HITACHI
Inspire the Next

Cosminexus V11 アプリケーションサーバ リファレンス API 編

文法書

3021-3-J21-70

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

Microsoft，Windows，Windows Server は，マイクロソフト 企業グループの商標です。

Oracle(R)，Java ，MySQL 及び NetSuite は，Oracle，その子会社及び関連会社の米国及びその他の国における登録商標です。

UNIX は，The Open Group の登録商標です。

その他記載の会社名，製品名などは，それぞれの会社の商標もしくは登録商標です。

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

■ 発行

2025 年 4 月 3021-3-J21-70

■ 著作権

変更内容

変更内容(3021-3-J21-70) uCosminexus Application Server 11-60，uCosminexus Client 11-60，uCosminexus Developer 11-60，uCosminexus Service Architect 11-60，uCosminexus Service Platform 11-60

追加・変更内容	変更個所
記載内容は変更なし（リンク情報だけを変更した）。	-

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

1 APIの概要

この章では，アプリケーションサーバで使用するAPIとタグライブラリの種類，およびこのマニュアルでの記述形式について説明します。

1.1　API とタグライブラリの種類

アプリケーションサーバで使用する API とタグライブラリの種類について説明します。

このマニュアルでは，アプリケーションごとに使用できる API とタグライブラリを三つに分類して説明します。

- J2EE アプリケーションで使用できる API とタグライブラリ
- バッチアプリケーションまたは EJB クライアントアプリケーションで使用できる API
- Web サービスを実行するシステムで使用できる API

J2EE アプリケーションで使用できる API とタグライブラリを次の表に示します。

表 1-1　J2EE アプリケーションで使用できる API

API とタグライブラリの種類	API とタグライブラリの説明	参照先マニュアル	参照先
Web コンテナで使用する API	Web コンテナで使用する API です。	このマニュアル	3 章
EJB クライアントアプリケーションで使用する API	EJB クライアントのセキュリティや通信タイムアウトなどを設定するための API です。		4 章
TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API	TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API です。		5 章
スレッドの非同期並行処理で使用する API	スレッドの非同期並行処理で使用する API です。		6 章
統合ユーザ管理フレームワークで使用する API	統合ユーザ管理機能を使用する場合に，ユーザ認証を実装するために使用する，統合ユーザ管理フレームワークのライブラリです。	アプリケーションサーバ 機能解説 セキュリティ管理機能編	15 章
統合ユーザ管理フレームワークで使用するタグライブラリ	統合ユーザ管理機能を使用する場合に，ユーザ認証を実装するために使用する，統合ユーザ管理フレームワークの JSP タグライブラリです。	アプリケーションサーバ 機能解説 セキュリティ管理機能編	16 章
ユーザログ機能で使用する API	J2EE アプリケーションが出力するログ（ユーザログ）をトレース共通ライブラリ形式で出力する場合に，ユーザログ出力を実装するための API です。	このマニュアル	7 章
監査ログ出力で使用する API	J2EE アプリケーションで監査ログを出力するための API です。		8 章
性能解析トレースで使用する API	性能解析トレースでアプリケーションサーバの処理性能を解析する場合に，ルートアプリケーション情報を文字列表現で取得するための API です。		9 章
JavaVM で使用する API	Java プログラムから直接 GC のメモリ情報を取得するための API です。		10 章

なお，API とタグライブラリのほかに，アノテーションと Dependency Injection も使用できます。アノテーションと Dependency Injection については，「2.　アプリケーションサーバで使用するアノテーションおよび Dependency Injection」を参照してください。

バッチアプリケーションまたは EJB クライアントアプリケーションで使用できる API を次の表に示します。

表 1-2　バッチアプリケーションまたは EJB クライアントアプリケーションで使用できる API

API とタグライブラリの種類	API とタグライブラリの説明	参照先マニュアル	参照先
EJB クライアントアプリケーションで使用する API	EJB クライアントアプリケーションのセキュリティや通信タイムアウトなどを設定するための API です。	このマニュアル	4 章
ユーザログ機能で使用する API	バッチアプリケーションまたは EJB クライアントアプリケーションが出力するログ（ユーザログ）をトレース共通ライブラリ形式で出力する場合に，ユーザログ出力を実装するための API です。		7 章
監査ログ出力で使用する API	バッチアプリケーションまたは EJB クライアントアプリケーションで監査ログを出力するための API です。		8 章
性能解析トレースで使用する API	性能解析トレースでアプリケーションサーバの処理性能を解析する場合に，ルートアプリケーション情報を文字列表現で取得するための API です。		9 章
JavaVM で使用する API	Java プログラムから直接 GC のメモリ情報を取得するための API です。		10 章

Web サービスを実行するシステムで使用できる API を次の表に示します。

表 1-3　Web サービスを実行するシステムで使用できる API

API の種類	API の説明	参照先マニュアル	参照先
JAX-WS 2.2 仕様に対応した SOAP Web サービスの開発で使用する API	SOAP Web サービスや Web サービスクライアントを開発するときに使用します。	アプリケーションサーバ Web サービス開発ガイド	19 章
JAX-RS 1.1 仕様に対応した RESTful Web サービスの開発で使用する API	RESTful Web サービス（Web リソース）を開発するときに使用します。なお，HTTP クライアントは，RESTful Web サービス用クライアント API か，または標準的な Java API を使用して開発します。		24 章
Web リソースクライアントの実装で使用する RESTful Web サービス用クライアント API	RESTful Web サービス（Web リソース）のクライアントを RESTful Web サービス用クライアント API で実装するときに使用します。		25 章

1.2　アノテーションの記述形式

1.2.1　記述形式の詳細

2 章では，アノテーションについて次の形式で説明します。なお，各アノテーションはアルファベットの順に説明します。

(1)　説明

アノテーションの機能について説明します。

(2)　属性

アノテーションに含まれる属性について説明します。各属性については，次の形式で説明します。

(a)　属性名

型
　属性の型を示します。

説明
　属性の機能について説明します。

デフォルト値
　属性のデフォルト値を示します。

1.3 API の記述形式

3 章から 10 章では，API について次の形式で説明します。なお，各 API は，アルファベットの順に説明します。

説明

API の機能について説明します。

形式

API の記述形式を示します。

パラメタ

API のパラメタについて説明します。

例外

API を利用する際に発生する例外について説明します。

戻り値

API の戻り値について説明しています。

注意事項

API を利用する上での注意事項について説明します。

2 アプリケーションサーバで使用するアノテーションおよび Dependency Injection

この章では，アプリケーションサーバで使用するアノテーションおよび Dependency Injection について説明します。

なお，アノテーション参照抑止機能を使用している場合，アノテーションの指定は参照されません。アノテーション参照抑止機能については，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「17.5　アノテーションの参照抑止」を参照してください。

2.1　アノテーションのサポート範囲

アノテーションは，ソースコードに注釈を付けることができる言語仕様です。

アノテーションのサポート範囲を次に示します。

2.1.1　javax.annotation パッケージに含まれるアノテーションのサポート範囲

javax.annotation パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

(1)　WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-1　WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様				インターセプタ（CDI 対象）	例外クラス	ManagedBean（JSF）	その他のクラス
	サーブレット	サーブレット（API）	サーブレットフィルタ	サーブレットフィルタ（API）	イベントリスナ	イベントリスナ（API）	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ				
								クラシックタグハンドラ	シンプルタグハンドラ					
@ManagedBean	−	−	−	−	−	−	−	−	−	−	−	−	−	○
@PostConstruct	○	−	○	−	○	−	−	○	○	×	○	−	○※1	○※3
@PreDestroy	○	−	○	−	○	−	−	○	○	×	○	−	○※1	○※3
@Priority	−	−	−	−	−	−	−	−	−	−	○	−	−	○
@Resource	○	−	○	−	○	−	−	○	○	×	○※2	−	○	○※3
@Resources	○	−	○	−	○	−	−	○	○	×	×	−	○	×

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

注※1

JSF に依存するアノテーションです。サポート範囲については，JSF 仕様のドキュメントを参照してください。

注※2

DI の実行可否は，インターセプタの指定方法やインターセプタメソッドの種類によって非サポートの場合があります。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「15. Interceptors」を参照してください。

注※3

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

(2) WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-2 WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@PostConstruct	○	○	○	－	○	○	×	－
@PreDestroy	○	○	○	－	○	○	×	－
@Resource	○	○	○	－	○	○	×	－
@Resources	○	○	○	－	○	○	×	－

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

(3) EJB-JAR ファイル（EJB3.1/3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-3　EJB-JAR ファイル（EJB3.1/3.0 対応）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Enterprise Bean						インターセプタ(CDI対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ				
					デフォルトインターセプタ以外	デフォルトインターセプタ			
@ManagedBean	−	−	−	−	−	−	−	−	○
@PostConstruct	−	○	−	×	○	○	○	−	○ ※2
@PreDestroy	−	○	−	×	○	○	○	−	○ ※2
@Priority	−	−	−	−	○	−	○	−	○
@Resource	−	○	−	×	○ ※1	○	○ ※1	−	○ ※2
@Resources	−	○	−	×	○ ※1	○	×	−	×

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

−：標準仕様で対応していない。

注※1

DI の実行可否は，インターセプタの指定方法やインターセプタメソッドの種類によって非サポートの場合があります。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「15.　Interceptors」を参照してください。

注※2

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

(4)　ライブラリ JAR の Servlet/JSP 仕様のクラス

ライブラリ JAR の Servlet/JSP 仕様のクラスに記述できるアノテーションの一覧を示します。

表 2-4　ライブラリ JAR（Servlet/JSP）に記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様			
	サーブレット	サーブレット(API)	サーブレットフィルタ	サーブレットフィルタ(API)	イベントリスナ	イベントリスナ(API)	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ
								クラシックタグハンドラ	シンプルタグハンドラ	
@PostConstruct	−	−	○	−	○	−	−	○	○	×
@PreDestroy	−	−	○	−	○	−	−	○	○	×
@Resource	−	−	○	−	○	−	−	○	○	×
@Resources	−	−	○	−	○	−	−	○	○	×

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

−：標準仕様で対応していない。

(5)　ライブラリ JAR のその他のクラス

ライブラリ JAR のその他のクラスに記述できるアノテーションの一覧を示します。

表 2-5　ライブラリ JAR のその他のクラスに記述できるアノテーション（javax.annotation パッケージ）

アノテーション名	Enterprise Bean					インターセプタ(CDI 対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
@ManagedBean	−	−	−	−	−	−	−	○
@PostConstruct	−	−	−	−	○	○	−	○※2
@PreDestroy	−	−	−	−	○	○	−	○※2
@Priority	−	−	−	−	○	○	−	○
@Resource	−	−	−	−	○※1	○※1	−	○※2
@Resources	−	−	−	−	○※1	×	−	×

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

−：標準仕様で対応していない。

注※1

DI の実行可否は，インターセプタの指定方法やインターセプタメソッドの種類によって非サポートの場合があります。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「15. Interceptors」を参照してください。

注※2

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

2.1.2 javax.annotation.security パッケージに含まれるアノテーションのサポート範囲

javax.annotation.security パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

(1) WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-6 WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）に記述できるアノテーション（javax.annotation.security パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様				例外クラス	Managed Bean（JSF）	その他のクラス
	サーブレット	サーブレット(API)	サーブレットフィルタ	サーブレットフィルタ(API)	イベントリスナ	イベントリスナ(API)	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ			
								クラシックタグハンドラ	シンプルタグハンドラ				
@DeclareRoles	○	○	○	－	○	－	－	－	－	－	－	－	－
@RunAs	○	×	－	－	－	－	－	－	－	－	－	－	－

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

(2) WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-7　WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.annotation.security パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@DeclareRoles	○	○	○	－	－	－	－	－
@RunAs	○	－	－	－	－	－	－	－

（凡例）

○：対応する。

－：標準仕様で対応していない。

(3)　EJB-JAR ファイル（EJB3.1/EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-8　EJB-JAR ファイル（EJB3.1/EJB3.0 対応）に記述できるアノテーション（javax.annotation.security パッケージ）

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@DeclareRoles	－	○	－	×	－	－	－	－
@DenyAll	－	○	－	×	－	－	－	－
@PermitAll	－	○	－	×	－	－	－	－
@RolesAllowed	－	○	－	×	－	－	－	－
@RunAs	－	○	－	×	－	－	－	－

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

(4)　ライブラリ JAR の Servlet/JSP 仕様のクラス

ライブラリ JAR の Servlet/JSP 仕様のクラスに記述できるアノテーションの一覧を示します。

表 2-9　ライブラリ JAR（Servlet/JSP）に記述できるアノテーション（javax.annotation.security パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様			
	サーブレット	サーブレット(API)	サーブレットフィルタ	サーブレットフィルタ(API)	イベントリスナ	イベントリスナ(API)	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ
								クラシックタグハンドラ	シンプルタグハンドラ	
@DeclareRoles	－	－	○	－	○	－	－	－	－	－

（凡例）

○：対応する。

－：標準仕様で対応していない。

(5)　ライブラリ JAR のその他のクラス

ライブラリ JAR のその他のクラスに記述できるアノテーションはありません。

2.1.3　javax.annotation.sql パッケージに含まれるアノテーションのサポート範囲

javax.annotation.sql パッケージのアノテーションのサポート範囲を次の表に示します。

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
javax.annotation.sql	@DataSourceDefinition	×
	@DataSourceDefinitions	×

（凡例）

×：対応しない。

2.1.4　javax.ejb パッケージに含まれるアノテーションのサポート範囲

javax.ejb パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

(1)　WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-10　WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様				インターセプタ（CDI 対象）	例外クラス	ManagedBean（JSF）	その他のクラス
	サーブレット	サーブレット（API）	サーブレットフィルタ	サーブレットフィルタ（API）	イベントリスナ	イベントリスナ（API）	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ				
								クラシックタグハンドラ	シンプルタグハンドラ					
@ApplicationException	－	－	－	－	－	－	－	－	－	－	－	○	－	－
@EJB	○	－	○	－	○	－	－	○	○	×	○※	－	○	○※
@EJBs	○	－	○	－	○	－	－	○	○	×	×	－	○	×

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

注※

DI の実行可否は，インターセプタの指定方法やインターセプタメソッドの種類によって非サポートの場合があります。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「15.　Interceptors」を参照してください。

(2)　WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-11　WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@EJB	○	○	○	－	○	○	×	－
@EJBs	○	○	○	－	○	○	×	－

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

(3)　EJB-JAR ファイル（EJB3.1 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-12　EJB-JAR ファイル（EJB3.1 対応）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Enterprise Bean						インターセプタ（CDI 対象）	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ				
					デフォルトインターセプタ以外	デフォルトインターセプタ			
@AccessTimeout※1	－	○	－	－	－	－	－	－	－
@ActivationConfigProperty	－	－	－	×	－	－	－	－	－
@AfterBegin※2	－	○	－	－	－	－	－	－	－
@AfterCompletion※2	－	○	－	－	－	－	－	－	－

アノテーション名	Enterprise Bean						インターセプタ(CDI対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ				
					デフォルトインターセプタ以外	デフォルトインターセプタ			
@ApplicationException	−	−	−	−	−	−	−	○	−
@Asynchronous※3	−	○	−	−	−	−	−	−	−
@BeforeCompletion※2	−	○	−	−	−	−	−	−	−
@ConcurrencyManagement※1	−	○	−	−	−	−	−	−	−
@DependsOn※1	−	○	−	−	−	−	−	−	−
@EJB	−	○	−	×	○ ※5	○	○ ※5	−	○ ※6
@EJBs	−	○	−	×	○ ※5	○	×	−	×
@Init※2	−	○	−	−	−	−	−	−	−
@Local	○	○	−	−	−	−	−	−	−
@LocalBean	−	○	−	−	−	−	−	−	−
@LocalHome	−	○	−	−	−	−	−	−	−
@Lock※1	−	○	−	−	−	−	−	−	−
@MessageDriven	−	−	−	×	−	−	−	−	−
@PostActivate	−	×	−	−	×	×	−	−	−
@PrePassivate	−	×	−	−	×	×	−	−	−
@Remote	○	○	−	−	−	−	−	−	−
@RemoteHome	−	○	−	−	−	−	−	−	−
@Remove※2	−	○	−	−	−	−	−	−	−

アノテーション名	Enterprise Bean						インターセプタ(CDI対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ				
					デフォルトインターセプタ以外	デフォルトインターセプタ			
@Schedule※3	−	○	−	×	−	−	−	−	−
@Schedules※3	−	○	−	×	−	−	−	−	−
@Singleton※1	−	○	−	−	−	−	−	−	−
@Startup※1	−	○	−	−	−	−	−	−	−
@Stateful※2	−	○	−	−	−	−	−	−	−
@StatefulTimeout	−	×	−	−	−	−	−	−	−
@Stateless※4	−	○	−	−	−	−	−	−	−
@Timeout※3	−	○	−	×	−	−	−	−	−
@TransactionAttribute	−	○	−	×	−	−	−	−	−
@TransactionManagement	−	○	−	×	−	−	−	−	−

(凡例)

○：対応する。

×：アプリケーションサーバでは対応しない。

−：標準仕様で対応していない。

注※1

Singleton Session Bean の場合にだけ使用できます。

注※2

Stateful Session Bean の場合にだけ使用できます。

注※3

Stateless Session Bean と Singleton Session Bean の場合にだけ使用できます。

注※4

Stateless Session Bean の場合にだけ使用できます。

注※5

DI の実行可否は，インターセプタの指定方法やインターセプタメソッドの種類によって非サポートの場合があります。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「15. Interceptors」を参照してください。

注※6

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

(4) EJB-JAR ファイル（EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-13 EJB-JAR ファイル（EJB3.0 対応）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@ActivationConfigProperty	−	−	−	×	−	−	−	−
@ApplicationException	−	−	−	−	−	−	○	−
@EJB	−	○	−	×	○	○	−	−
@EJBs	−	○	−	×	○	○	−	−
@Init※1	−	○	−	−	−	−	−	−
@Local	○	○	−	−	−	−	−	−
@LocalHome	−	○	−	−	−	−	−	−
@MessageDriven	−	−	−	×	−	−	−	−
@PostActivate	−	×	−	−	×	×	−	−
@PrePassivate	−	×	−	−	×	×	−	−
@Remote	○	○	−	−	−	−	−	−

