドで取得したスタブおよびインタフェースを置かないでください。

付録 L.3　Java プログラムのコンパイル（javac コマンド）

Eclipse を使用しないで J2EE アプリケーションを開発する場合，J2EE アプリケーションを構成するプログラム（JSP，サーブレットなど）をテキストエディタなどで実装し，JDK の javac コマンドによってコンパイルします。使用する JDK については，「付録 L.1(1)　Developer が提供する機能を使用しない場合の開発環境」を参照してください。

注意事項

Java の仕様として，実行環境の JDK のバージョンよりも新しいバージョンでコンパイルされたクラスファイルは実行できません。実行環境のアプリケーションサーバをインストールする際に選択した JDK バージョンよりも新しいバージョンの JDK でコンパイルする場合，-source/-target オプションや--release オプションを指定して，実行環境の JDK バージョンでサポートされるクラスファイルを生成してください。

次に，コンパイル時に指定する javac コマンドのコンパイルオプションを示します。

-classpath（クラスパス）

- ソースファイル中に Java EE 標準 API だけを使用している場合
  推奨モードの J2EE サーバで使用するアプリケーションのコンパイルでは，
  <Application Server のインストールディレクトリ>/CC/javaee/1100/lib/javaee-api.jar
  を追加します。Jakarta EE の API を使用する場合は，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「20.　パッケージ名変換機能」を参照してください。
  V9 互換モードの J2EE サーバで使用するアプリケーションのコンパイルでは，
  <Application Server のインストールディレクトリ>/CC/client/lib/j2ee-javax.jar
  を追加します。
- java.corba モジュールに含まれる API を使用するアプリケーションをコンパイルする場合は，

<Application Server のインストールディレクトリ>/jdk/lib/hcompatlib/hjdk.tpb.jar
を追加します。

- パッケージ名が com.hitachi で始まる Cosminexus アプリケーションサーバ独自の API（タイムアウト設定 API など）を使用している場合
  推奨モードの J2EE サーバで使用するアプリケーションのコンパイルでは，
  <Application Server のインストールディレクトリ>/CC/javaee/1100/lib/HiEJBClientStatic.jar
  および
  <Application Server のインストールディレクトリ>/TPB/lib/vbjorb.jar
  を追加します。
  V9 互換モードの J2EE サーバで使用するアプリケーションのコンパイルでは，
  <Application Server のインストールディレクトリ>/CC/client/lib/HiEJBClientStatic.jar
  および
  <Application Server のインストールディレクトリ>/TPB/lib/vbjorb.jar
  を追加します。
- 統合ユーザ管理が提供する API を使用している場合
  <Application Server のインストールディレクトリ>/manager/lib/usradmin.jar を指定します。
- Cosminexus XML Processor の JAXB 機能を使用する場合
  <Application Server のインストールディレクトリ>/jaxp/lib/csmjaxb.jar および
  <Application Server のインストールディレクトリ>/jaxp/lib/csmstax.jar を追加します。
- Web サービスの機能を使用する場合
  マニュアル「アプリケーションサーバ Web サービス開発ガイド」を参照してください。

サーバ起動・停止フック機能を使用して処理を実装する場合は，クラスパスに ejbserver.jar を追加する必要があります。ejbserver.jar の指定については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「19.3.2　コンテナ拡張ライブラリの作成と利用の流れ」を参照してください。

注意事項

サーバ起動・停止フック機能は，サーブレット，JSP，および Enterprise Bean などのプログラムからは使用できません。

-encoding（ソースファイルのエンコーディング）

javac 実行時のデフォルトエンコーディングとソースファイルのエンコーディングが異なるときに指定します。

-g（デバッグオプション）

デバッグ情報を生成します。「-g:none」は推奨しません。「-g:none」を指定すると例外発生時に行番号が取得できないため，デバッグの効率が低下します。

参考

Developer が提供する機能を使用しないで Eclipse を使ってコンパイルすることもできます。javac コマンドのコンパイルオプションの情報を参考に Eclipse の設定をしてください。

付録 L.4　アーカイブ形式の J2EE アプリケーションの作成

Developer が提供するプラグインを使用しないでアーカイブ形式の J2EE アプリケーションを開発する場合，jar コマンドを使用してアーカイブを作成します。ここでは，WAR ファイル，EJB-JAR ファイル，および EAR ファイルの作成について説明します。

(1)　WAR ファイルの作成（jar コマンド）

jar コマンドを使用して，Web アプリケーションとして使用するプログラムやファイルを WAR ファイルにアーカイブします。WAR ファイルにアーカイブする場合のディレクトリ構成，および jar コマンドの指定例を示します。

(a)　WAR ファイルのディレクトリ構成

表 L–2　jar コマンドによるアーカイブ時のディレクトリ構成（WAR）

ディレクトリ名またはファイル名	説明	必須
/	アーカイブ内のルートディレクトリです。	○
任意	JSP や HTML，そのほかの web クライアントからアクセスされるファイルです。WEB-INF 以外のサブディレクトリに格納することもできます。	−
/WEB-INF/	web クライアントから直接アクセスできないファイルを格納するディレクトリです。	○
web.xml	servlet 仕様で規定された DD です。	○※
*.tld	JavaServer Pages(TM)仕様で規定されたタグライブラリ・ディスクリプタのファイルです。サブディレクトリの下に格納してもかまいません。	−
/WEB-INF/classes/	サーブレットやそのほかのクラスファイルを格納するディレクトリです。格納するファイルがないときは不要です。	−
*.class	パッケージ名称に従って，ディレクトリ階層にサーブレットやそのほかのクラスファイル（*.class）を格納します。	−
/WEB-INF/lib/	サーブレットやそのほかのクラスを含む JAR ファイル（*.jar）を格納するディレクトリです。格納するファイルがないときは不要です。	−
*.jar	サーブレットやそのほかのクラスを含む JAR ファイル（*.jar）を格納します。	−
/META-INF/	管理情報を格納するディレクトリです。jar コマンドによって自動的に作成されます。	−

ディレクトリ名またはファイル名	説明	必須
MANIFEST.MF	jar コマンドの m オプションで指定されたファイルが格納されます。用意しなくてもアーカイブ内に自動的に作成されます。	−

（凡例）

○：アーカイブするときに必須であることを示します

−：該当しません

注※

次のどちらかの条件を満たす場合は，省略できます。

・WAR ファイルに含まれるファイルが，JSP ファイルまたは静的コンテンツだけの場合。

・サーブレット，フィルタ，リスナをアノテーションで定義した Web アプリケーションの場合。

(b) WAR ファイル作成時の jar コマンド指定例

カレントディレクトリおよびサブディレクトリに次のようにファイルが用意されているとします。

```
index.html
howto.jsp
feedback.jsp
images/banner.gif
images/jumping.gif
WEB-INF/web.xml
WEB-INF/lib/jspbean.jar
WEB-INF/classes/com/mycorp/servlets/MyServlet.class
WEB-INF/classes/com/mycorp/util/MyUtils.class
```

このとき，次のようにコマンドを実行すると，「MyApp.war」という名称の WAR ファイルが作成されます。

```
jar cf ..¥MyApp.war .
```

(2) EJB-JAR ファイルの作成（jar コマンド）

jar コマンドを使用して，Enterprise Bean や DD を EJB-JAR ファイルにアーカイブします。EJB-JAR ファイルにアーカイブする場合のディレクトリ構成，および jar コマンドの指定例を示します。