アノテーション名	Enterprise Bean						例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
					デフォルトインターセプタ以外	デフォルトインターセプタ		
@RemoteHome	−	○	−	−	−	−	−	−
@Remove※1	−	○	−	−	−	−	−	−
@Stateful※1	−	○	−	−	−	−	−	−
@Stateless※2	−	○	−	−	−	−	−	−
@Timeout※2	−	○	−	×	−	−	−	−
@TransactionAttribute	−	○	−	×	−	−	−	−
@TransactionManagement	−	○	−	×	−	−	−	−

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

−：標準仕様で対応していない。

注※1

Stateful Session Bean の場合にだけ使用できます。

注※2

Stateless Session Bean の場合にだけ使用できます。

(5)　ライブラリ JAR の Servlet/JSP 仕様のクラス

ライブラリ JAR の Servlet/JSP 仕様のクラスに記述できるアノテーションの一覧を示します。

表 2-14　ライブラリ JAR（Servlet/JSP）に記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様			
	サーブレット	サーブレット(API)	サーブレットフィルタ	サーブレットフィルタ(API)	イベントリスナ	イベントリスナ(API)	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ
								クラシックタグハンドラ	シンプルタグハンドラ	
@EJB	－	－	○	－	○	－	－	○	○	×
@EJBs	－	－	○	－	○	－	－	○	○	×

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

(6)　ライブラリ JAR のその他のクラス

ライブラリ JAR のその他のクラスに記述できるアノテーションの一覧を示します。

表 2-15　ライブラリ JAR のその他のクラスに記述できるアノテーション（javax.ejb パッケージ）

アノテーション名	Enterprise Bean					インターセプタ(CDI 対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
@ApplicationException	－	－	－	－	－	－	○	－
@EJB	－	－	－	－	○※1	○※1	－	○※2
@EJBs	－	－	－	－	○※1	×	－	×
@Local	○	－	－	－	－	－	－	－
@PostActivate	－	－	－	－	×	－	－	－
@PrePassivate	－	－	－	－	×	－	－	－
@Remote	○	－	－	－	－	－	－	－

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

注※1

DI の実行可否は，インターセプタの指定方法やインターセプタメソッドの種類によって非サポートの場合があります。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「15.　Interceptors」を参照してください。

注※2

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

2.1.5 javax.interceptor パッケージに含まれるアノテーションのサポート範囲

javax.interceptor パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

javax.interceptor パッケージのアノテーションは，CDI アプリケーションでも利用できます。ただし，EJB と組み合わせて利用する場合は注意が必要です。注意事項の詳細は，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「9. アプリケーションサーバでの CDI の利用」を参照してください。

(1) WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-16 WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）に記述できるアノテーション（javax.interceptor パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様				インターセプタ(CDI 対象)	例外クラス	ManagedBean（JSF）	その他のクラス
	サーブレット	サーブレット(API)	サーブレットフィルタ	サーブレットフィルタ(API)	イベントリスナ	イベントリスナ(API)	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ				
								クラシックタグハンドラ	シンプルタグハンドラ					
@AroundConstruct	−	−	−	−	−	−	−	−	−	−	○	−	−	−
@AroundInvoke	−	−	−	−	−	−	−	−	−	−	○	−	−	○※
@AroundTimeout	−	−	−	−	−	−	−	−	−	−	×	−	−	−
@ExcludeClassInterceptors	−	−	−	−	−	−	−	−	−	−	−	−	−	○※
@ExcludeDefaultInterceptors	−	−	−	−	−	−	−	−	−	−	−	−	−	○※

アノテーション名	Servlet 仕様						JSP 仕様				インターセプタ（CDI 対象）	例外クラス	ManagedBean（JSF）	その他のクラス
	サーブレット	サーブレット（API）	サーブレットフィルタ	サーブレットフィルタ（API）	イベントリスナ	イベントリスナ（API）	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ				
								クラシックタグハンドラ	シンプルタグハンドラ					
@Interceptor	－	－	－	－	－	－	－	－	－	－	○	－	－	－
@InterceptorBinding	－	－	－	－	－	－	－	－	－	－	－	－	－	○
@Interceptors	－	－	－	－	－	－	－	－	－	－	－	－	－	○※

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

注※

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

(2) WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションはありません。

(3) EJB-JAR ファイル（EJB3.1/EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2-17　EJB-JAR ファイル（EJB3.1/EJB3.0 対応）に記述できるアノテーション（javax.interceptor パッケージ）

アノテーション名	Enterprise Bean						インターセプタ(CDI対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ				
					デフォルトインターセプタ以外	デフォルトインターセプタ			
@AroundConstruct	－	－	－	－	○	×	○	－	－
@AroundInvoke	－	○	－	×	○	○	○	－	○※
@AroundTimeout	－	×	－	×	×	×	×	－	－
@ExcludeClassInterceptors	－	○	－	×	－	－	－	－	○※
@ExcludeDefaultInterceptors	－	○	－	×	－	－	－	－	○※
@Interceptor	－	－	－	－	○	－	○	－	－
@InterceptorBinding	－	－	－	－	－	－	－	－	○
@Interceptors	－	○	－	×	－	－	－	－	○※

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

注※

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

(4)　ライブラリ JAR の Servlet/JSP 仕様のクラス

ライブラリ JAR の Servlet/JSP 仕様のクラスに記述できるアノテーションはありません。

(5)　ライブラリ JAR のその他のクラス

ライブラリ JAR のその他のクラスに記述できるアノテーションの一覧を示します。

表 2-18　ライブラリ JAR のその他のクラスに記述できるアノテーション（javax.interceptor パッケージ）

アノテーション名	Enterprise Bean					インターセプタ (CDI 対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
@AroundConstruct	－	－	－	－	○	○	－	－
@AroundInvoke	－	－	－	－	○	○	－	○※
@AroundTimeout	－	－	－	－	×	×	－	－
@ExcludeClassInterceptors	－	－	－	－	－	－	－	○※
@ExcludeDefaultInterceptors	－	－	－	－	－	－	－	○※
@Interceptor	－	－	－	－	○	○	－	－
@InterceptorBinding	－	－	－	－	－	－	－	○
@Interceptors	－	－	－	－	－	－	－	○※

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

注※

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

2.1.6　javax.jws パッケージに含まれるアノテーションのサポート範囲

javax.jws パッケージに含まれるアノテーションのサポート範囲，および各アノテーションの詳細について，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「16.2　Java から WSDL へのマッピングのカスタマイズ」を参照してください。

2.1.7　javax.persistence パッケージに含まれるアノテーションのサポート範囲

javax.persistence パッケージのアノテーションの適用範囲を説明します。ここでは，JPA コンテナに依存するアノテーションについて，コンポーネントごとに記述できるアノテーションを説明します。この項に記載がないアノテーションについては，JPA 標準仕様および JPA プロバイダの仕様に依存します。各仕様に依存するアノテーションについては，各仕様のドキュメントを参照してください。

(1)　WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-19　WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）に記述できるアノテーション（javax.persistence パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様				インターセプタ (CDI 対象)	例外クラス	ManagedBean(JSF)	その他のクラス
	サーブレット	サーブレット (API)	サーブレットフィルタ	サーブレットフィルタ (API)	イベントリスナ	イベントリスナ (API)	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ				
								クラシックタグハンドラ	シンプルタグハンドラ					
@PersistenceContext	○	－	○	－	○	－	－	○	○	×	○	－	○	○
@PersistenceContexts	○	－	○	－	○	－	－	○	○	×	×	－	○	－
@PersistenceProperty	○	－	○	－	○	－	－	○	○	×	○	－	○	○
@PersistenceUnit	○	－	○	－	○	－	－	○	○	×	○	－	○	○
@PersistenceUnits	○	－	○	－	○	－	－	○	○	×	×	－	○	－

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

－：標準仕様で対応していない。

(2) WAR ファイル（Servlet 2.5 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-20 WAR ファイル（Servlet 2.5 対応）に記述できるアノテーション（javax.persistence パッケージ）

アノテーション名	Servlet 仕様			JSP 仕様				その他のクラス
	サーブレット	サーブレットフィルタ	イベントリスナ	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ	
					クラシックタグハンドラ	シンプルタグハンドラ		
@PersistenceContext	○	○	○	−	○	○	×	−
@PersistenceContexts	○	○	○	−	○	○	×	−
@PersistenceProperty	○	○	○	−	○	○	×	−
@PersistenceUnit	○	○	○	−	○	○	×	−
@PersistenceUnits	○	○	○	−	○	○	×	−

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

−：標準仕様で対応していない。

(3) EJB-JAR ファイル（EJB3.1/EJB3.0 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2–21　EJB-JAR ファイル（EJB3.1/EJB3.0 対応）に記述できるアノテーション（javax.persistence パッケージ）

アノテーション名	Enterprise Bean						インターセプタ(CDI対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ				
					デフォルトインターセプタ以外	デフォルトインターセプタ			
@PersistenceContext	−	○	−	×	○※	○	○	−	○
@PersistenceContexts	−	○	−	×	○※	○	×	−	−
@PersistenceProperty	−	○	−	×	○※	○	○	−	○
@PersistenceUnit	−	○	−	×	○※	○	○	−	○
@PersistenceUnits	−	○	−	×	○※	○	×	−	−

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

−：標準仕様で対応していない。

注※

DI の実行可否は，インターセプタの指定方法やインターセプタメソッドの種類によって非サポートの場合があります。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「15.　Interceptors」を参照してください。

(4)　ライブラリ JAR の Servlet/JSP 仕様のクラス

ライブラリ JAR の Servlet/JSP 仕様のクラスに記述できるアノテーションの一覧を示します。

表 2-22　ライブラリ JAR（Servlet/JSP）に記述できるアノテーション（javax.persistence パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様			
	サーブレット	サーブレット(API)	サーブレットフィルタ	サーブレットフィルタ(API)	イベントリスナ	イベントリスナ(API)	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ
								クラシックタグハンドラ	シンプルタグハンドラ	
@PersistenceContext	−	−	○	−	○	−	−	○	○	×
@PersistenceContexts	−	−	○	−	○	−	−	○	○	×
@PersistenceProperty	−	−	○	−	○	−	−	○	○	×
@PersistenceUnit	−	−	○	−	○	−	−	○	○	×
@PersistenceUnits	−	−	○	−	○	−	−	○	○	×

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

−：標準仕様で対応していない。

(5)　ライブラリ JAR のその他のクラス

ライブラリ JAR のその他のクラスに記述できるアノテーションの一覧を示します。

表 2-23　ライブラリ JAR のその他のクラスに記述できるアノテーション（javax.persistence パッケージ）

アノテーション名	Enterprise Bean					インターセプタ(CDI 対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
@PersistenceContext	−	−	−	−	○※	○	−	○
@PersistenceContexts	−	−	−	−	○※	×	−	−
@PersistenceProperty	−	−	−	−	○※	○	−	○
@PersistenceUnit	−	−	−	−	○※	○	−	○
@PersistenceUnits	−	−	−	−	○※	×	−	−

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

−：標準仕様で対応していない。

注※

DI の実行可否は，インターセプタの指定方法やインターセプタメソッドの種類によって非サポートの場合があります。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「15. Interceptors」を参照してください。

2.1.8 javax.servlet.annotation パッケージに含まれるアノテーションのサポート範囲

javax.servlet.annotation パッケージのアノテーションの適用範囲を説明します。ここでは，コンポーネントごとに記述できるアノテーションを説明します。

(1) WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-24 WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）に記述できるアノテーション（javax.servlet.annotation パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様				例外クラス	Managed Bean(JSF)	その他のクラス
	サーブレット	サーブレット(API)	サーブレットフィルタ	サーブレットフィルタ(API)	イベントリスナ	イベントリスナ(API)	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ			
								クラシックタグハンドラ	シンプルタグハンドラ				
@HandlesTypes	−	−	−	−	−	−	−	−	−	−	−	−	○
@HttpConstraint	○	○	−	−	−	−	−	−	−	−	−	−	−
@HttpMethodConstraint	○	○	−	−	−	−	−	−	−	−	−	−	−
@MultipartConfig	○	○	−	−	−	−	−	−	−	−	−	−	−
@ServletSecurity	○	○	−	−	−	−	−	−	−	−	−	−	−
@WebFilter	−	−	○	−	−	−	−	−	−	−	−	−	−
@WebInitParam	○	−	○	−	−	−	−	−	−	−	−	−	−

アノテーション名	Servlet 仕様						JSP 仕様				例外クラス	Managed Bean(JSF)	その他のクラス
	サーブレット	サーブレット(API)	サーブレットフィルタ	サーブレットフィルタ(API)	イベントリスナ	イベントリスナ(API)	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ			
								クラシックタグハンドラ	シンプルタグハンドラ				
@WebListener	−	−	−	−	○	−	−	−	−	−	−	−	−
@WebServlet	○	−	−	−	−	−	−	−	−	−	−	−	−

（凡例）

○：対応する。

−：標準仕様で対応していない。

(2)　EJB-JAR ファイル

EJB-JAR ファイルに記述できるアノテーションはありません。

(3)　ライブラリ JAR の Servlet/JSP 仕様のクラス

ライブラリ JAR の Servlet/JSP 仕様のクラスに記述できるアノテーションはありません。

(4)　ライブラリ JAR のその他のクラス

ライブラリ JAR のその他のクラスに記述できるアノテーションはありません。

2.1.9　javax.xml.ws パッケージに含まれるアノテーションのサポート範囲

javax.xml.ws パッケージに含まれるアノテーションのサポート範囲，および各アノテーションの詳細については，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「16.2　Java から WSDL へのマッピングのカスタマイズ」を参照してください。

2.1.10　javax.xml.ws.soap パッケージに含まれるアノテーションのサポート範囲

javax.xml.ws.soap パッケージに含まれるアノテーションのサポート範囲，および各アノテーションの詳細については，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「16.2　Java から WSDL へのマッピングのカスタマイズ」を参照してください。

2.1.11 javax.xml.ws.spi パッケージに含まれるアノテーションのサポート範囲

javax.xml.ws.spi パッケージに含まれるアノテーションのサポート範囲，および各アノテーションの詳細については，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「16.2 Java から WSDL へのマッピングのカスタマイズ」を参照してください。

2.1.12 CDI のアノテーションのサポート範囲

CDI のアノテーションのサポート範囲を次の表に示します。

パッケージ	含まれるアノテーション	アプリケーションサーバのサポートの有無
javax.decorator	@Decorator	○
	@Delegate	○
javax.enterprise.context	@ApplicationScoped	○
	@BeforeDestroyed	○
	@ConversationScoped	○
	@Dependent	○
	@Destroyed	○
	@Initialized	○
	@NormalScope	○
	@RequestScoped	○
	@SessionScoped	○
javax.enterprise.context.control	@ActivateRequestContext	○
javax.enterprise.event	@Observes	○
	@ObservesAsync	○
javax.enterprise.inject	@Alternative	○
	@Any	○
	@Decorated	○
	@Default	○
	@Disposes	○
	@Intercepted	○
	@Model	○

パッケージ	含まれるアノテーション	アプリケーションサーバのサポートの有無
	@New	○
	@Produces	○
	@Specializes	○
	@Stereotype	○
	@TransientReference	○
	@Typed	○
	@Vetoed	○
javax.enterprise.inject.spi	@WithAnnotations	×
javax.inject	@Inject	○
	@Named	○
	@Qualifier	○
	@Scope	○
	@Singleton	○

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

ここでは，コンポーネントごとに記述できるアノテーション（@Inject アノテーション）を説明します。なお，@Inject アノテーション以外のアノテーションについては，CDI に依存します。CDI に依存するアノテーションについては，CDI 仕様のドキュメントを参照してください。

(1) WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2–25　WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）に記述できるアノテーション（javax.inject パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様				インターセプタ（CDI 対象）	例外クラス	ManagedBean（JSF）	その他のクラス
	サーブレット	サーブレット（API）	サーブレットフィルタ	サーブレットフィルタ（API）	イベントリスナ	イベントリスナ（API）	JSP ファイル	タグハンドラ		タグライブラリイベントリスナ				
								クラシックタグハンドラ	シンプルタグハンドラ					
@Inject	○	−	○	−	○	−	×	×	×	×	○	×	○	○※

（凡例）

○：対応する。

−：標準仕様で対応していない。

×：アプリケーションサーバでは対応しない。

注※

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

(2)　EJB-JAR ファイル（EJB3.1 対応）

EJB-JAR ファイルに記述できるアノテーションの一覧を示します。

表 2–26　EJB-JAR ファイル（EJB3.1 対応）に記述できるアノテーション（javax.inject パッケージ）

アノテーション名	Enterprise Bean						インターセプタ（CDI 対象）	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ				
					デフォルトインターセプタ以外	デフォルトインターセプタ			
@Inject	×	○	×	×	○	○	○	×	○※

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

注※

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

(3) ライブラリ JAR の Servlet/JSP 仕様のクラス

ライブラリ JAR の Servlet/JSP 仕様のクラスに記述できるアノテーションはありません。

(4) ライブラリ JAR のその他のクラス

ライブラリ JAR のその他のクラスに記述できるアノテーションの一覧を示します。

表 2-27 ライブラリ JAR のその他のクラスに記述できるアノテーション（javax.inject パッケージ)

アノテーション名	Enterprise Bean					インターセプタ (CDI 対象)	例外クラス	その他のクラス
	インタフェース	Session Bean	Entity Bean	Message-driven Bean	インターセプタ			
@Inject	×	×	×	×	○	○	×	○※

（凡例）

○：対応する。

×：アプリケーションサーバでは対応しない。

注※

該当するコンポーネントが CDI の機能を含むコンポーネントの場合だけ使用できます。

2.1.13 JSF のアノテーションのサポート範囲

JSF のアノテーションのサポート範囲を次の表に示します。

パッケージ	含まれるアノテーション	アプリケーションサーバのサポートの有無
javax.faces.annotation	@ApplicationMap	○
	@FacesConfig	○
	@FlowMap	○
	@HeaderMap	○
	@HeaderValuesMap	○