(a) EJB-JAR ファイルのディレクトリ構成

表 L-3 jar コマンドによるアーカイブ時のディレクトリ構成（EJB-JAR）

ディレクトリ名またはファイル名	説明	必須
/	アーカイブ内のルートディレクトリです。	○
*.class	Enterprise Bean のクラス，Enterprise Bean のホームインタフェース，コンポーネントインタフェース，ビジネスインタフェース，およびそれらが依存するそのほかの	○

ディレクトリ名またはファイル名	説明	必須
	クラス（インタフェース）のクラスファイル（*.class）をパッケージ名に従ったディレクトリ階層で格納します。	
/META-INF/	管理情報を格納するディレクトリです。jar コマンドによって自動的に作成されます。	－
ejb-jar.xml	EJB 仕様で規定された DD です。	○※
MANIFEST.MF	マニフェストファイルです。jar コマンドの m オプションを指定することで，アーカイブ内に自動的に作成されます。	－

（凡例）

○：アーカイブするときに必須であることを示します

－：該当しません

注※

EJB2.0 仕様の Enterprise Bean では必須です。アノテーションに対応した Enterprise Bean では不要です。

(b) EJB-JAR ファイル作成時の jar コマンド指定例

カレントディレクトリおよびサブディレクトリに次のようにファイルが用意されているとします。

```
com/mycorp/account/Account.class
com/mycorp/account/AccountEJB.class
com/mycorp/account/AccountHome.class
com/mycorp/account/InsufficientBalanceException.class
META-INF/ejb-jar.xml
```

このとき，次のようにコマンドを実行すると，「MyEJB.jar」という名称の EJB-JAR ファイルが作成されます。

```
jar cf ..¥MyEJB.jar .¥
```

(3) EAR ファイルの作成（jar コマンド）

jar コマンドを使用して，WAR ファイル，EJB-JAR ファイル，ライブラリ JAR ファイル，および DD を EAR ファイルにアーカイブします。EAR ファイルにアーカイブする場合のディレクトリ構成，および jar コマンドの指定例を示します。

(a) EAR ファイルのディレクトリ構成

表 L-4　jar コマンドによるアーカイブ時のディレクトリ構成（EAR）

ディレクトリ名またはファイル名	説明	必須
/	アーカイブ内のルートディレクトリです。	○
.jar，.war	J2EE アプリケーションに含まれる EJB-JAR ファイル，WAR ファイル，およびそのほかの JAR ファイルです。hitachi-runtime.jar という名称は使用できません。	○

ディレクトリ名またはファイル名	説明	必須
	application.xml に J2EE アプリケーションの構成モジュールとして定義されていない拡張子が小文字の JAR ファイル（.jar）は，ライブラリ JAR として扱われ，J2EE アプリケーションのクラスパスに追加されます。	
/META-INF/	管理情報を格納するディレクトリです。jar コマンドによって自動的に作成されます。	－
application.xml	JavaEE 仕様で規定された DD です。	○※
MANIFEST.MF	マニフェストファイルです。jar コマンドの m オプションを指定することで，アーカイブ内に自動的に作成されます。	－

（凡例）

○：アーカイブするときに必須であることを示します

－：該当しません

注※

省略できます。

(b) EAR ファイル作成時の jar コマンド指定例

カレントディレクトリおよびサブディレクトリに次のようにファイルが用意されているとします。

```
MyEJB.jar
MyUtil.jar
MyApp.war
META-INF/application.xml
```

このとき，次のようにコマンドを実行すると，「MyApp.ear」という名称の EAR ファイルが作成されます。

```
jar cf ..¥MyApp.ear .¥
```

付録 L.5 実行環境への配布

Eclipse のプラグインを使用しない場合，実行環境へ J2EE アプリケーションを配布するには，サーバ管理コマンドを使用して，J2EE アプリケーションのインポートや属性の設定をします。

(1) J2EE アプリケーションのインポート

J2EE アプリケーションをインポートします。J2EE アプリケーションの形式ごとにインポート方法を説明します。

(a) 展開ディレクトリ形式の J2EE アプリケーションの場合

J2EE アプリケーションを J2EE サーバにインポートするには，cjimportapp コマンドを使用します。展開ディレクトリ形式の J2EE アプリケーションをインポートするには，アプリケーションディレクトリをイ

ンポートする方法と，EAR または ZIP 形式の J2EE アプリケーションを展開ディレクトリ形式の J2EE アプリケーションとしてインポートする方法があります。それぞれのインポート方法について説明します。

なお，cjimportapp コマンドの使用方法については，マニュアル「Cosminexus アプリケーションサーバ アプリケーション設定操作ガイド」の「8.1.2　展開ディレクトリ形式の J2EE アプリケーションのインポート」を参照してください。

- アプリケーションディレクトリのインポート

  作成したアプリケーションディレクトリを J2EE サーバにインポートするには，cjimportapp コマンドに-a オプションを指定します。

  すでに J2EE サーバ上に同じアプリケーションディレクトリを持つ J2EE アプリケーションがある場合，インポートできません。

- EAR ファイル／ZIP ファイルのインポート

  EAR ファイルとして作成済みの J2EE アプリケーションまたは Developer からエクスポートした実行時情報付き ZIP ファイルを，展開ディレクトリ形式にしてインポートするには，cjimportapp コマンドに-d オプションを指定します。

  インポートした EAR ファイル／ZIP ファイル内のディレクトリ，EJB-JAR ファイル，および WAR ファイルの名称によって，作成されるディレクトリ名に衝突が起こることがあります。ディレクトリ名の衝突については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「18.5.3　EAR ファイル／ZIP ファイルの展開ディレクトリ形式でのデプロイ」のディレクトリ名の生成規則を参照してください。

(b)　アーカイブ形式の J2EE アプリケーションの場合

EAR ファイルとして作成済みの J2EE アプリケーションまたは Developer からエクスポートした実行時情報付き ZIP ファイルを，アーカイブ形式でインポートするには，cjimportapp コマンドを使用します。

なお，cjimportapp コマンドの使用方法については，マニュアル「Cosminexus アプリケーションサーバ アプリケーション設定操作ガイド」の「8.1.1　アーカイブ形式の J2EE アプリケーションのインポート」を参照してください。

(2)　J2EE アプリケーションの属性設定

J2EE アプリケーションの属性設定方法について，アノテーションを使用していない場合と，アノテーションを使用している場合の，それぞれの方法を説明します。

- アノテーションを使用していない場合

  cjsetappprop コマンドを使用して，J2EE アプリケーションの DD を書き換えます。cjsetappprop コマンドの使用方法については，マニュアル「Cosminexus アプリケーションサーバ アプリケーション設定操作ガイド」の「9.　J2EE アプリケーションのプロパティ設定」を参照してください。

- アノテーションを使用している場合

Java ソースコードに記述したアノテーションを編集して，コンパイルします。J2EE アプリケーションの開始時に変更した情報が読み込まれます。なお，アプリケーションディレクトリ以下の DD を修正しても，J2EE サーバは検知しません。

(3) J2EE アプリケーションの開始

cjstartapp コマンドを使用して J2EE アプリケーションを開始します。

cjstartapp コマンドの使用方法については，マニュアル「Cosminexus アプリケーションサーバ アプリケーション設定操作ガイド」の「10.2.1 J2EE アプリケーションの開始」を参照してください。

(4) クラスファイルの変更

Java ソースファイルを修正した場合は，その Java ソースファイルをコンパイルしてクラスファイルを生成します。

(5) J2EE アプリケーションのリロード（展開ディレクトリ形式）

展開ディレクトリ形式の J2EE アプリケーションの場合，リロード機能を使用できます。リロード機能を使用していると，J2EE サーバはクラスファイルの更新を検知し，更新後のクラスファイルを自動的に再読み込みします。リロード機能を使用するには，アプリケーションサーバにあらかじめリロードの設定をしておく必要があります。また，cjreloadapp コマンドを使用すると，任意のタイミングでリロードを実行できます。