パッケージ	含まれるアノテーション	アプリケーションサーバのサポートの有無
	@InitParameterMap	○
	@ManagedProperty	○
	@RequestCookieMap	○
	@RequestMap	○
	@RequestParameterMap	○
	@RequestParameterValuesMap	○
	@SessionMap	○
	@ViewMap	○
javax.faces.application	@ResourceDependencies	○
	@ResourceDependency	○
javax.faces.bean	@ApplicationScoped	○
	@CustomScoped	○
	@ManagedBean	○
	@ManagedProperty	○
	@NoneScoped	○
	@ReferencedBean	○
	@RequestScoped	○
	@SessionScoped	○
	@ViewScoped	○
javax.faces.component	@FacesComponent	○
javax.faces.component.behavior	@FacesBehavior	○
javax.faces.context	@RequestCookieMap	○
	@SessionMap	○
javax.faces.convert	@FacesConverter	○
javax.faces.event	@ListenerFor	○
	@ListenersFor	○
	@NamedEvent	○
	@WebsocketEvent.Closed	○
	@WebsocketEvent.Opened	○
javax.faces.flow	@FlowScoped	×

パッケージ	含まれるアノテーション	アプリケーションサーバのサポートの有無
javax.faces.flow.builder	@FlowBuilderParameter	×
	@FlowDefinition	×
javax.faces.model	@FacesDataModel	○
javax.faces.push	@Push	○
javax.faces.render	@FacesBehaviorRenderer	○
	@FacesRenderer	○
javax.faces.validator	@FacesValidator	○
javax.faces.view	@ViewScoped	○
javax.faces.view.facelets	@FaceletsResourceResolver	○

（凡例）

○：対応する。

×：対応しない。

ここでは，コンポーネントごとに記述できるアノテーション（@ManagedBean アノテーション）を説明します。なお，@ManagedBean アノテーション以外のアノテーションについては，JSF に依存します。JSF に依存するアノテーションについては，JSF 仕様のドキュメントを参照してください。

(1) WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）

WAR ファイルに記述できるアノテーションの一覧を示します。

表 2-28 WAR ファイル（Servlet 4.0/Servlet 3.1/Servlet 3.0 対応）に記述できるアノテーション（javax.faces.bean パッケージ）

アノテーション名	Servlet 仕様						JSP 仕様				例外クラス	ManagedBean（JSF）	その他のクラス
	サーブレット	サーブレット(API)	サーブレットフィルタ	サーブレットフィルタ(API)	イベントリスナ	イベントリスナ(API)	JSPファイル	タグハンドラ		タグライブラリイベントリスナ			
								クラシックタグハンドラ	シンプルタグハンドラ				
@ManagedBean	－	－	－	－	－	－	－	－	－	－	－	○	－

（凡例）

○：対応する。

－：標準仕様で対応していない。

(2) EJB-JAR ファイル（EJB3.1 対応）

EJB-JAR ファイルに記述できるアノテーションはありません。

(3) ライブラリ JAR の Servlet/JSP 仕様のクラス

ライブラリ JAR の Servlet/JSP 仕様のクラスに記述できるアノテーションはありません。

(4) ライブラリ JAR のその他のクラス

ライブラリ JAR のその他のクラスに記述できるアノテーションはありません。

2.1.14 Bean Validation のアノテーションのサポート範囲

Bean Validation のアノテーションのサポート範囲を次の表に示します。なお，アプリケーションサーバでは，Bean Validation は JSF と CDI から使用できます。

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
javax.validation	@Constraint	○
	@GroupSequence	○
	@OverridesAttribute	○
	@OverridesAttribute.List	○
	@ReportAsSingleViolation	○
	@Valid	○
javax.validation.constraints	@AssertFalse	○
	@AssertFalse.List	○
	@AssertTrue	○
	@AssertTrue.List	○
	@DecimalMax	○
	@DecimalMax.List	○
	@DecimalMin	○
	@DecimalMin.List	○
	@Digits	○
	@Digits.List	○
	@Email	○

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
	@Email.List	○
	@Past	○
	@Past.List	○
	@PastOrPresent	○
	@PastOrPresent.List	○
	@Future	○
	@Future.List	○
	@FutureOrPresent	○
	@FutureOrPresent.List	○
	@Max	○
	@Max.List	○
	@Min	○
	@Min.List	○
	@Negative	○
	@Negative.List	○
	@NegativeOrZero	○
	@NegativeOrZero.List	○
	@NotBlank	○
	@NotBlank.List	○
	@NotEmpty	○
	@NotEmpty.List	○
	@Size	○
	@Size.List	○
	@NotNull	○
	@NotNull.List	○
	@Null	○
	@Null.List	○
	@Pattern	○
	@Pattern.List	○
	@Positive	○

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
	@Positive.List	○
	@PositiveOrZero	○
	@PositiveOrZero.List	○
javax.validation.constraintvalidation	@SupportedValidationTarget	○
javax.validation.executable	@ValidateOnExecution	○
javax.validation.groups	@ConvertGroup	○
	@ConvertGroup.List	○
javax.validation.valueextraction	@ExtractedValue	○
	@UnwrapByDefault	○

（凡例）

○：対応する。

Bean Validation のアノテーションについては，Bean Validation 仕様のドキュメントを参照してください。

Bean Validation のアノテーションの定義可能範囲を次の表に示します。

項番	連携対象	javax.validation パッケージ	javax.validation.constraints パッケージ	サポートバージョン
1	JSF 連携	クラスパス上のクラス	@ManagedBean を指定したクラス	09-00
2	CDI 連携 ユーザアプリケーション	クラスパス上のクラス	JavaBeans クラス※	09-50

注※

JavaBeans クラスのインスタンスをユーザプログラムが管理する場合，そのクラスでは Bean Validation のアノテーションを使用できます。

JavaBeans クラスのインスタンスをコンテナが管理する場合（Servlet/EJB など），そのクラスでは Bean Validation のアノテーションを使用できません。

2.1.15　JAX-RS のアノテーションのサポート範囲

JAX-RS のアノテーションのサポート範囲を次の表に示します。

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
javax.ws.rs	@ApplicationPath	○

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
	@Consumes	○
	@CookieParam	○
	@DefaultValue	○
	@DELETE	○
	@Encoded	○
	@FormParam	○
	@GET	○
	@HEAD	○
	@HeaderParam	○
	@HttpMethod	○
	@MatrixParam	○
	@OPTIONS	○
	@Path	○
	@PathParam	○
	@POST	○
	@Produces	○
	@PUT	○
	@QueryParam	○
	@BeanParam	○
	@ConstrainedTo	○
	@NameBinding	○
javax.ws.rs.container	@PreMatching	○
	@Suspended	○
javax.ws.rs.core	@Context	○
javax.ws.rs.ext	@Provider	○
	@ParamConverter.Lazy	○

（凡例）

○：対応する。

JAX-RS のアノテーションについては，JAX-RS 仕様のドキュメントを参照してください。

2.1.16　Java Batch のアノテーションのサポート範囲

Java Batch のアノテーションのサポート範囲を次の表に示します。

パッケージ	アノテーション	サポートの有無
javax.batch.api	@BatchProperty	○

（凡例）

○：対応する。

JavaBatch のアノテーションについては，JavaBatch 仕様のドキュメントを参照してください。

2.1.17　WebSocket のアノテーションのサポート範囲

WebSocket のアノテーションのサポート範囲を次の表に示します。

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
javax.websocket	@ClientEndpoint	○
	@OnClose	○
	@OnError	○
	@OnMessage	○
	@OnOpen	○
javax.websocket.server	@PathParam	○
	@ServerEndpoint	○

（凡例）

○：対応する。

WebSocket のアノテーションについては，WebSocket 仕様のドキュメントを参照してください。

2.1.18　JTA のアノテーションのサポート範囲

JTA のアノテーションのサポート範囲を次の表に示します。

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
javax.transaction	@Transactional	○
	@TransactionScoped	×

（凡例）

○：対応する。

×：対応しない。

JTA のアノテーションについては，JTA 仕様のドキュメントを参照してください。

2.1.19　JMS のアノテーションのサポート範囲

JMS のアノテーションのサポート範囲を次の表に示します。

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
javax.jms	@JMSConnectionFactory	×
	@JMSConnectionFactoryDefinition	×
	@JMSConnectionFactoryDefinitions	×
	@JMSDestinationDefinition	×
	@JMSDestinationDefinitions	×
	@JMSPasswordCredential	×
	@JMSSessionMode	×

（凡例）

×：対応しない。

JMS のアノテーションについては，JMS 仕様のドキュメントを参照してください。

2.1.20　JCA のアノテーションのサポート範囲

JCA のアノテーションのサポート範囲を次の表に示します。

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
javax.resource	@AdministeredObjectDefinition	×
	@AdministeredObjectDefinitions	×
	@ConnectionFactoryDefinition	×
	@ConnectionFactoryDefinitions	×
javax.resource.spi	@Activation	×
	@AdministeredObject	×
	@AuthenticationMechanism	×

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
	@ConfigProperty	×
	@ConnectionDefinition	×
	@ConnectionDefinitions	×
	@Connector	×
	@SecurityPermission	×

（凡例）

×：対応しない。

JCA のアノテーションについては，JCA 仕様のドキュメントを参照してください。

2.1.21　JSON-B のアノテーションのサポート範囲

JSON-B のアノテーションのサポート範囲を次の表に示します。

パッケージ	アノテーション	アプリケーションサーバのサポートの有無
javax.json.bind.annotation	@JsonbAnnotation	○
	@JsonbCreator	○
	@JsonbDateFormat	○
	@JsonbNillable	○
	@JsonbNumberFormat	○
	@JsonbProperty	○
	@JsonbPropertyOrder	○
	@JsonbTransient	○
	@JsonbTypeAdapter	○
	@JsonbTypeDeserializer	○
	@JsonbTypeSerializer	○
	@JsonbVisibility	○

（凡例）

○：対応する。

JSON-B のアノテーションについては，JSON-B 仕様のドキュメントを参照してください。

2.2　javax.annotation パッケージ

javax.annotation パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@PostConstruct	サーブレット，Enterprise Bean インスタンスなどが生成された直後にコールバックするメソッドを設定します。
@PreDestroy	サーブレット，Enterprise Bean インスタンスなどが削除される直前にコールバックするメソッドを設定します。
@Resource	リソースへの参照を宣言します。
@Resources	@Resource を複数設定します。

それぞれのアノテーションの詳細について，次に説明します。

2.2.1　@PostConstruct

(1)　説明

サーブレット，Enterprise Bean インスタンスなどが生成された直後にコールバックするメソッドを設定します。

(2)　属性

@PostConstruct の属性はありません。

2.2.2　@PreDestroy

(1)　説明

サーブレット，Enterprise Bean インスタンスなどが削除される直前にコールバックするメソッドを設定します。

(2)　属性

@PreDestroy の属性はありません。

2.2.3　@Resource

(1)　説明

リソースへの参照を宣言します。クラス，メソッド，およびフィールドに設定できます。メソッドやフィールドに設定した場合，Dependency Injection の対象となります。ただし，メソッドは set メソッドである必要があります。

(2)　属性

@Resource の属性の一覧を次の表に示します。

属性名	機能
name	リソース参照の名称を設定します。設定した名称は JNDI 名として使用されます。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。
type	リソースの Java タイプを設定します。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。
authenticationType	リソースに使用する認証タイプを設定します。
shareable	リソースを共用するかどうかを設定します。
mappedName	参照先リソースを特定するためにリソース表示名やキュー名を設定します。
lookup	参照する別のリソース参照の Portable Global JNDI 名，またはリソースの別名を設定します。
description	リソースの説明を設定します。

各属性の詳細を次に示します。

(a)　name 属性

型

String

説明

リソース参照の名称を設定します。設定した名称は JNDI 名として使用されます。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。

なお，リソースの別名を指定することもできます。J2EE リソースの別名の設定については，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「2.6.6　J2EE リソースの別名の設定」を参照してください。

デフォルト値

- メソッドに設定した場合

  アノテーションを設定したクラス名/set メソッドのプロパティ

- フィールドに設定した場合
  アノテーションを設定したクラス名/フィールド名

(b) type 属性

型

Class

説明

リソースの Java タイプを設定します。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。

デフォルト値

- メソッド設定した場合
  メソッドの引数の型
- フィールドに設定した場合
  フィールドの型

type 属性と DD の対応

type 属性は J2EE 仕様と異なり，設定値（Java Type）によって対応する DD が変わります。Java Type によって異なる DD の対応を次の表に示します。

表 2-29　type 属性による DD の対応表

type 属性	J2EE 仕様で対応する DD のタグ	アプリケーションサーバ仕様で対応する DD のタグ※1
java.lang.String※2	env-entry	env-entry
java.lang.Character※2	env-entry	env-entry
java.lang.Integer※2	env-entry	env-entry
java.lang.Boolean※2	env-entry	env-entry
java.lang.Double※2	env-entry	env-entry
java.lang.Byte※2	env-entry	env-entry
java.lang.Short※2	env-entry	env-entry
java.lang.Long※2	env-entry	env-entry
java.lang.Float※2	env-entry	env-entry
javax.xml.rpc.Service	service-ref	例外※3
javax.xml.ws.Service	service-ref	例外※3
javax.jws.WebService	service-ref	例外※3
javax.sql.DataSource	resource-ref	resource-ref

type 属性	J2EE 仕様で対応する DD のタグ	アプリケーションサーバ仕様で対応する DD のタグ※1
javax.jms.ConnectionFactory	resource-ref	resource-ref
javax.jms.QueueConnectionFactory	resource-ref	resource-ref
javax.jms.TopicConnectionFactory	resource-ref	resource-ref
javax.mail.Session	resource-ref	resource-ref
java.net.URL	resource-ref	例外※3
javax.resource.cci.ConnectionFactory	resource-ref	resource-ref
org.omg.CORBA_2_3.ORB	resource-ref	resource-ref
リソースアダプタによって定義されるほかのコネクションファクトリ	resource-ref	resource-env-ref
javax.jms.Queue	message-destination-ref	resource-env-ref
javax.jms.Topic	message-destination-ref	resource-env-ref
javax.resource.cci.InteractionSpec	resource-env-ref	例外※3
javax.transaction.UserTransaction	resource-env-ref	resource-env-ref
javax.xml.ws.WebServiceContext	未定義	resource-env-ref※4
上記以外のすべてのタイプ※5	resource-env-ref	resource-env-ref

注※1

mappedName 要素に「!#」が含まれていた場合は，Java Type とは関係なく，<resource-env-ref>タグに対応づけます。

注※2

標準 DD から値を取得できないため，属性ファイルには表示しますが，DI は行いません。

注※3

インポート時に例外となります。

注※4

08-70 より前のアプリケーションサーバでは，上記以外のすべてのタイプとして扱われます。

注※5

09-00 以降のアプリケーションサーバでは，java.lang.Class と java.lang.Enum のサブクラスは<env-entry>で扱われません。

(c) authenticationType 属性

型

AuthenticationType

説明

リソースに使用する認証タイプを設定します。

デフォルト値

CONTAINER

(d) shareable 属性

型

boolean

説明

リソースを共用するかどうかを設定します。

デフォルト値

true

(e) mappedName 属性

型

String

説明

参照先リソースを特定するためにリソース表示名やキュー名を設定します。

リソース表示名に半角英数字およびアンダースコア(_)以外の文字を含む場合，半角英数字およびアンダースコア(_)以外の文字をアンダースコア(_)に置き換えて設定してください。

デフォルト値

""

mappedName 属性の設定条件

mappedName 属性は type 属性によって設定条件が変わります。@Resource での mappedName 属性の設定条件を次の表に示します。

表 2-30 @Resource の mappedName()の設定条件

設定条件（Java Type，リソースなど）	使用可否※1
java.lang.String	×
java.lang.Character	×
java.lang.Integer	×
java.lang.Boolean	×
java.lang.Double	×
java.lang.Byte	×
java.lang.Short	×
java.lang.Long	×
java.lang.Float	×

設定条件（Java Type，リソースなど）	使用可否※1
javax.xml.rpc.Service	×
javax.sql.DataSource	○
javax.jms.ConnectionFactory	○
javax.jms.QueueConnectionFactory	○
javax.jms.TopicConnectionFactory	○
javax.mail.Session	○
java.net.URL	×
javax.resource.cci.ConnectionFactory	○
org.omg.CORBA_2_3.ORB	×
javax.jms.Queue※2	○
javax.jms.Topic	○
javax.resource.cci.InteractionSpec	×
javax.transaction.UserTransaction	×
javax.ejb.EjbContext	×
javax.ejb.SessionContext	×
javax.ejb.TimerService	×
JavaBeans リソース	○

（凡例）

○：使用できます。

×：使用できません。

注※1

管理対象オブジェクトへの対応づけは，Java Type に関係なく，mappedName 要素で対応づけます。リソースアダプタの表示名と管理対象オブジェクト名の区切り文字には，「!#」を使用してください。

注※2

TP1/Message Queue - Access または Reliable Messaging を使用時に javax.jms.Queue を使用する場合，リソースアダプタの表示名とキューの表示名の区切り文字には，「#」を使用してください。

(f) lookup 属性

型

String

説明

参照する別のリソース参照の Portable Global JNDI 名，またはリソースの別名を設定します。

デフォルト値

""

(g) description 属性

型

String

説明

リソースの説明を設定します。

デフォルト値

""

2.2.4 @Resources

(1) 説明

@Resource を複数設定します。なお，クラスにだけ設定できます。

(2) 属性

@Resources の属性の一覧を次の表に示します。

属性名	機能
value	複数のリソース（@Resource）を定義します。

各属性の詳細を次に示します。

(a) value 属性

型

Resource[]

説明

複数のリソース（@Resource）を定義します。

デフォルト値

なし

2.3　javax.annotation.security パッケージ

javax.annotation.security パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@DeclareRoles	セキュリティロールの参照を設定します。
@DenyAll	すべてのセキュリティロールに対して，アクセスを拒否するメソッドに設定します。
@PermitAll	すべてのセキュリティロールに対して，アクセスを許可するクラスまたはメソッドに設定します。
@RolesAllowed	クラスまたはメソッドに対してアクセスを許可するセキュリティロールを設定します。
@RunAs	サーブレット，または Enterprise Bean を実行する際に適用するセキュリティロールを設定します。

2.3.1　@DeclareRoles

(1)　説明

セキュリティロールの参照を設定します。なお，クラスにだけ設定できます。

(2)　属性

@DeclareRoles の属性の一覧を次の表に示します。

属性名	機能
value	参照するセキュリティロール名を設定します。

各属性の詳細を次に示します。

(a)　value 属性

型

　String[]