J2EE アプリケーションの更新検知とリロードを有効にするための設定については，マニュアル「Cosminexus アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「18.8.12 J2EE アプリケーションの更新検知とリロードの設定」を参照してください。cjreloadapp コマンドでのリロードについては，マニュアル「Cosminexus アプリケーションサーバ アプリケーション設定操作ガイド」の「8.1.2 展開ディレクトリ形式の J2EE アプリケーションのインポート」を参照してください。

注意事項

アプリケーションがリロードできない場合およびリロードが禁止されたアプリケーションの場合，リロードを利用した開発はできません。

(6) J2EE アプリケーションの停止

cjstopapp コマンドを使用して J2EE アプリケーションを停止します。

cjstopapp コマンドの使用方法については，マニュアル「Cosminexus アプリケーションサーバ アプリケーション設定操作ガイド」の「10.2.2 J2EE アプリケーションの停止」を参照してください。

(7) J2EE アプリケーションの削除

cjdeleteapp コマンドを使用して J2EE アプリケーションを削除します。このとき，削除対象の J2EE アプリケーションのアプリケーションディレクトリは削除されません。J2EE サーバへの登録は解除されます。

cjdeleteapp コマンドの使用方法については，マニュアル「Cosminexus アプリケーションサーバ アプリケーション設定操作ガイド」の「10.4　J2EE アプリケーションの削除」を参照してください。

(8)　WAR アプリケーションのインポート

WAR アプリケーションをインポートします。WAR アプリケーションの形式ごとにインポート方法を説明します。

WAR アプリケーションの詳細は，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「18.9　WAR アプリケーション」を参照してください。

(a)　展開ディレクトリ形式の WAR アプリケーションの場合

展開ディレクトリ形式の WAR アプリケーションを J2EE サーバにインポートするには，cjimportwar コマンドに-a オプションを指定して使用します。

詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「18.9.2　実行できる WAR アプリケーションの形式」を参照してください。

(b)　アーカイブ形式の WAR アプリケーションの場合

WAR ファイルとして作成済みの WAR アプリケーションを，アーカイブ形式でインポートするには，cjimportwar コマンドに-f オプションを指定して使用します。

詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「18.9.2　実行できる WAR アプリケーションの形式」を参照してください。

付録 L.6　RMI-IIOP スタブの用意

J2EE アプリケーションからほかの J2EE アプリケーションに含まれる Enterprise Bean や，ほかの J2EE サーバ上で動作する Enterprise Bean を呼び出す場合には，呼び出し側に RMI-IIOP スタブを用意する必要があります。RMI-IIOP スタブを用意する方法は，次の 3 とおりです。

- J2EE サーバによってデプロイ時に自動的に作成

  呼び出し先 Enterprise Bean のリモートインタフェースを呼び出し元の J2EE アプリケーションに含めます。呼び出し元サーバの定義ファイル（usrconf.properties）の指定を変更することで，J2EE アプリケーションの開始時にスタブクラスが自動生成されます。呼び出し元サーバの定義ファイルでは，次を指定します。

  ```
  ejbserver.deploy.stub.generation.scope=app
  ```

- J2EE サーバによって呼び出し先サーバから自動的に取得

  呼び出し先サーバの定義ファイル（usrconf.properties）の指定で，実行時にスタブクラスが自動的にダウンロードされるようにします。呼び出し先サーバの定義ファイルでは，次を指定します。

```
ejbserver.DynamicStubLoading.Enabled=true
```

- 手動で取得して J2EE アプリケーション内に含める

  呼び出す Enterprise Bean をデプロイし，デプロイしたサーバからスタブクラス（stubs.jar）を取得して呼び出し元の J2EE アプリケーションに含めます。

RMI-IIOP スタブが自動的に作成，取得される場合，および手動で取得する場合の，呼び出し先サーバと呼び出し元サーバの J2EE アプリケーションのディレクトリ構成を示します。

(1) RMI-IIOP スタブが自動的に作成，取得される場合の J2EE アプリケーションのディレクトリ構成

デプロイ時に RMI-IIOP スタブが自動的に作成される場合，または実行時に RMI-IIOP スタブが自動的にダウンロードされる場合の，呼び出し先の Enterprise Bean を含む J2EE アプリケーションの構成を次に示します。

表 L-5 自動で作成，取得する場合の J2EE アプリケーションの構成（呼び出し先）

ディレクトリ名またはファイル名	説明
/	アーカイブ内のルートディレクトリです。
myejb.jar	呼び出し先の Enterprise Bean を含む EJB-JAR ファイルです。
/META-INF/	管理情報を格納するディレクトリです。jar コマンドによって自動的に作成されます。
application.xml	JavaEE 仕様で規定された DD です。

デプロイ時に RMI-IIOP スタブが自動的に作成される場合，または実行時に RMI-IIOP スタブが自動的にダウンロードされる場合の，呼び出し元のサーブレットを含む J2EE アプリケーションの構成を次に示します。

表 L-6 自動で作成，取得する場合の J2EE アプリケーションの構成（呼び出し元）

ディレクトリ名またはファイル名	説明
/	アーカイブ内のルートディレクトリです。
myapp.war	呼び出し元のサーブレットが含まれる Web アプリケーションです。WAR ファイル内のディレクトリ構成例を示します。 /（ルートディレクトリ） /WEB-INF/ web.xml /WEB-INF/lib/ *.jar（呼び出し先 Enterprise Bean のホームインタフェース，コンポーネントインタフェースを含む JAR ファイル）

ディレクトリ名またはファイル名	説明
	/WEB-INF/classes/ *.class（呼び出し元のサーブレットのクラスファイル）
/META-INF/	管理情報を格納するディレクトリです。jar コマンドによって自動的に作成されます。
application.xml	Java EE 仕様で規定された DD です。

(2) RMI-IIOP スタブを手動で取得する場合の J2EE アプリケーションのディレクトリ構成

手動でスタブクラスを取得して，呼び出し元の J2EE アプリケーションに含ませる場合の，J2EE アプリケーションのディレクトリ構成を示します。呼び出し先の J2EE アプリケーションの構成は，「付録 L.6(1) RMI-IIOP スタブが自動的に作成，取得される場合の J2EE アプリケーションのディレクトリ構成」で示す呼び出し先の構成と同じです。

表 L-7 手動で含ませる場合の J2EE アプリケーションの構成（呼び出し元）

ディレクトリ名またはファイル名	説明
/	アーカイブ内のルートディレクトリです。
myapp.war	呼び出し元のサーブレットが含まれる Web アプリケーションです。WAR ファイル内のディレクトリ構成例を示します。 /（ルートディレクトリ） /WEB-INF/ web.xml /WEB-INF/lib/ *.jar（呼び出し先 Enterprise Bean のホームインタフェース，コンポーネントインタフェースを含む JAR ファイル） stubs.jar※（手動で取得した RMI-IIOP スタブを含む JAR ファイル） /WEB-INF/classes/ *.class（呼び出し元のサーブレットのクラスファイル）
/META-INF/	管理情報を格納するディレクトリです。jar コマンドによって自動的に作成されます。
application.xml	JavaEE 仕様で規定された DD です。

注※

クラスローダの構成上，スタブクラスとリモートインタフェースは両方とも同じ場所（/WEB-INF/lib）に配置する必要があります。

付録 M　用語解説

マニュアルで使用する用語について

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 用語解説」を参照してください。

索引

H

J

L

M

N

R

S

T

U

W

と

は

ひ

ふ

ほ

め

ゆ

ら

り

ろ