説明

　参照するセキュリティロール名を設定します。

デフォルト値

　なし

2.3.2 @DenyAll

(1) 説明

すべてのセキュリティロールに対して，アクセスを拒否するメソッドまたはクラスに設定します。

(2) 属性

@DenyAll の属性はありません。

2.3.3 @PermitAll

(1) 説明

すべてのセキュリティロールに対して，アクセスを許可するクラスまたはメソッドに設定します。

(2) 属性

@PermitAll の属性はありません。

2.3.4 @RolesAllowed

(1) 説明

クラスまたはメソッドに対してアクセスを許可するセキュリティロールを設定します。

(2) 属性

@RolesAllowed の属性の一覧を次の表に示します。

属性名	機能
value	アプリケーションでのメソッドにアクセスするのが許可されたロールリストを設定します。

各属性の詳細を次に示します。

(a) value 属性

型

String[]

説明

アプリケーションでのメソッドにアクセスするのが許可されたロールリストを設定します。

デフォルト値

なし

2.3.5 @RunAs

(1) 説明

サーブレット，または Enterprise Bean を実行する際に適用するセキュリティロールを設定します。なお，クラスにだけ設定できます。

(2) 属性

@RunAs の属性の一覧を次の表に示します。

属性名	機能
value	Enterprise Bean を実行する際に適用するセキュリティロール名を設定します。

各属性の詳細を次に示します。

(a) value 属性

型

String

説明

サーブレット，または Enterprise Bean を実行する際に適用するセキュリティロール名を設定します。

デフォルト値

なし

2.4 javax.ejb パッケージ

javax.ejb パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@AccessTimeout	Container Managed Concurrency が設定された Singleton Session Bean の，同時アクセスのタイムアウト値を設定します。
@AfterBegin	Stateful Session Bean の，トランザクション開始直後にコールバックされるメソッドに設定します。
@AfterCompletion	Stateful Session Bean の，トランザクション決着後にコールバックされるメソッドに設定します。
@ApplicationException	アプリケーション例外とする例外クラスに設定します。
@Asynchronous	非同期で実行するビジネスメソッドに設定します。 Stateless Session Bean または Singleton Session Bean のクラス，メソッドに設定します。
@BeforeCompletion	Stateful Session Bean の，トランザクション決着直前にコールバックされるメソッドに設定します。
@ConcurrencyManagement	Singleton Session Bean の ConcurrencyManagement の種類を設定します。 Singleton Session Bean のクラスにだけ設定します。
@DependsOn	Singleton Session Bean 同士の依存関係を指定するために設定します。Singleton Session Bean のクラスにだけ設定します。
@EJB	EJB のビジネスインタフェースまたはホームインタフェースへの参照を設定します。
@EJBs	@EJB を複数設定します。
@Init	Stateful Session Bean のホームインタフェースで定義した create<METHOD>()を実行した際，コールバックするメソッドに設定します。
@Local	Enterprise Bean のローカルビジネスインタフェースを設定します。
@LocalBean	Session Bean を No-Interface view として指定する場合に設定します。Session Bean のクラスにだけ設定します。
@LocalHome	ローカルホームインタフェース，およびローカルコンポーネントインタフェースを使用した呼び出しをサポートする Enterprise Bean のクラスに設定します。
@Lock	Container Managed Concurrency が設定された Singleton Session Bean の，ビジネスメソッドへのアクセス時の排他制御の方法を設定します。
@PostActivate	Stateful Session Bean が活性化された直後にコールバックするメソッドに設定します。
@PrePassivate	Stateful Session Bean が非活性化される直前にコールバックするメソッドに設定します。
@Remote	Enterprise Bean のリモートビジネスインタフェースを設定します。アノテーションをインタフェースに設定した場合，そのインタフェースがリモートビジネスインタフェースとなります。

アノテーション名	機能
@RemoteHome	リモートホームインタフェース，およびリモートコンポーネントインタフェースを使用した呼び出しをサポートする Enterprise Bean のクラスに設定します。
@Remove	Stateful Session Bean を削除する働きを持つビジネスメソッドに設定します。
@Schedule	EJB タイマーサービスの，カレンダーベースの自動生成タイマーがコールバックされるタイムアウトメソッドに設定します。
@Schedules	@Schedule を複数設定します。コールバックされるタイムアウトメソッドに設定します。
@Singleton	Singleton Session Bean のクラスに設定します。
@Startup	アプリケーション開始時に Singleton Session Bean を同時に開始する場合に設定します。Singleton Session Bean のクラスに設定します。
@Stateful	Stateful Session Bean のクラスに設定します。
@Stateless	Stateless Session Bean のクラスに設定します。
@Timeout	TimerService 使用時にコールバックするタイムアウトメソッドに設定します。
@TransactionAttribute	Enterprise Bean が CMT で動作する場合のトランザクション属性を設定します。
@TransactionManagement	Enterprise Bean のトランザクション管理種別を設定します。

2.4.1 @AccessTimeout

(1) 説明

Container Managed Concurrency が設定された Singleton Session Bean の，同時アクセスのタイムアウト値を設定します。

(2) 属性

@AccessTimeout の属性の一覧を次の表に示します。

属性名	機能
value	タイムアウト値を設定します。
unit	タイムアウト値の単位を設定します。

各属性の詳細を次に示します。

(a) value 属性

型

long

説明

タイムアウト値を設定します。

デフォルト値

なし

(b) unit 属性

型

TimeUnit

説明

タイムアウト値の単位を設定します。

デフォルト値

MILLISECONDS

2.4.2 @AfterBegin

(1) 説明

Stateful Session Bean の，トランザクション開始直後にコールバックされるメソッドに設定します。

(2) 属性

@AfterBegin の属性はありません。

2.4.3 @AfterCompletion

(1) 説明

Stateful Session Bean の，トランザクション決着後にコールバックされるメソッドに設定します。

(2) 属性

@AfterCompletion の属性はありません。

2.4.4 @ApplicationException

(1) 説明

アプリケーション例外とする例外クラスに設定します。

(2) 属性

@ApplicationException の属性の一覧を次の表に示します。

属性名	機能
rollback	例外発生時にコンテナがトランザクションをロールバックするかどうかを設定します。
inherited	アプリケーション例外とするかどうかについて，クラスで設定されている定義をサブクラスにも適用するかどうかを設定します。

各属性の詳細を次に示します。

(a) rollback 属性

型

boolean

説明

例外発生時にコンテナがトランザクションをロールバックするかどうかを設定します。

デフォルト値

false

(b) inherited 属性

型

boolean

説明

アプリケーション例外とするかどうかについて，クラスで設定されている定義をサブクラスにも適用するかどうかを設定します。

デフォルト値

true

2.4.5 @Asynchronous

(1) 説明

非同期で実行するビジネスメソッドに設定します。Stateless Session Bean，または Singleton Session Bean のクラスおよびメソッドに設定します。

(2) 属性

@Asynchronous の属性はありません。

2.4.6 @BeforeCompletion

(1) 説明

Stateful Session Bean の，トランザクション決着直前にコールバックされるメソッドに設定します。

(2) 属性

@BeforeCompletion の属性はありません。

2.4.7 @ConcurrencyManagement

(1) 説明

Singleton Session Bean の ConcurrencyManagement の種類を設定します。Singleton Session Bean のクラスにだけ設定します。

(2) 属性

@ConcurrencyManagement の属性の一覧を次の表に示します。

属性名	機能
value	Singleton Session Bean の ConcurrencyManagement の種類を設定します。

各属性の詳細を次に示します。

(a) value 属性

型

ConcurrencyManagementType

説明

Singleton Session Bean の ConcurrencyManagement の種類を設定します。

デフォルト値

CONTAINER

2.4.8 @DependsOn

(1) 説明

Singleton Session Bean 同士の依存関係を指定するために設定します。Singleton Session Bean のクラスにだけ設定します。

(2) 属性

@DependsOn の属性の一覧を次の表に示します。

属性名	機能
value	依存する Singleton Session Bean の EJB 名を列挙します。

各属性の詳細を次に示します。

(a) value 属性

型

String[]

説明

依存する Singleton Session Bean の EJB 名を列挙します。

デフォルト値

なし

2.4.9 @EJB

(1) 説明

EJB のビジネスインタフェースまたはホームインタフェースへの参照を設定します。クラス，メソッド，およびフィールドに設定できます。メソッドやフィールドに設定した場合，Dependency Injection の対象となります。ただし，メソッドは set メソッドである必要があります。

(2) 属性

@EJB の属性の一覧を次の表に示します。

属性名	機能
name	リソース参照の名称を設定します。設定した名称は JNDI 名として使用されます。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。
beanInterface	ビジネスインタフェースまたはホームインタフェースのクラスを設定します。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。
beanName	参照する EJB のパッケージなしクラス名を設定します。ただし，参照する EJB クラスを定義するアノテーション（@Stateless，@Stateful，@Singleton）に name 属性が設定されている場合，name 属性の値を設定します。また，DD による定義をサポートする EJB の場合，DD の<ejb-name>タグの値を設定します。
mappedName	参照する EJB の Portable Global JNDI 名，または EJB の別名を設定します。ただし，beanName 属性が設定されている場合，beanName 属性の設定が優先されます。
lookup	参照する EJB の Portable Global JNDI 名，または EJB の別名を設定します。ただし，beanName 属性や mappedName 属性が設定されている場合，beanName 属性や mappedName 属性の設定が優先されます。
description	参照する EJB の説明を設定します。

各属性の詳細を次に示します。

(a) name 属性

型

String

説明

リソース参照の名称を設定します。設定した名称は JNDI 名として使用されます。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。

デフォルト値

- メソッドに設定した場合
  アノテーションを設定したクラス名/set メソッドのプロパティ
- フィールドに設定した場合
  アノテーションを設定したクラス名/フィールド名

(b) beanInterface 属性

型

Class

説明

ビジネスインタフェースまたはホームインタフェースのクラスを設定します。アノテーションをメソッドまたはフィールドに設定する場合，省略できます。

デフォルト値

- メソッド設定した場合
  メソッドの引数の型
- フィールドに設定した場合
  フィールドの型

(c) beanName 属性

型

String

説明

参照する EJB のパッケージなしクラス名を設定します。ただし，参照する EJB クラスを定義するアノテーション（@Stateless，@Stateful，@Singleton）に name 属性が設定されている場合，name 属性の値を設定します。また，DD による定義をサポートする EJB の場合，DD の<ejb-name>タグの値を設定します。

デフォルト値

""

(d) mappedName 属性

型

String

説明

参照する EJB の Portable Global JNDI 名，または EJB の別名を設定します。ただし，beanName 属性が設定されている場合，beanName 属性の設定が優先されます。

デフォルト値

""

(e) lookup 属性

型

String

説明

参照する EJB の Portable Global JNDI 名，または EJB の別名を設定します。ただし，beanName 属性や mappedName 属性が設定されている場合，beanName 属性や mappedName 属性の設定が優先されます。

デフォルト値

""

(f) description 属性

型

String

説明

参照する EJB の説明を設定します。

デフォルト値

""

2.4.10 @EJBs

(1) 説明

@EJB を複数設定します。なお，クラスにだけ設定できます。

(2) 属性

@EJBs の属性の一覧を次の表に示します。

属性名	機能
value	@EJB を設定します。

各属性の詳細を次に示します。

(a) value 属性

型

EJB[]

説明

@EJB を設定します。

デフォルト値

なし

2.4.11　@Init

(1)　説明

Stateful Session Bean のホームインタフェースで定義した create<METHOD>()を実行した際，コールバックするメソッドに設定します。

(2)　属性

@Init の属性の一覧を次の表に示します。

属性名	機能
value	対応する create<METHOD>()名を設定します。

各属性の詳細を次に示します。

(a)　value 属性

型

String

説明

対応する create<METHOD>()名を設定します。

デフォルト値

""

2.4.12　@Local

(1)　説明

Enterprise Bean のローカルビジネスインタフェースを設定します。アノテーションをインタフェースに設定した場合，そのインタフェースがローカルビジネスインタフェースとなります。Bean クラスに設定した場合，value 属性にローカルビジネスインタフェースを設定する必要があります。

(2)　属性

@Local の属性の一覧を次の表に示します。

属性名	機能
value	アノテーションを Bean クラスに設定した場合，ローカルビジネスインタフェースのクラスを設定します。

各属性の詳細を次に示します。

(a) value 属性

型

Class[]

説明

アノテーションを Bean クラスに設定した場合，ローカルビジネスインタフェースのクラスを設定します。

デフォルト値

{}

2.4.13 @LocalBean

(1) 説明

Session Bean を No-Interface view として指定する場合に設定します。Session Bean のクラスにだけ設定します。

(2) 属性

@LocalBean の属性はありません。

2.4.14 @LocalHome

(1) 説明

ローカルホームインタフェース，およびローカルコンポーネントインタフェースを使用した呼び出しをサポートする Enterprise Bean のクラスに設定します。

(2) 属性

@LocalHome の属性の一覧を次の表に示します。

属性名	機能
value	ローカルホームインタフェースを設定します。

各属性の詳細を次に示します。

(a) value 属性

型

Class

説明

ローカルホームインタフェースを設定します。

デフォルト値

なし

2.4.15 @Lock

(1) 説明

Container Managed Concurrency が設定された Singleton Session Bean の，ビジネスメソッドへのアクセス時の排他制御の方法を設定します。

(2) 属性

@Lock の属性の一覧を次の表に示します。

属性名	機能
value	ビジネスメソッドへのアクセス時に，同時アクセスを許可する（READ）か，許可しない（WRITE）かを設定します。

各属性の詳細を次に示します。

(a) value 属性

型

LockType

説明

ビジネスメソッドへのアクセス時に，同時アクセスを許可する（READ）か，許可しない（WRITE）かを設定します。

デフォルト値

WRITE

2.4.16　@PostActivate

(1)　説明

Stateful Session Bean が活性化された直後にコールバックするメソッドに設定します。アノテーションを設定できますが，活性化，非活性化の状態変化をサポートしないため，動作しません。

(2)　属性

@PostActivate の属性はありません。

2.4.17　@PrePassivate

(1)　説明

Stateful Session Bean が非活性化される直前にコールバックするメソッドに設定します。アノテーションを設定できますが，活性化，非活性化の状態変化をサポートしないため，動作しません。

(2)　属性

@PrePassivate の属性はありません。

2.4.18　@Remote

(1)　説明

Enterprise Bean のリモートビジネスインタフェースを設定します。アノテーションをインタフェースに設定した場合，そのインタフェースがリモートビジネスインタフェースとなります。Bean クラスに設定した場合，value 属性にリモートビジネスインタフェースを設定する必要があります。

(2)　属性

@Remote の属性の一覧を次の表に示します。

属性名	機能
value	アノテーションを Bean クラスに設定した場合，リモートビジネスインタフェースのクラスを設定します。

各属性の詳細を次に示します。

(a) value 属性

型

Class[]

説明

アノテーションを Bean クラスに設定した場合，リモートビジネスインタフェースのクラスを設定します。

デフォルト値

{}

2.4.19 @RemoteHome

(1) 説明

リモートホームインタフェース，およびリモートコンポーネントインタフェースを使用した呼び出しをサポートする Enterprise Bean のクラスに設定します。

(2) 属性

@RemoteHome の属性の一覧を次の表に示します。

属性名	機能
value	リモートホームインタフェースを設定します。

各属性の詳細を次に示します。

(a) value 属性

型

Class

説明

リモートホームインタフェースを設定します。

デフォルト値

なし

2.4.20　@Remove

(1)　説明

Stateful Session Bean を削除する働きを持つビジネスメソッドに設定します。

(2)　属性

@Remove の属性の一覧を次の表に示します。

属性名	機能
retainIfException	メソッドがアプリケーション例外で異常終了した場合に削除するかどうかを設定します。

各属性の詳細を次に示します。

(a)　retainIfException 属性

型

　boolean

説明

　メソッドがアプリケーション例外で異常終了した場合に削除するかどうかを設定します。

デフォルト値

　false

2.4.21　@Schedule

(1)　説明

EJB タイマーサービスの，カレンダーベースの自動生成タイマーがコールバックされるタイムアウトメソッドに設定します。

(2)　属性

@Schedule の属性の一覧を次の表に示します。

属性名	機能
dayOfMonth	タイムアウトする日を設定します。
dayOfWeek	タイムアウトする曜日を設定します。
hour	タイムアウトする時を設定します。

属性名	機能
info	タイマーと結び付く任意の文字情報を設定します。
minute	タイムアウトする分を設定します。
month	タイムアウトする月を設定します。
persistent	タイマーの永続化を設定します。タイマーの永続化機能はサポートしないため，true を設定してもタイマーは永続されないで，非永続（false）として動作します。
second	タイムアウトする秒を設定します。
timezone	タイムアウトするタイムゾーンを設定します。
year	タイムアウトする年を設定します。

各属性の詳細を次に示します。

(a) dayOfMonth 属性

型

String

説明

タイムアウトする日を設定します。

デフォルト値

"*"

(b) dayOfWeek 属性

型

String

説明

タイムアウトする曜日を設定します。

デフォルト値

"*"

(c) hour 属性

型

String

説明

タイムアウトする時を設定します。

デフォルト値

"0"

(d) info 属性

型

String

説明

タイマーと結び付く任意の文字情報を設定します。

デフォルト値

""

(e) minute 属性

型

String

説明

タイムアウトする分を設定します。

デフォルト値

"0"

(f) month 属性

型

String

説明

タイムアウトする月を設定します。

デフォルト値

"*"

(g) persistent 属性

型

boolean

説明

タイマーの永続化を設定します。タイマーの永続化機能はサポートしないため，true を設定してもタイマーは永続されないで，非永続（false）として動作します。

デフォルト値

true

(h) second 属性

型

String

説明

タイムアウトする秒を設定します。

デフォルト値

"0"

(i) timezone 属性

型

String

説明

タイムアウトするタイムゾーンを設定します。

デフォルト値

""

(j) year 属性

型

String

説明

タイムアウトする年を設定します。

デフォルト値

"*"

2.4.22 @Schedules

(1) 説明

@Schedule を複数設定します。コールバックされるタイムアウトメソッドに設定します。

(2) 属性

@Schedules の属性の一覧を次の表に示します。

属性名	機能
value	@Schedule を設定します。

各属性の詳細を次に示します。

(a) value 属性

型

Schedule[]

説明

@Schedule を設定します。

デフォルト値

なし

2.4.23 @Singleton

(1) 説明

Singleton Session Bean のクラスに設定します。

(2) 属性

@Singleton の属性の一覧を次の表に示します。

属性名	機能
name	Singleton Session Bean の名称を設定します。
mappedName	Singleton Session Bean の別名を設定します。
description	Singleton Session Bean の説明を設定します。

各属性の詳細を次に示します。

(a) name 属性

型

String

説明

Singleton Session Bean の名称を設定します。

デフォルト値

Singleton Session Bean のパッケージ名を除いたクラス名

(b) mappedName 属性

型

String

説明

Singleton Session Bean の別名を設定します。

デフォルト値

""

(c) description 属性

型

String

説明

Singleton Session Bean の説明を設定します。

デフォルト値

""

2.4.24 @Startup

(1) 説明

アプリケーション開始時に Singleton Session Bean を同時に開始する場合に設定します。Singleton Session Bean のクラスに設定します。

(2) 属性

@Startup の属性はありません。

2.4.25 @Stateful

(1) 説明

Stateful Session Bean のクラスに設定します。

(2) 属性

@Stateful の属性の一覧を次の表に示します。

属性名	機能
name	Stateful Session Bean の名称を設定します。
mappedName	Stateful Session Bean の別名を設定します。
description	Stateful Session Bean の説明を設定します。

各属性の詳細を次に示します。

(a) name 属性

型

String

説明

Stateful Session Bean の名称を設定します。

デフォルト値

Stateful Session Bean のパッケージを除いたクラス名

(b) mappedName 属性

型

String

説明

Stateful Session Bean の別名を設定します。

デフォルト値

""

(c) description 属性

型

String

説明

Stateful Session Bean の説明を設定します。

デフォルト値

""

2.4.26　@Stateless

(1)　説明

Stateless Session Bean のクラスに設定します。

(2)　属性

@Stateless の属性の一覧を次の表に示します。

属性名	機能
name	Stateless Session Bean の名称を設定します。
mappedName	Stateless Session Bean の別名を設定します。
description	Stateless Session Bean の説明を設定します。

各属性の詳細を次に示します。

(a)　name 属性

型

String

説明

Stateless Session Bean の名称を設定します。

デフォルト値

Stateless Session Bean のパッケージを除いたクラス名

(b)　mappedName 属性

型

String

説明

Stateless Session Bean の別名を設定します。

デフォルト値

""

(c)　description 属性

型

String

説明

Stateless Session Bean の説明を設定します。

デフォルト値

""

2.4.27 @Timeout

(1) 説明

TimerService 使用時にコールバックするタイムアウトメソッドに設定します。

(2) 属性

@Timeout の属性はありません。

2.4.28 @TransactionAttribute

(1) 説明

Enterprise Bean が CMT で動作する場合のトランザクション属性を設定します。クラス，およびメソッドに設定できます。

(2) 属性

@TransactionAttribute の属性の一覧を次の表に示します。

属性名	機能
value	トランザクション属性を設定します。

各属性の詳細を次に示します。

(a) value 属性

型

TransactionAttributeType

説明

トランザクション属性を設定します。

デフォルト値

REQUIRED

2.4.29　@TransactionManagement

(1)　説明

Enterprise Bean のトランザクション管理種別を設定します。なお，クラスにだけ設定できます。

(2)　属性

@TransactionManagement の属性の一覧を次の表に示します。

属性名	機能
value	トランザクション管理種別を設定します。

各属性の詳細を次に示します。

(a)　value 属性

型

　TransactionManagementType

説明

　トランザクション管理種別を設定します。

デフォルト値

　CONTAINER

2.5　javax.faces.bean パッケージ

javax.faces.bean パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@ManagedBean	JSF が使用する Managed Bean を設定します。

これ以外のアノテーションについては，JSF 仕様のドキュメントを参照してください。

2.5.1　@ManagedBean

(1)　説明

JSF が使用する Managed Bean を設定します。

(2)　属性

@ManagedBean の属性の一覧を次の表に示します。

属性名	機能
eager	Managed Bean を Web アプリケーション開始時に生成するかどうかを設定します。
name	ManagedBean の名前を設定します。

各属性の詳細を次に示します。

(a)　eager 属性

型

boolean

説明

Managed Bean を Web アプリケーション開始時に生成するかどうかを設定します。true を設定した場合，Bean のスコープをアプリケーションスコープにする必要があります。

デフォルト値

false

(b)　name 属性

型

String

説明

ManagedBean の名前を設定します。

未設定または空文字を設定した場合，アノテーションを指定したクラスのクラス名の先頭を小文字にした名前が使用されます。

例：java.examlpes.Bean の場合，名前は bean になります。

デフォルト値

""

2.6　javax.interceptor パッケージ

javax.interceptor パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@AroundConstruct	コンストラクタの呼び出しをインターセプトするメソッドに設定します。
@AroundInvoke	ビジネスメソッドの呼び出しをインターセプトするメソッドに設定します。
@ExcludeClassInterceptors	クラスインターセプタを適用しないメソッドに設定します。
@ExcludeDefaultInterceptors	デフォルトインターセプタを適用しないクラス，およびメソッドに設定します。
@Interceptor	インターセプタクラスに設定します。
@InterceptorBinding	InterceptorBinding 型として宣言するアノテーションに設定します。
@Interceptors	適用するインターセプタクラスを設定します。クラス，およびメソッドに設定できます。

2.6.1　@AroundConstruct

(1)　説明

コンストラクタの呼び出しをインターセプトするメソッドに設定します。

(2)　属性

@AroundConstruct の属性はありません。

2.6.2　@AroundInvoke

(1)　説明

ビジネスメソッドの呼び出しをインターセプトするメソッドに設定します。

(2)　属性

@AroundInvoke の属性はありません。

2.6.3　@ExcludeClassInterceptors

(1)　説明

クラスインターセプタを適用しないメソッドに設定します。

(2)　属性

@ExcludeClassInterceptors の属性はありません。

2.6.4　@ExcludeDefaultInterceptors

(1)　説明

デフォルトインターセプタを適用しないクラス，およびメソッドに設定します。

(2)　属性

@ExcludeDefaultInterceptors の属性はありません。

2.6.5　@Interceptor

(1)　説明

インターセプタクラスに設定します。

(2)　属性

@Interceptor の属性はありません。

2.6.6　@InterceptorBinding

(1)　説明

InterceptorBinding 型として宣言するアノテーションに設定します。

(2)　属性

@InterceptorBinding の属性はありません。

2.6.7 @Interceptors

(1) 説明

適用するインターセプタクラスを設定します。クラス，およびメソッドに設定できます。

(2) 属性

@Interceptors の属性の一覧を次の表に示します。

属性名	機能
value	適用するインターセプタクラスを設定します。

各属性の詳細を次に示します。

(a) value 属性

型

Class[]

説明

適用するインターセプタクラスを設定します。

デフォルト値

なし

2.7　javax.persistence パッケージ

javax.persistence パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーションの区分	アノテーション名	概要
EntityManager と EntityManagerFactory のリファレンス関連のアノテーション	@PersistenceContext	コンテナ管理の EntityManager を定義します。
	@PersistenceContexts	@PersistenceContext を複数同時に記述する場合に使用します。
	@PersistenceProperty	コンテナ管理の EntityManager にプロパティを設定します。
	@PersistenceUnit	EntityManagerFactory への永続化ユニットを定義します。
	@PersistenceUnits	@PersistenceUnit を複数同時に記述する場合に使用します。

2.7.1　@PersistenceContext

(1)　説明

コンテナ管理の EntityManager のリファレンスを定義するアノテーションです。ルックアップをするクラスに付加します。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2)　属性

@PersistenceContext の属性の一覧を次の表に示します。

属性名	任意/必須	属性の説明
name	任意	EntityManager のルックアップ名を指定する属性です。
unitName	任意	persistence.xml ファイルで定義されている永続化ユニットの名前を指定する属性です。
type	任意	永続化コンテキストのライフサイクルの種類を指定する属性です。
properties	任意	@PersistenceProperty で指定するベンダ依存のプロパティを指定する属性です。

(a)　name 属性

型

String

説明

EntityManager のルックアップ名を指定する属性です。

DI を使用する場合は指定不要です。

デフォルト値

空の文字列

(b) unitName 属性

型

String

説明

persistence.xml ファイルで定義された永続化ユニットの名前を指定する属性です。

unitName 属性を指定した場合，JNDI 名前空間でアクセスできる EntityManagerFactory が使用する永続化ユニットを同じ名前にしてください。

デフォルト値

空の文字列

(c) type 属性

型

PersistenceContextType

説明

永続化コンテキストのライフサイクルの種類を指定する属性です。

指定できる値は，次の 2 種類です。

- TRANSACTION：トランザクションスコープの永続化コンテキスト
- EXTENDED：拡張永続化コンテキスト

デフォルト値

TRANSACTION

(d) properties 属性

型

PersistenceProperty[]

説明

@PersistenceProperty で指定する JPA プロバイダのベンダに依存するプロパティを指定する属性です。

指定できる値は，@PersistenceProperty の配列で指定できる範囲です。詳細は，「2.7.3 @PersistenceProperty」を参照してください。

properties 属性を指定した場合，認識できないプロパティは無視します。

デフォルト値

空の配列

2.7.2 @PersistenceContexts

(1) 説明

@PersistenceContext を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@PersistenceContexts の属性の一覧を次の表に示します。

属性名	任意/必須	属性の説明
value	必須	@PersistenceContext の配列を指定する属性です。

(a) value 属性

型

PersistenceContext[]

説明

@PersistenceContext の配列を指定する属性です。

指定できる値は，@PersistenceContext の配列で指定できる範囲です。詳細は，「2.7.1 @PersistenceContext」を参照してください。

デフォルト値

なし

2.7.3 @PersistenceProperty

(1) 説明

コンテナ管理の EntityManager にプロパティを設定するアノテーションです。

現状，使用できるプロパティはありません。

適用可能要素は，@PersistenceContext の properties 属性です。

(2) 属性

@PersistenceProperty の属性の一覧を次の表に示します。

属性名	任意/必須	属性の説明
name	必須	プロパティの名前を指定する属性です。
value	必須	プロパティの値を指定する属性です。

(a) name 属性

型

String

説明

プロパティの名前を指定する属性です。

デフォルト値

なし

(b) value 属性

型

String

説明

プロパティの値を指定する属性です。

指定できる値は，name 属性に指定したプロパティの仕様に依存します。

デフォルト値

なし

2.7.4 @PersistenceUnit

(1) 説明

EntityManagerFactory のリファレンスを定義するアノテーションです。ルックアップするクラスに付加します。

適用可能要素は，クラス，メソッド，およびフィールドです。

(2) 属性

@PersistenceUnit の属性の一覧を次の表に示します。

属性名	任意/必須	属性の説明
name	任意	EntityManagerFactory のルックアップ名を指定する属性です。
unitName	任意	persistence.xml ファイルで定義されている永続化ユニットの名前を指定する属性です。

(a) name 属性

型

String

説明

EntityManagerFactory のルックアップ名を指定する属性です。JNDI 名前空間に登録する EntityManagerFactory の名前を指定します。

指定できる値は，文字列です。

DI を使用する場合は指定不要です。

デフォルト値

空の文字列

(b) unitName 属性

型

String

説明

persistence.xml ファイルで定義された永続化ユニットの名前を指定する属性です。

unitName 属性を指定した場合，JNDI 名前空間でアクセスできる EntityManagerFactory が使用する永続化ユニットを同じ名前にしてください。

デフォルト値

空の文字列

2.7.5 @PersistenceUnits

(1) 説明

@PersistenceUnit を複数同時に記述する場合に指定するアノテーションです。

適用可能要素は，クラスです。

(2) 属性

@PersistenceUnits の属性の一覧を次の表に示します。

属性名	任意/必須	属性の説明
value	必須	@PersistenceUnit の配列を指定する属性です。

(a) value 属性

型

PersistenceUnit[]

説明

@PersistenceUnit の配列を指定する属性です。

指定できる値は，@PersistenceUnit の配列で指定できる範囲です。詳細は，「2.7.4 @PersistenceUnit」を参照してください。

デフォルト値

なし

2.8　javax.servlet.annotation パッケージ

javax.servlet.annotation パッケージに含まれるアノテーションの一覧を次の表に示します。

アノテーション一覧

アノテーション名	機能
@HandlesTypes	ServletContainerInitializer インタフェースの実装クラスが扱うクラスの型を設定します。
@HttpConstraint	デフォルトのセキュリティ制約を設定します。
@HttpMethodConstraint	HTTP メソッド単位のセキュリティ制約を設定します。
@MultipartConfig	サーブレットが multipart/form-data リクエストを扱うための設定をします。
@ServletSecurity	サーブレットのセキュリティ制約を設定します。
@WebInitParam	サーブレットまたはフィルタの初期パラメタを設定します。
@WebFilter	フィルタを設定します。
@WebListener	リスナを設定します。
@WebServlet	サーブレットを設定します。

2.8.1　@HandlesTypes

(1)　説明

ServletContainerInitializer インタフェースの実装クラスが扱うクラスの型を設定します。

(2)　属性

@HandlesTypes の属性の一覧を次の表に示します。

属性名	機能
value	ServletContainerInitializer インタフェースの実装クラスが扱うクラスやアノテーションの型を設定します。設定されたクラスを継承(extends)または実装(implements)しているクラス，もしくは設定されたアノテーションを付けているクラスのリストが，ServletContainerInitializer インタフェースの実装クラスに渡されます。

各属性の詳細を次に示します。

(a)　value 属性

型

Class[]

説明

ServletContainerInitializer インタフェースの実装クラスが扱うクラスやアノテーションの型を設定します。設定されたクラスを継承(extends)または実装(implements)しているクラス，もしくは設定されたアノテーションを付けているクラスのリストが，ServletContainerInitializer インタフェースの実装クラスに渡されます。

デフォルト値

{}

2.8.2 @HttpConstraint

(1) 説明

デフォルトのセキュリティ制約を設定します。

(2) 属性

@HttpConstraint の属性の一覧を次の表に示します。

属性名	機能
value	ロールを設定しない場合の振る舞いを設定します。
rolesAllowed	認証に用いるユーザ名のリストを設定します。
transportGuarantee	クライアントとサーバ間の通信方法を設定します。

各属性の詳細を次に示します。

(a) value 属性

型

ServletSecurity.EmptyRoleSemantic

説明

ロールを設定しない場合の振る舞いを設定します。

デフォルト値

javax.servlet.annotation.ServletSecurity.EmptyRoleSemantic.
PERMIT

(b) rolesAllowed 属性

型

String[]

説明

認証に用いるユーザ名のリストを設定します。

デフォルト値

{}

(c) transportGuarantee 属性

型

ServletSecurity.TransportGuarantee

説明

クライアントとサーバ間の通信方法を設定します。

デフォルト値

javax.servlet.annotation.ServletSecurity.
TransportGuarantee.
NONE

2.8.3 @HttpMethodConstraint

(1) 説明

HTTP メソッドのセキュリティ制約を設定します。

(2) 属性

@HttpMethodConstraint の属性の一覧を次の表に示します。

属性名	機能
value	セキュリティ制約を適用する HTTP メソッドを設定します。
emptyRoleSemantic	ロールを設定しない場合の振る舞いを設定します。
rolesAllowed	認証に用いるユーザ名のリストを設定します。
transportGuarantee	クライアントとサーバ間の通信方法を設定します。

各属性の詳細を次に示します。

(a) value 属性

型

String

説明

セキュリティ制約を適用する HTTP メソッドを設定します。

デフォルト値

なし

(b) emptyRoleSemantic 属性

型

ServletSecurity.EmptyRoleSemantic

説明

ロールを設定しない場合の振る舞いを設定します。

デフォルト値

javax.servlet.annotation.ServletSecurity.
EmptyRoleSemantic.
PERMIT

(c) rolesAllowed 属性

型

String[]

説明

認証に用いるユーザ名のリストを設定します。

デフォルト値

{}

(d) transportGuarantee 属性

型

ServletSecurity.TransportGuarantee

説明

クライアントとサーバ間の通信方法を設定します。

デフォルト値

javax.servlet.annotation.ServletSecurity.
TransportGuarantee.
NONE

2.8.4 @MultipartConfig

(1) 説明

サーブレットが multipart/form-data リクエストを扱うための設定をします。

@MultipartConfig の属性の一覧を次の表に示します。

属性名	機能
fileSizeThreshold	アップロードされたファイルがディスクに書き込まれるサイズのしきい値を設定します。
location	アップロードされるファイルを保存するディレクトリを設定します。
maxFileSize	アップロードされるファイルの最大サイズを設定します。
maxRequestSize	multipart/form-data リクエストの最大サイズを設定します。

各属性の詳細を次に示します。

(2) 属性

(a) fileSizeThreshold 属性

型

int

説明

アップロードされたファイルがディスクに書き込まれるサイズのしきい値を設定します。

デフォルト値

0

(b) location 属性

型

String

説明

アップロードされるファイルを保存するディレクトリを設定します。

デフォルト値

""

(c) maxFileSize 属性

型

long

説明

アップロードされるファイルの最大サイズを設定します。

デフォルト値

-1L（無制限）

(d) maxRequestSize 属性

型

long

説明

multipart/form-data リクエストの最大サイズを設定します。

デフォルト値

-1L（無制限）

2.8.5 @ServletSecurity

(1) 説明

サーブレットのセキュリティ制約を設定します。

(2) 属性

@ServletSecurity の属性の一覧を次の表に示します。

属性名	機能
httpMethodConstraints	サーブレットの HTTP メソッドごとのセキュリティ制約を設定します。
value	サーブレットのデフォルトのセキュリティ制約を設定します。

各属性の詳細を次に示します。

(a) httpMethodConstraints 属性

型

HttpMethodConstraint[]

説明

サーブレットの HTTP メソッドごとのセキュリティ制約を設定します。

デフォルト値

{}

(b) value 属性

型

HttpConstraint

説明

サーブレットのデフォルトのセキュリティ制約を設定します。

デフォルト値

@javax.servlet.annotation.HttpConstraint

2.8.6 @WebInitParam

(1) 説明

サーブレットまたはフィルタの初期パラメタを設定します。

(2) 属性

@WebInitParam の属性の一覧を次の表に示します。

属性名	機能
description	パラメタの説明を設定します。
name	パラメタ名を設定します。
value	パラメタの値を設定します。

各属性の詳細を次に示します。

(a) description 属性

型

String

説明

パラメタの説明を設定します。

デフォルト値

""

(b) name 属性

型

String

説明

パラメタ名を設定します。

デフォルト値

(c) value 属性

型

String

説明

パラメタの値を設定します。

デフォルト値

なし

2.8.7 @WebFilter

(1) 説明

フィルタを設定します。

(2) 属性

@WebFilter の属性の一覧を次の表に示します。

属性名	機能
asyncSupported	フィルタが非同期リクエスト処理をサポートするかどうかを設定します。
description	フィルタの説明を設定します。
dispatcherTypes	フィルタの適応条件を設定します。
displayName	表示名を設定します。
filterName	フィルタ名を設定します。
initParams	フィルタの初期パラメタを設定します。
largeIcon	GUI ツールで使用する大アイコンを設定します。
servletNames	マッピングを行うサーブレットのサーブレット名を設定します。
smallIcon	GUI ツールで使用する小アイコンを設定します。
urlPatterns	マッピングする URL パターンを設定します。
value	マッピングする URL パターンを設定します。urlPatterns と同時に設定した場合は無視されます。

各属性の詳細を次に示します。

(a) asyncSupported 属性

型

boolean

説明

フィルタが非同期リクエスト処理をサポートするかどうかを設定します。

デフォルト値

false

(b) description 属性

型

String

説明

フィルタの説明を設定します。

デフォルト値

""

(c) dispatcherTypes 属性

型

DispatcherType[]

説明

フィルタの適応条件を設定します。

デフォルト値

javax.servlet.DispatcherType.REQUEST

(d) displayName 属性

型

String

説明

表示名を設定します。

デフォルト値

""

(e) filterName 属性

型

String

説明

フィルタ名を設定します。

デフォルト値

""

(f) initParams 属性

型

WebInitParam[]

説明

フィルタの初期パラメタを設定します。

デフォルト値

{}

(g) largeIcon 属性

型

String

説明

GUI ツールで使用する大アイコンを設定します。

デフォルト値

""

(h) servletNames 属性

型

String[]

説明

マッピングを行うサーブレットのサーブレット名を設定します。

デフォルト値

{}

(i) smallIcon 属性

型

String

説明

GUI ツールで使用する小アイコンを設定します。

デフォルト値

""

(j) urlPatterns 属性

型

String[]

説明

マッピングする URL パターンを設定します。

デフォルト値

{}

(k) value 属性

型

String[]

説明

マッピングする URL パターンを設定します。urlPatterns と同時に設定した場合は無視されます。

デフォルト値

{}

2.8.8 @WebListener

(1) 説明

リスナを設定します。

(2) 属性

@WebListener の属性の一覧を次の表に示します。

属性名	機能
value	リスナの説明を設定します。

(a) value 属性

型

String

説明

リスナの説明を設定します。

デフォルト値

""

2.8.9 @WebServlet

(1) 説明

サーブレットを設定します。

(2) 属性

@WebServlet の属性の一覧を次の表に示します。

属性名	機能
asyncSupported	サーブレットが非同期リクエスト処理をサポートするかどうかを設定します。
description	サーブレットの説明を設定します。
displayName	表示名を設定します。
initParams	サーブレットの初期パラメタを設定します。
largeIcon	GUI ツールで使用する大アイコンを設定します。
loadOnStartup	サーブレットの開始順序を設定します。
name	サーブレット名を設定します。
smallIcon	GUI ツールで使用する小アイコンを設定します。
urlPatterns	マッピングする URL パターンを設定します。
value	マッピングする URL パターンを設定します。urlPatterns と同時に設定した場合は無視されます。

各属性の詳細を次に示します。

(a) asyncSupported 属性

型

boolean

説明

サーブレットが非同期リクエスト処理をサポートするかどうかを設定します。

デフォルト値

false

(b) description 属性

型

String

説明

サーブレットの説明を設定します。

デフォルト値

""

(c) displayName 属性

型

String

説明

表示名を設定します。

デフォルト値

""

(d) initParams 属性

型

WebInitParam[]

説明

サーブレットの初期パラメタを設定します。

デフォルト値

{}

(e) largeIcon 属性

型

String

説明

GUI ツールで使用する大アイコンを設定します。

デフォルト値

""

(f) loadOnStartup 属性

型

int

説明

サーブレットの開始順序を設定します。

デフォルト値

-1

(g) name 属性

型

String

説明

サーブレット名を設定します。

デフォルト値

""

(h) smallIcon 属性

型

String

説明

GUI ツールで使用する小アイコンを設定します。

デフォルト値

""

(i) urlPatterns 属性

型

String[]

説明

マッピングする URL パターンを設定します。

デフォルト値

{}

(j) value 属性

型

String[]

説明

マッピングする URL パターンを設定します。urlPatterns と同時に設定した場合は無視されます。

デフォルト値

{}

2.9 アプリケーションサーバが対応する Dependency Injection

Dependency Injection（DI）とは，ターゲットクラスのフィールドや set メソッドにアノテーション（@EJB，@Resource，@Inject）を設定することで，オブジェクトの参照を J2EE サーバが自動的にセットする機能です。

EJB コンテナ上で動作するクラスの中で，ターゲットクラスとなるクラスを次に示します。

- Enterprise Bean
- インターセプタ

また，Web コンテナ上で動作するクラスの中で，ターゲットクラスとなるクラスを次に示します。

- サーブレット
- フィルタ
- リスナ
- タグハンドラ

Enterprise Bean のホームインタフェース，またはビジネスインタフェースへの参照を DI する場合は，@EJB を設定します。

@Resource を設定した場合は，次の表に示すリソースのタイプを DI できます。

表 2-31 @Resource で DI できるリソースのタイプ

リソースのタイプ	DI の可否※1
java.lang.String※2	×
java.lang.Character※2	×
java.lang.Integer※2	×
java.lang.Boolean※2	×
java.lang.Double※2	×
java.lang.Byte※2	×
java.lang.Short※2	×
java.lang.Long※2	×
java.lang.Float※2	×
javax.xml.rpc.Service	×
javax.xml.ws.Service	×
javax.jws.WebService	×

リソースのタイプ	DI の可否※1
javax.sql.DataSource※3	○
javax.jms.ConnectionFactory	○
javax.jms.QueueConnectionFactory※4	○
javax.jms.TopicConnectionFactory	○
javax.mail.Session	○
java.net.URL	×
javax.resource.cci.ConnectionFactory※5	○
org.omg.CORBA_2_3.ORB	○※6
javax.jms.Queue※3，※7	○
javax.jms.Topic※7	○
javax.resource.cci.InteractionSpec	×
javax.transaction.UserTransaction	○※8
javax.ejb.EJBContext	○※9
javax.ejb.SessionContext	○※9
javax.ejb.TimerService	○※9，※10
JavaBeans リソース	○
管理対象オブジェクトの独自のインタフェース	○

（凡例）

○：使用できます。

×：使用できません。

注※1

管理対象オブジェクトへの対応づけは，Java Type に関係なく，mappedName 要素で対応づけます。リソースアダプタの表示名と管理対象オブジェクト名の区切り文字には，「!#」を使用してください。

注※2

<env-entry-value>に値を設定できないので，DI，lookup で得られる値を設定できません。

注※3

DB Connector が該当します。

注※4

TP1/Message Queue - Access，Reliable Messaging が該当します。

注※5

TP1 Connector が該当します。

注※6

ORB の shareable 属性は true が設定されているものとして動作します。なお，注入される ORB オブジェクトは，ほかのコンポーネントでも使用される共有のインスタンスです。

注※7

Connector 1.5 に準拠したリソースアダプタを使用する場合は，JMS で定義する管理対象オブジェクト（javax.jms.Destination インタフェースまたはサブインタフェース）をリソースアダプタの標準 DD（ra.xml）の<connector>-<resourceadapter>-<adminobject>-<adminobject-interface>タグに指定してください。

注※8

CMT で動作する Enterprise Bean またはインターセプタでは使用できません。

注※9

Web コンテナ上で動作するクラスでは使用できません。

注※10

Stateful SessionBean や，Stateful SessionBean に適用されたインターセプタでは使用できません。

3 Web コンテナで使用する API

この章では，Web コンテナで使用する API について説明します。ここでは，アプリケーションサーバの Web コンテナ独自の例外クラスについて説明します。

3.1 例外クラス

Web コンテナの API のうち，アプリケーションサーバが提供している例外クラスについて説明します。

Web コンテナの例外クラスを次の表に示します。

表 3-1 Web コンテナの例外クラス

例外名	内容
com.hitachi.software.web.dbsfo.DatabaseAccessException	データベースセッションフェイルオーバ機能でデータベースへのアクセスに失敗したことを通知する例外です。 この例外が出力された場合，データベースが正常に稼働しているか，データベースと J2EE サーバの通信路に問題が発生していないか，およびデータベースセッションフェイルオーバ機能が無効となっていないか確認し，次の対策をしてください。 データベースに障害が発生している場合 データベースの回復手順に従い原因を対策してください。 データベースと J2EE サーバの通信路に問題が発生している場合 通信路の問題を解決してください。通信路に問題が発生した場合，データベース上の排他が未解放となっていることがあります。業務を再開する前に無効な接続を確認して未解放となっている排他を解放してください。 データベースセッションフェイルオーバ機能が無効の場合 拡張子または URI によるデータベースセッションフェイルオーバ機能の抑止によってデータベースセッションフェイルオーバ機能が無効となったリクエスト処理内では，HttpSession オブジェクトの操作はしないでください。 また，J2EE サーバの webserver.dbsfo.exception_type_backcompat プロパティに true を指定している場合，データベースセッションフェイルオーバ機能の抑止の対象となる URL で HTTP セッションの操作をすると，この例外がスローされます。データベースや通信路に問題がない場合にこの例外が発生したときは，データベースセッションフェイルオーバ機能の抑止の対象となる URL で HTTP セッションの操作をしていないかを確認してください。 DatabaseAccessException クラスは java.lang.IllegalStateException クラスを継承しています。
HttpSessionLimitExceededException クラス	HttpSession オブジェクトが上限値を超えたことを通知する例外です。 この例外は，HttpSession オブジェクト数の上限が設定できる，J2EE サーバモードの場合に適用されます。 J2EE アプリケーションを構成するプログラム（サーブレットなど）で com.hitachi.software.web.session.HttpSessionLimitExceededException クラスを使用する場合は，<Application Server のインストールディレクトリ>¥CC¥lib¥ejbserver.jar をクラスパスに追加して，Java プログラムをコンパイルしてください。 HttpSessionLimitExceededException クラスは java.lang.IllegalStateException クラスを継承しています。

例外名	内容
com.hitachi.software.web.dbsfo.SessionOperationException	HttpSession の操作ができない状態であることを通知する例外です。 この例外がスローされる場合を次に示します。 • データベースセッションフェイルオーバ抑止機能を使用した場合，データベースセッションフェイルオーバ機能が無効となったリクエスト処理内では HttpSession オブジェクトの操作はできません。HttpSession オブジェクトを取得するために javax.servlet.http.HttpServletRequest#getSession()または getSession(boolean create)を呼び出した場合，この例外がスローされます。 • 参照専用リクエストでは，HTTP セッションの無効化はできません。参照専用リクエストで javax.servlet.http.HttpSession#invalidate()を呼び出した場合，この例外がスローされます。 • Web アプリケーション単位の同時実行スレッド数制御の実行待ちキューを使用して 503 エラーを返す設定をしている場合は，DD（web.xml）で指定するエラーページでは HTTP セッションの作成および無効化はできません。DD（web.xml）で指定したエラーページで HTTP セッションを作成したり，javax.servlet.http.HttpSession#invalidate()を呼び出したりすると，この例外がスローされます。 この例外がスローされた場合は，次の点を確認してください。 • データベースセッションフェイルオーバ抑止機能を使用している場合は，抑止する拡張子，または URI の設定に問題がないかを確認してください。設定に問題がない場合は，Web アプリケーションを確認して，データベースセッションフェイルオーバ抑止機能の対象となる URL で HTTP セッションの操作をしていないかどうか確認してください。 • 参照専用リクエスト定義機能を使用している場合は，参照専用リクエストの拡張子，または URI の設定に問題がないかを確認してください。設定に問題がない場合は，Web アプリケーションを確認して，参照専用リクエストで HTTP セッションを無効化していないか確認してください。 • 実行待ちキューを使用して 503 エラーを返す設定をしている場合は，DD（web.xml）で指定したエラーページで HTTP セッションの作成または無効化していないか確認してください。 SessionOperationException クラスは java.lang.IllegalStateException クラスを継承しています。

4 EJB クライアントアプリケーションで使用する API

この章では，EJB クライアントアプリケーションで使用する API および例外クラスについて説明します。

4.1　EJB クライアントアプリケーションで使用する API の一覧

EJB クライアントアプリケーションで使用する API には，セキュリティ機能および通信タイムアウトを設定する API があります。API の一覧を次の表に示します。

表 4-1　EJB クライアントアプリケーションで使用する API の一覧

クラス名	機能
EJBClientInitializer クラス	EJB クライアント用の J2EE サービスを初期化します。
LoginInfoManager クラス	セキュリティ機能を設定します。この API 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 セキュリティ管理機能編」の「17.1 LoginInfoManager クラス」を参照してください。
RequestTimeoutConfigFactory クラス	RMI-IIOP タイムアウトを設定するために必要な RequestTimeoutConfig オブジェクトを取得します。
RequestTimeoutConfig クラス	RMI-IIOP タイムアウトを設定します。
UserTransactionFactory クラス	EJB クライアントでトランザクションを使用するための UserTransaction オブジェクトを取得します。

4.2 EJBClientInitializer クラス

説明

EJB クライアント用の J2EE サービスを初期化します。

EJBClientInitializer クラスのパッケージ名は，
com.hitachi.software.ejb.ejbclient.EJBClientInitializer です。

メソッド一覧

メソッド名	機能
initialize メソッド	EJB クライアント用の J2EE サービスを初期化します。

initialize メソッド

説明

EJB クライアントアプリケーション用の J2EE サービスを初期化します。また，トランザクション処理中に EJB クライアントが停止した場合，EJB クライアントを再起動したあとに，グローバルトランザクションのリカバリ処理を開始します。

EJB クライアントプロセスの開始直後に，EJB クライアントのユーザコードから，initialize メソッドを呼び出してください。

なお，initialize メソッドを呼び出す前に，javax.naming.InitialContext を生成した場合，または UserTransactionFactory クラスの getUserTransaction メソッドを呼び出した場合，その時点で初期化処理が行われます。

形式

```
public static void initialize()
  throws InitializeFailedException;
```

パラメタ

なし

例外

com.hitachi.software.ejb.ejbclient.InitializeFailedException：

サービスの初期化に失敗しました。

戻り値

なし

注意事項

サービスの初期化処理で例外が発生した場合は，EJB クライアント実行時のシステムプロパティが正しく設定されていないおそれがあります。例外のメッセージに従って対処してください。

4.3 RequestTimeoutConfigFactory クラス

説明

RMI-IIOP 通信タイムアウトを設定するオブジェクトである RequestTimeoutConfig を取得するためのファクトリです。getRequestTimeoutConfig メソッドで RequestTimeoutConfig を取得したあと，RequestTimeoutConfig のメソッドでタイムアウトを設定します。

RequestTimeoutConfigFactory クラスのパッケージ名は，com.hitachi.software.ejb.ejbclient です。

メソッド一覧

メソッド名	機能
getRequestTimeoutConfig メソッド	RequestTimeoutConfig オブジェクトを取得します。

getRequestTimeoutConfig メソッド

説明

RequestTimeoutConfig オブジェクトを取得します。

形式

```
public static RequestTimeoutConfig getRequestTimeoutConfig();
```

パラメタ

なし

例外

なし

戻り値

RequestTimeoutConfig：

RequestTimeoutConfig オブジェクトを返却します。

4.4 RequestTimeoutConfig クラス

説明

RMI-IIOP 通信タイムアウトを設定するオブジェクトです。

RequestTimeoutConfig クラスのパッケージ名は，com.hitachi.software.ejb.ejbclient です。

メソッド一覧

メソッド名	機能
setRequestTimeout メソッド（形式 1）	RMI-IIOP 通信タイムアウトを設定します。 オブジェクトにタイムアウトを設定します。
setRequestTimeout メソッド（形式 2）	RMI-IIOP 通信タイムアウトを設定します。 スレッドにタイムアウトを設定します。
unsetRequestTimeout メソッド	setRequestTimeout メソッド（形式 2）で設定した RMI-IIOP 通信タイムアウトの設定をデフォルト設定に戻します。

setRequestTimeout メソッド（形式 1）

説明

RMI-IIOP 通信タイムアウトを設定します。obj パラメタのコピーを生成し，sec パラメタをタイムアウト値として設定したオブジェクトを返却します。このメソッドで設定したタイムアウトは，返却されたオブジェクトに対して有効です。

形式

```
public java.rmi.Remote setRequestTimeout(java.rmi.Remote obj,
                                         int sec)
  throws IllegalArgumentException,
         IllegalStateException;
```

パラメタ

obj：

タイムアウトを設定するオブジェクト（EJBHome または EJBObject）を指定します。

sec：

0～86400 の整数でタイムアウト時間（単位：秒）を指定します。0 を指定した場合，タイムアウトを設定しません。

例外

java.lang.IllegalArgumentException：

タイムアウト設定対象として不正なオブジェクト，またはタイムアウト時間として不正な値を指定しました。

java.lang.IllegalStateException：

タイムアウトの設定に失敗しました。

戻り値

タイムアウト設定済みのオブジェクトを返却します。

注意事項

このメソッドでタイムアウトを設定する場合，setRequestTimeout メソッド（形式 2）を使用してタイムアウトを設定する場合に比べて，処理に時間が掛かります。

setRequestTimeout メソッド（形式 2）

説明

RMI-IIOP 通信タイムアウトを設定します。実行中のスレッドに対し，パラメタ sec をタイムアウト値として設定します。このメソッドで設定したタイムアウトは，現在実行中のスレッドに対して有効です。なお，処理の終了時には，unset メソッドを使用して必ずタイムアウトの設定を解除してください。同一スレッド内でこのメソッドを複数回呼び出した場合，タイムアウトの設定値が上書きされます。

形式

```
public void setRequestTimeout(int sec)
  throws IllegalArgumentException,
         IllegalStateException;
```

パラメタ

sec：

0～86400 の整数でタイムアウト時間（単位：秒）を指定します。0 を指定した場合，タイムアウトを設定しません。

例外

java.lang.IllegalArgumentException：

タイムアウト設定対象として不正なオブジェクト，またはタイムアウト時間として不正な値を指定しました。

java.lang.IllegalStateException：

タイムアウトの設定に失敗しました。

戻り値

なし

注意事項

このメソッドでタイムアウトを設定する場合は，処理が終わった時点で必ず unsetRequestTimeout メソッドを呼び出してタイムアウトの設定を解除してください。解除しないと，ほかのクライアントからの呼び出しに対して該当スレッドが使用された場合に，そのクライアントにとって意図しない通信タイムアウトが発生するおそれがあります。

unsetRequestTimeout メソッド

説明

RMI-IIOP 通信タイムアウトの設定を解除します。実行中のスレッドに対し，setRequestTimeout（形式 2）で設定したタイムアウトを解除します。なお，setRequestTimeout（形式 2）でスレッドにタイムアウトを設定した場合は，処理の終了時に必ずこのメソッドを使用してタイムアウトの設定を解除してください。setRequestTimeout（形式 2）を呼び出さないでこのメソッドを呼び出した場合や，同一スレッド内でこのメソッドを複数回呼び出した場合でも，例外は発生しません。

形式

```
public void unsetRequestTimeout()
  throws IllegalStateException;
```

パラメタ

なし

例外

java.lang.IllegalStateException：

タイムアウトの解除に失敗しました。

戻り値

なし

4.5　UserTransactionFactory クラス

説明

EJB クライアントでトランザクションを使用するためのオブジェクトである UserTransaction オブジェクトを取得するためのファクトリです。

UserTransactionFactory クラスのパッケージ名は，com.hitachi.software.ejb.ejbclient.UserTransactionFactory です。

メソッド一覧

メソッド名	機能
getUserTransaction メソッド	UserTransaction オブジェクトを取得します。

getUserTransaction メソッド

説明

UserTransaction オブジェクトを取得します。

形式

```
public static UserTransaction getUserTransaction();
```

例外

java.lang.IllegalStateException：

EJB クライアント以外から API を発行しました。または，UserTransaction オブジェクトの取得に失敗しました。

戻り値

javax.transaction.UserTransaction オブジェクト

4.6　例外クラス

EJB クライアントアプリケーションの API で使用する例外クラスのうち，アプリケーションサーバが提供しているクラスについて説明します。

EJB クライアントアプリケーションの API で使用する例外クラスを次の表に示します。

表 4-2　EJB クライアントアプリケーションの API で使用する例外クラス

例外名	内容
com.hitachi.software.ejb.security.base.authentication.NotFoundServerException	LoginInfoManager クラスの login メソッドでログインしようとした場合に，ログイン先の J2EE サーバに接続できなかったときに送出されます。 ejbserver.serverName プロパティに指定する J2EE サーバ名が，ログイン先の J2EE サーバ名と同じになっていることを確認してください。また，ログイン先の J2EE サーバが起動していることを確認してください。
com.hitachi.software.ejb.security.base.authentication.InvalidUserNameException	LoginInfoManager クラスの login メソッドでログインしようとした場合に，ユーザ名が不正だったときに送出されます。 ユーザ名が正しいかを確認してください。
com.hitachi.software.ejb.security.base.authentication.InvalidPasswordException	LoginInfoManager クラスの login メソッドでログインしようとした場合に，パスワードが不正だったときに送出されます。 パスワードが正しいかを確認してください。

5 TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API

この章では，TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API について説明します。

5.1 TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API の一覧

TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API の一覧を次の表に示します。

表 5-1 TP1 インバウンドアダプタによって OpenTP1 と連携する場合に使用する API の一覧

インタフェース名	機能
TP1InMessage インタフェース	OpenTP1 の RPC に指定された入力メッセージを取得したり，応答用の出力メッセージを生成したりするためのインタフェースです。
TP1MessageListener インタフェース	TP1 インバウンドアダプタが OpenTP1 から RPC を受信した場合に呼び出す onMessage メソッドを提供するインタフェースです。
TP1OutMessage インタフェース	OpenTP1 からの RPC の応答時に返す出力メッセージを保持するためのインタフェースです。

5.2 TP1InMessage インタフェース

説明

OpenTP1 の RPC に指定された入力メッセージオブジェクトを取得したり，応答用の出力メッセージオブジェクトを生成したりするためのインタフェースです。

TP1InMessage インタフェースのパッケージ名は，com.hitachi.software.ejb.adapter.tp1 です。

形式

```
public interface TP1InMessage
{
  public byte[] getInputData();
  public TP1OutMessage createOutMessage();
}
```

メソッド一覧

メソッド名	機能
getInputData メソッド	OpenTP1 の RPC に指定された入力メッセージオブジェクトを取得するメソッドです。
createOutMessage メソッド	OpenTP1 の RPC の応答用の出力メッセージオブジェクトを生成するメソッドです。

getInputData メソッド

説明

OpenTP1 の RPC に指定された入力メッセージオブジェクトを取得するメソッドです。

形式

```
public byte[] getInputData();
```

パラメタ

なし

例外

なし

戻り値

OpenTP1 の RPC に指定された入力メッセージオブジェクトです。サイズは，RPC の in_len で指定された値です。

createOutMessage メソッド

説明

OpenTP1 の RPC の応答用の出力メッセージオブジェクトを生成するメソッドです。

形式

```
public TP1OutMessage createOutMessage();
```

パラメタ

なし

例外

なし

戻り値

サービスの出力メッセージオブジェクトです。OpenTP1 の RPC の応答時に返す出力メッセージを保持します。

5.3 TP1MessageListener インタフェース

説明

TP1 インバウンドアダプタが OpenTP1 から RPC を受信した場合に呼び出す onMessage メソッドを提供するインタフェースです。TP1 インバウンドアダプタから呼び出すサービスでビジネスロジックを実装する必要があります。

TP1MessageListener インタフェースのパッケージ名は，com.hitachi.software.ejb.adapter.tp1 です。

形式

```
public interface TP1MessageListener
{
  public TP1OutMessage onMessage(TP1InMessage in);
}
```

メソッド一覧

メソッド名	機能
onMessage メソッド	TP1 インバウンドアダプタが OpenTP1 から RPC を受信した場合に呼び出されるメソッドです。

onMessage メソッド

説明

TP1 インバウンドアダプタが OpenTP1 から RPC を受信した場合に呼び出すメソッドです。

形式

```
public TP1OutMessage onMessage(TP1InMessage in);
```

パラメタ

in：

サービスの入力メッセージオブジェクトを指定します。

例外

なし

戻り値

サービスの出力メッセージオブジェクトです。OpenTP1 の RPC の応答時に返す出力パラメタを保持します。

5.4　TP1OutMessage インタフェース

説明

OpenTP1 からの RPC の応答時に返す出力メッセージを保持するためのインタフェースです。

TP1OutMessage インタフェースのパッケージ名は，com.hitachi.software.ejb.adapter.tp1 です。

形式

```
public interface TP1OutMessage
{
public byte[] getOutputData(int outLen) throws IllegalArgumentException;
public int getMaxOutputLength();
}
```

メソッド一覧

メソッド名	機能
getOutputData メソッド	OpenTP1 の RPC の応答を格納する byte 配列を取得するメソッドです。
getMaxOutputLength メソッド	OpenTP1 の RPC の応答の長さを返すメソッドです。

getOutputData メソッド

説明

OpenTP1 の RPC の応答を格納する byte 配列を取得するメソッドです。取得した byte 配列に出力データを格納することで，RPC の応答データを設定できます。

getOutputData メソッドが複数回呼び出された場合，最後に呼び出された getOutputData メソッドが取得した byte 配列が，OpenTP1 への応答として使用されます。それ以前に呼び出された getOutputData メソッドが取得した byte 配列は使用されません。

形式

```
public byte[] getOutputData(int outLen) throws IllegalArgumentException;
```

パラメタ

outLen：

応答の長さ（バイト数）を，0～<getMaxOutputLength メソッドで得られる長さ>で指定します。

例外

IllegalArgumentException：

パラメタの outLen に 0～<getMaxOutputLength メソッドで得られる長さ>以外の値を指定しました。

戻り値

OpenTP1 の RPC の応答を格納する byte 配列です。サイズは，パラメタの out_Len で指定された値です。

getMaxOutputLength メソッド

説明

OpenTP1 の RPC の要求で指定された応答の長さを返します。getMaxOutputLength メソッドが返す値が，getOutputData メソッドのパラメタの outLen に指定できる最大の長さとなります。
getMaxOutputLength メソッドが返す値には，getOutputData メソッドのパラメタの outLen に指定した値は反映されません。

形式

```
public int getMaxOutputLength();
```

パラメタ

なし

例外

なし

戻り値

OpenTP1 の RPC の要求で指定された応答の長さです。

6 スレッドの非同期並行処理で使用する API

この章では，スレッドの非同期並行処理で使用する API について説明します。ここでは，Timer and Work Manager for Application Servers 仕様が定義する API と動作が異なるアプリケーションサーバの API について説明します。

6.1 Timer and Work Manager for Application Servers 仕様と動作が異なるアプリケーションサーバの API の一覧

Timer and Work Manager for Application Servers 仕様が定義する API と動作が異なるアプリケーションサーバの API の名称および動作を次の表に示します。

表 6-1 Timer and Work Manager for Application Servers 仕様と動作が異なるアプリケーションサーバの API の一覧

クラス名	メソッド名	アプリケーションサーバでの動作
commonj.timers.TimerManager クラス	schedule(TimerListener listener,Date time)メソッド	listener が javax.ejb.EnterpriseBean を継承している場合，IllegalArgumentException を返します。
	schedule(TimerListener listener,long delay)メソッド	
	schedule(TimerListener listener,Date firstTime,long period)メソッド	
	schedule(TimerListener listener,long delay,long period)メソッド	
	scheduleAtFixedRate(TimerListener listener,Date firstTime,long period)メソッド	
	scheduleAtFixedRate (TimerListener listener,long delay,long period)メソッド	
commonj.work.WorkManager クラス	schedule(Work work)メソッド	work が null の場合，WorkException をスローします。
	schedule(Work work,WorkListener wl)メソッド	work が null の場合，WorkException を返します。 WorkListener が javax.ejb.EnterpriseBean を継承している場合，IllegalArgumentException を返します。

7 ユーザログ機能で使用する API

この章では，ユーザログ機能で使用する API について説明します。

7.1　ユーザログ機能で使用する API の一覧

J2EE アプリケーション，バッチアプリケーション，または EJB クライアントアプリケーションが出力するログ（ユーザログ）をトレース共通ライブラリ形式で出力する場合に使用する API の一覧を次に示します。

表 7-1　ユーザログ機能で使用する API の一覧

クラス名	機能
CJLogRecord クラス	LogRecord クラスに，MsgID や AppName パラメタを追加したクラスです。このクラスのメソッドを利用して作成した LogRecord オブジェクト（CJLogRecord オブジェクト）を Logger.log メソッドに渡すことで，MsgID や AppName のフィールド値も実行時に指定した値で出力できます。

7.2 CJLogRecord クラス

説明

java.util.logging.LogRecord クラスに MsgID や AppName パラメタを追加したクラスです。MsgID や AppName が指定された場合の LogRecord オブジェクト（以降，CJLogRecord オブジェクトと呼びます）を作成するためのスタティックメソッドを提供しています。

CJLogRecord クラスのパッケージ名は，com.hitachi.software.ejb.application.userlog です。

メソッド一覧

メソッド名	機能
create メソッド（形式 1）	Level，Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 2）	Level，Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 3）	Level，Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 4）	Level，Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 5）	Level，Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 6）	Level，Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 7）	Level，Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 8）	Level，Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 9）	Level，Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
create メソッド（形式 10）	Level，Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 1）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 2）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 3）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。

メソッド名	機能
createp メソッド（形式 4）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 5）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 6）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 7）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 8）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 9）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown ，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createp メソッド（形式 10）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 1）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 2）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 3）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 4）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 5）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 6）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

メソッド名	機能
createrb メソッド（形式 7）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 8）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 9）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。
createrb メソッド（形式 10）	Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

なお，CJLogRecord クラスの継承元である LogRecord クラスおよび各メソッドのパラメタに指定する Level は，java.util.logging パッケージに属するクラスです。

create メソッド（形式 1）

説明

Level，Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 2）

説明

Level，Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 String appName,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 3）

説明

Level，Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object param1,
                                 String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：
　LogRecord にセットするオブジェクトを指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 4）

説明

Level，Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object param1,
                                 String appName,
                                 String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：
　LogRecord にセットするオブジェクトを指定します。

appName：
　AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 5）

説明

Level，Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 6）

説明

Level，Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 String appName,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 7）

説明

Level，Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object[] params,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 8）

説明

Level，Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Object[] params,
                                 String appName,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 9）

説明

Level，Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 Object[] params,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

create メソッド（形式 10）

説明

Level，Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord create(Level level,
                                 String msg,
                                 Throwable thrown,
                                 Object[] params,
                                 String appName,
                                 String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 1）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 2）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  String appName,
                                  String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：
　ロギングの要求を発行したメソッド名を指定します。

msg：
　文字列メッセージ，またはメッセージカタログのキーを指定します。

appName：
　AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：
　MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 3）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object param1,
                                  String msgID);
```

パラメタ

level：
　メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：
　ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：

LogRecord にセットするオブジェクトを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 4）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object param1,
                                  String appName,
                                  String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：

LogRecord にセットするオブジェクトを指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 5）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 6）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  String appName,
                                  String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 7）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object[] params,
                                  String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 8）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Object[] params,
                                  String appName,
                                  String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 9）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  Object[] params,
                                  String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createp メソッド（形式 10）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createp(Level level,
                                  String sourceClass,
                                  String sourceMethod,
                                  String msg,
                                  Throwable thrown,
                                  Object[] params,
                                  String appName,
                                  String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 1）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 2）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
                                   String sourceMethod,
                                   String bundleName,
                                   String msg,
                                   String appName,
                                   String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 3）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                   String sourceClass,
```

```
                                    String sourceMethod,
                                    String bundleName,
                                    String msg,
                                    Object param1,
                                    String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：

LogRecord にセットするオブジェクトを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 4）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
                                    String bundleName,
                                    String msg,
```

```
                                    Object param1,
                                    String appName,
                                    String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

param1：

LogRecord にセットするオブジェクトを指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 5）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
                                    String bundleName,
```

```
                                    String msg,
                                    Throwable thrown,
                                    String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 6）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
                                    String bundleName,
                                    String msg,
                                    Throwable thrown,
```

```
                                    String appName,
                                    String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 7）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
                                    String bundleName,
                                    String msg,
```

```
                                    Object[] params,
                                    String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 8）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
                                    String bundleName,
                                    String msg,
                                    Object[] params,
```

```
                                    String appName,
                                    String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 9）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，Object 配列および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
```

```
                                String bundleName,
                                String msg,
                                Throwable thrown,
                                Object[] params,
                                String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

createrb メソッド（形式 10）

説明

Level，発生元クラス名（sourceClass），発生元メソッド名（sourceMethod），リソースバンドル名（bundleName），Message，Thrown，Object 配列，AppName および MsgID を渡して，CJLogRecord オブジェクトを作成します。

形式

```
public static CJLogRecord createrb(Level level,
                                    String sourceClass,
                                    String sourceMethod,
                                    String bundleName,
                                    String msg,
                                    Throwable thrown,
                                    Object[] params,
                                    String appName,
                                    String msgID);
```

パラメタ

level：

メッセージレベル識別子（例えば，SEVERE など）を指定します。

sourceClass：

ロギングの要求を発行したクラス名を指定します。

sourceMethod：

ロギングの要求を発行したメソッド名を指定します。

bundleName：

msg を地域化するためのリソースバンドル名を指定します。

msg：

文字列メッセージ，またはメッセージカタログのキーを指定します。

thrown：

LogRecord にセットする例外オブジェクトを指定します。

params：

ユーザが利用する（Logger.log メソッドの Object 配列に直接渡す予定だった）ユーザ固有の Object 配列を指定します。

appName：

AppName フィールドに出力する値（アプリケーション識別名）を指定します。

msgID：

MsgID フィールドに出力する値（メッセージ文字列）を指定します。

戻り値

CJLogRecord オブジェクトを返却します。

8 監査ログ出力で使用する API

この章では，J2EE アプリケーションまたはバッチアプリケーションで監査ログを出力する場合に使用する API について説明します。

8.1　監査ログ出力で使用する API の一覧

監査ログ出力で使用する API の一覧を次の表に示します。

表 8-1　監査ログ出力で使用する API の一覧

クラス名	機能
AuditLogRecord クラス	監査ログのレコードを表すクラスです。
UserAuditLogger クラス	J2EE アプリケーションまたはバッチアプリケーションから監査ログを出力するためのロガークラスです。
例外クラス	監査ログの API で発生する例外を表す例外クラスです。

監査ログ出力で使用する API を使用する場合，次の JAR ファイルをクラスパスに指定してコンパイルする必要があります。

Windows の場合

```
%COSMINEXUS_HOME%¥common¥lib¥auditlog.jar
```

UNIX の場合

```
/opt/Cosminexus/common/lib/auditlog.jar
```

8.2 AuditLogRecord クラス

説明

監査ログのレコードを表すクラスです。

J2EE アプリケーションまたはバッチアプリケーションから監査ログとして出力する情報は，このクラスの set で始まるメソッドを使用して設定します。

監査ログに出力する情報のうち，監査事象の種別，監査事象の結果，および動作情報については，指定できる値が決まっています。このクラスでは，これらの情報を指定するための定数をフィールドとして定義しています。

なお，設定する値の前後に空白を指定した場合，空白は削除されます。また，空文字（""）や空白だけを設定した場合，値は設定されていないものとみなされます。

AuditLogRecord クラスのパッケージ名は，com.hitachi.software.auditlog です。

形式

```
public class AuditLogRecord
  extends Object
  implements Serializable
```

メソッド一覧

メソッド名	機能
getAfterInfo メソッド	変更後情報を取得します。
getAuthority メソッド	権限情報を取得します。
getBeforeInfo メソッド	変更前情報を取得します。
getCategory メソッド	監査事象の種別を取得します。
getDetectionPoint メソッド	検出場所を取得します。
getHaid メソッド	冗長化識別情報を取得します。
getLocation メソッド	ロケーション情報を取得します。
getMessage メソッド	自由記述を取得します。
getMessageId メソッド	メッセージ ID を取得します。
getObjectInfo メソッド	オブジェクト情報を取得します。
getObjectLocation メソッド	オブジェクトロケーション情報を取得します。
getOperation メソッド	動作情報を取得します。
getOutputPoint メソッド	出力元の場所を取得します。
getReceiverHost メソッド	リクエスト送信先ホストを取得します。
getReceiverPort メソッド	リクエスト送信先ポート番号を取得します。
getResult メソッド	監査事象の結果を取得します。

メソッド名	機能
getSenderHost メソッド	リクエスト送信元ホストを取得します。
getSenderPort メソッド	リクエスト送信元ポート番号を取得します。
getServiceInstance メソッド	サービスインスタンス名を取得します。
getSubjectId メソッド	サブジェクト識別情報を取得します。
getSubjectPoint メソッド	指示元の場所を取得します。
setAfterInfo メソッド	変更後情報を設定します。
setAuthority メソッド	権限情報を設定します。
setBeforeInfo メソッド	変更前情報を設定します。
setCategory メソッド	監査事象の種別を設定します。
setDetectionPoint メソッド	検出場所を設定します。
setHaid メソッド	冗長化識別情報を設定します。
setLocation メソッド	ロケーション情報を設定します。
setMessage メソッド	自由記述を設定します。
setMessageId メソッド	メッセージ ID を設定します。
setObjectInfo メソッド	オブジェクト情報を設定します。
setObjectLocation メソッド	オブジェクトロケーション情報を設定します。
setOperation メソッド	動作情報を設定します。
setOutputPoint メソッド	出力元の場所を設定します。
setReceiverHost メソッド	リクエスト送信先ホストを設定します。
setReceiverPort メソッド	リクエスト送信先ポート番号を設定します。
setResult メソッド	監査事象の結果を設定します。
setSenderHost メソッド	リクエスト送信元ホストを設定します。
setSenderPort メソッド	リクエスト送信元ポート番号を設定します。
setServiceInstance メソッド	サービスインスタンス名を設定します。
setSubjectId メソッド	サブジェクト識別情報を設定します。
setSubjectPoint メソッド	指示元の場所を設定します。

監査ログのレコードに設定する値の推奨値

監査ログのレコードに設定する値には，項目ごとに推奨されている長さおよび文字種があります。監査ログの各項目の推奨値を次の表に示します。

表 8-2　監査ログのレコードに設定する値の推奨値

項目	推奨値	
	長さ（バイト）	文字種
変更後情報	規定はありません。	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
権限情報	0～128	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
変更前情報	規定はありません。	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
監査事象の種別	種別によって異なります。	「A～Z」（%x41-5A）および「a～z」（/ %x61-7A）のアルファベット。 なお，アプリケーションサーバでは，監査事象の種別の推奨値をフィールドとして定義しています。「表 8-3　監査事象の種別の推奨値を表すフィールドの一覧」を参照してください。
検出場所	0～255	次の文字。 •「0～9」の数値（%x30-39） •「A～Z」（%x41-5A）および「a～z」（%x61-7A）のアルファベット •「-」（%x2D） •「.」(%x2E)
冗長化識別情報	1～2	「0～9」の数値（%x30-39）。
ロケーション情報	0～32	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
自由記述	規定はありません。	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
メッセージ ID	9～32	次の文字。 •「0～9」の数値（%x30-39） •「A～Z」のアルファベット（%x41-5A） •「-」（%x2D）
オブジェクト情報	0～256	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
オブジェクトロケーション情報	規定はありません。	次の文字以外の US-ASCII。

項目	推奨値	
	長さ（バイト）	文字種
		• CR（%x0D） • LF（%x0A）
動作情報	0～32	任意。 なお，アプリケーションサーバでは，動作情報の推奨値をフィールドとして定義しています。「表 8-4　動作情報の推奨値を表すフィールドの一覧」を参照してください。
出力元の場所	0～255	次の文字。 •「0～9」の数値（%x30-39） •「A～Z」（%x41-5A）および「a～z」（%x61-7A）のアルファベット •「-」（%x2D） •「.」(%x2E)
リクエスト送信先ホスト	0～255	次の文字。 •「0～9」の数値（%x30-39） •「A～Z」（%x41-5A）および「a～z」（%x61-7A）のアルファベット •「-」（%x2D） •「.」(%x2E)
リクエスト送信先ポート番号	1～5	「0～9」の数値（%x30-39）。
監査事象の結果	結果によって異なります。	「A～Z」（%x41-5A）および「a～z」（%x61-7A）のアルファベット。 なお，アプリケーションサーバでは，監査事象の結果の推奨値をフィールドとして定義しています。「表 8-6　監査事象の結果の推奨値を表すフィールドの一覧」を参照してください。
リクエスト送信元ホスト	0～255	次の文字。 •「0～9」の数値（%x30-39） •「A～Z」（%x41-5A）および「a～z」（%x61-7A）のアルファベット •「-」（%x2D） •「.」(%x2E)
リクエスト送信元ポート番号	1～5	「0～9」の数値（%x30-39）。
サービスインスタンス名	0～128	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）
サブジェクト識別情報	0～256	次の文字以外の US-ASCII。 • CR（%x0D） • LF（%x0A）

項目	推奨値	
	長さ（バイト）	文字種
指示元の場所	0～255	次の文字。 • 「0～9」の数値（%x30-39） • 「A～Z」（%x41-5A）および「a～z」（%x61-7A）のアルファベット • 「-」（%x2D） • 「.」(%x2E)

監査事象の種別の推奨値を表すフィールド

監査事象の種別の推奨値を表すフィールドの一覧を，次の表に示します。監査事象の種別は，setCategory メソッドで設定します。

表 8-3　監査事象の種別の推奨値を表すフィールドの一覧

フィールド名	実際の値	意味
public static final String CATEGORY_ACCESS_CONTROL	"AccessControl"	管理者または一般利用者が，管理リソースまたはセキュリティリソースへのアクセスを試みて，成功または失敗したことを示す事象です。
public static final String CATEGORY_ANOMALY_EVENT	"AnomalyEvent"	しきい値オーバーなどの異常が発生したことを示す事象です。
public static final String CATEGORY_AUTHENTICATION	"Authentication"	管理者または一般利用者が，認証を試みて，成功または失敗したことを示す事象です。
public static final String CATEGORY_CONFIGURATION_ACCESS	"ConfigurationAccess"	管理者が許可された運用操作を実行して，操作が正常終了または失敗したことを示す事象です。
public static final String CATEGORY_CONTENT_ACCESS	"ContentAccess"	重要なデータへのアクセスを試みて，成功または失敗したことを示す事象です。
public static final String CATEGORY_EXTERNAL_SERVICE	"ExternalService"	外部サービスとの通信結果を示す事象です。
public static final String CATEGORY_FAILURE	"Failure"	ソフトウェアの異常を示す事象です。
public static final String CATEGORY_LINK_STATUS	"LinkStatus"	機器間のリンク状態を示す事象です。
public static final String CATEGORY_MAINTENANCE	"Maintenance"	保守作業を実行して，操作が正常終了または失敗したことを示す事象です。
public static final String CATEGORY_MANAGEMENT_ACTION	"ManagementAction"	プログラムの重要なアクションが実行されたことを示す事象です。

フィールド名	実際の値	意味
		ほかの監査事象を契機として実行するアクションを示します。
public static final String CATEGORY_START_STOP	"StartStop"	ソフトウェアの起動と終了を示す事象です。

動作情報の推奨値を表すフィールド

動作情報の推奨値を表すフィールドの一覧を，次の表に示します。動作情報は，setOperation メソッドで設定します。

表 8-4　動作情報の推奨値を表すフィールドの一覧

フィールド名	実際の値	意味
public static final String OPERATION_ADD	"Add"	動作情報の意味は監査事象の種別との組み合わせで決まります。監査事象の種別と組み合わせた動作情報の意味については，「表 8-5　監査事象の種別と動作情報の組み合わせ」を参照してください。
public static final String OPERATION_BACKUP	"Backup"	
public static final String OPERATION_DELETE	"Delete"	
public static final String OPERATION_DOWN	"Down"	
public static final String OPERATION_ENFORCE	"Enforce"	
public static final String OPERATION_INSTALL	"Install"	
public static final String OPERATION_INVOKE	"Invoke"	
public static final String OPERATION_LOGIN	"Login"	
public static final String OPERATION_LOGOFF	"Logoff"	
public static final String OPERATION_LOGON	"Logon"	
public static final String OPERATION_LOGOUT	"Logout"	
public static final String OPERATION_MAINTAIN	"Maintain"	
public static final String OPERATION_NOTIFY	"Notify"	
public static final String OPERATION_OCCUR	"Occur"	

フィールド名	実際の値	意味
public static final String OPERATION_RECEIVE	"Receive"	
public static final String OPERATION_REFER	"Refer"	
public static final String OPERATION_REQUEST	"Request"	
public static final String OPERATION_RESPONSE	"Response"	
public static final String OPERATION_SEND	"Send"	
public static final String OPERATION_START	"Start"	
public static final String OPERATION_STOP	"Stop"	
public static final String OPERATION_UNINSTALL	"Uninstall"	
public static final String OPERATION_UP	"Up"	
public static final String OPERATION_UPDATE	"Update"	

動作情報の意味は監査事象の種別との組み合わせで決まります。監査事象の種別と，動作情報の組み合わせについて，次の表に示します。

表 8-5　監査事象の種別と動作情報の組み合わせ

監査事象の種別	動作情報	意味
StartStop	Start	開始または起動を表します。
	Stop	終了または停止を表します。
Authentication	Login	ログインを表します。
	Logout	ログアウトを表します。
	Logon	ログオンを表します。
	Logoff	ログオフを表します。
AccessControl	Enforce	実施を表します。
ConfigurationAccess	Refer	設定情報の参照を表します。
	Add	設定情報の追加を表します。
	Update	設定情報の更新を表します。

監査事象の種別	動作情報	意味
	Delete	設定情報の削除を表します。
Failure	Occur	発生を表します。
LinkStatus	Up	リンク活性を表します。
	Down	リンク非活性を表します。
ExternalService	Request	要求を表します。
	Response	応答を表します。
	Send	発信を表します。
	Receive	受信を表します。
ContentAccess	Refer	参照を表します。
	Add	追加を表します。
	Update	更新またはアップデートを表します。
	Delete	削除を表します。
Maintenance	Install	インストールを表します。
	Uninstall	アンインストールを表します。
	Update	更新またはアップデートを表します。
	Backup	バックアップを表します。
	Maintain	保守作業を表します。
AnomalyEvent	Occur	発生を表します。
ManagementAction	Invoke	管理者などの呼び出しを表します。
	Notify	管理者などへの通知を表します。

監査事象の結果の推奨値を表すフィールド

監査事象の結果の推奨値を表すフィールドの一覧を，次の表に示します。監査事象の結果は，setResult メソッドで設定します。

表 8-6　監査事象の結果の推奨値を表すフィールドの一覧

フィールド名	実際の値	意味
public static final String RESULT_FAILURE	"Failure"	事象の失敗を表します。
public static final String RESULT_OCCURRENCE	"Occurrence"	成功，失敗の分類がない事象の発生を表します。
public static final String RESULT_SUCCESS	"Success"	事象の成功を表します。

getAfterInfo メソッド

説明

変更後情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getAfterInfo();
```

パラメタ

なし

例外

なし

戻り値

変更後情報が返されます。

getAuthority メソッド

説明

権限情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getAuthority();
```

パラメタ

なし

例外

なし

戻り値

権限情報が返されます。

getBeforeInfo メソッド

説明

変更前情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getBeforeInfo();
```

パラメタ

なし

例外

なし

戻り値

変更前情報が返されます。

getCategory メソッド

説明

監査事象の種別を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public String getCategory();
```

パラメタ

なし

例外

なし

戻り値

監査事象の種別が返されます。

getDetectionPoint メソッド

説明

検出場所を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getDetectionPoint();
```

パラメタ

なし

例外

なし

戻り値

検出場所が返されます。

getHaid メソッド

説明

冗長化識別情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getHaid();
```

パラメタ

なし

例外

なし

戻り値

冗長化識別情報が返されます。

getLocation メソッド

説明

ロケーション情報を取得します。

この項目に値を設定していない場合，アプリケーションサーバによって自動的に付加された値（性能解析トレースのルートアプリケーション情報）が出力されます。

形式

```
public String getLocation();
```

パラメタ

なし

例外

なし

戻り値

ロケーション情報が返されます。

getMessage メソッド

説明

自由記述を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getMessage();
```

パラメタ

なし

例外

なし

戻り値

自由記述が返されます。

getMessageId メソッド

説明

メッセージ ID を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public String getMessageId();
```

パラメタ

なし

例外

なし

戻り値

メッセージ ID が返されます。

getObjectInfo メソッド

説明

オブジェクト情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getObjectInfo();
```

パラメタ

なし

例外

なし

戻り値

オブジェクト情報が返されます。

getObjectLocation メソッド

説明

オブジェクトロケーション情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getObjectLocation();
```

パラメタ

なし

例外

なし

戻り値

オブジェクトロケーション情報が返されます。

getOperation メソッド

説明

動作情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getOperation();
```

パラメタ

なし

例外

なし

戻り値

動作情報が返されます。

getOutputPoint メソッド

説明

出力元の場所を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getOutputPoint();
```

パラメタ

なし

例外

なし

戻り値

出力元の場所が返されます。

getReceiverHost メソッド

説明

リクエスト送信先ホストを取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getReceiverHost();
```

パラメタ

なし

例外

なし

戻り値

リクエスト送信先ホストが返されます。

getReceiverPort メソッド

説明

リクエスト送信先ポート番号を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public int getReceiverPort();
```

パラメタ

なし

例外

なし

戻り値

リクエスト送信先ポート番号が返されます。

getResult メソッド

説明

監査事象の結果を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public String getResult();
```

パラメタ

なし

例外

なし

戻り値

監査事象の結果が返されます。

getSenderHost メソッド

説明

リクエスト送信元ホストを取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getSenderHost();
```

パラメタ

なし

例外

なし

戻り値

リクエスト送信元ホストが返されます。

getSenderPort メソッド

説明

リクエスト送信元ポート番号を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public int getSenderPort();
```

パラメタ

なし

例外

なし

戻り値

リクエスト送信元ポート番号が返されます。

getServiceInstance メソッド

説明

サービスインスタンス名を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getServiceInstance();
```

パラメタ

なし

例外

なし

戻り値

サービスインスタンス名が返されます。

getSubjectId メソッド

説明

サブジェクト識別情報を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にアプリケーションサーバによって自動的に付加された値（OS のアカウント）が出力されます。なお，OS のアカウントとして出力される情報は，OS ごとに異なります。

Windows の場合
　ユーザ名が出力されます。

UNIX の場合
　実効ユーザ ID が出力されます。

形式

```
public String getSubjectId();
```

パラメタ

なし

例外

なし

戻り値

サブジェクト識別情報が返されます。

getSubjectPoint メソッド

説明

指示元の場所を取得します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public String getSubjectPoint();
```

パラメタ

なし

例外

なし

戻り値

指示元の場所が返されます。

setAfterInfo メソッド

説明

変更後情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setAfterInfo(String info);
```

パラメタ

info：
　変更後情報を指定します。

例外

なし

戻り値

なし

setAuthority メソッド

説明

権限情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setAuthority(String authority);
```

パラメタ

authority：
　権限情報を指定します。

例外

なし

戻り値

なし

setBeforeInfo メソッド

説明

変更前情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setBeforeInfo(String info);
```

パラメタ

info：
　変更前情報を指定します。

例外

なし

戻り値

なし

setCategory メソッド

説明

監査事象の種別を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public void setCategory(String category);
```

パラメタ

category：
　監査事象の種別を指定します。

例外

なし

戻り値

なし

setDetectionPoint メソッド

説明

検出場所を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setDetectionPoint(String detectionPoint);
```

パラメタ

detectionPoint：
　検出場所を指定します。

例外

なし

戻り値

なし

setHaid メソッド

説明

冗長化識別情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setHaid(String haid);
```

パラメタ

haid：

冗長化識別情報を指定します。

例外

なし

戻り値

なし

setLocation メソッド

説明

ロケーション情報を設定します。

この項目に値を設定していない場合，アプリケーションサーバによって，性能解析トレースのルートアプリケーション情報が設定されます。

形式

```
public void setLocation(String location);
```

パラメタ

location：

ロケーション情報を指定します。

例外

なし

戻り値

なし

setMessage メソッド

説明

自由記述を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setMessage(String message);
```

パラメタ

message：
　自由記述を指定します。

例外

なし

戻り値

なし

setMessageId メソッド

説明

メッセージ ID を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public void setMessageId(String messageId);
```

パラメタ

messageId：
　メッセージ ID をで指定します。

例外

なし

戻り値

なし

setObjectInfo メソッド

説明

オブジェクト情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setObjectInfo(String objectInfo);
```

パラメタ

objectInfo：
　オブジェクト情報を指定します。

例外

なし

戻り値

なし

setObjectLocation メソッド

説明

オブジェクトロケーション情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setObjectLocation(String objectLocation);
```

パラメタ

objectLocation：
　オブジェクトロケーション情報を指定します。

例外

なし

戻り値

なし

setOperation メソッド

説明

動作情報を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setOperation(String operation);
```

パラメタ

operation：
　動作情報を指定します。

例外

なし

戻り値

なし

setOutputPoint メソッド

説明

出力元の場所を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setOutputPoint(String outputPoint);
```

パラメタ

outputPoint：
　出力元の場所を指定します。

例外

なし

戻り値

なし

setReceiverHost メソッド

説明

リクエスト送信先ホストを設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setReceiverHost(String receiverHost);
```

パラメタ

receiverHost：
　リクエスト送信先ホストを指定します。

例外

なし

戻り値

なし

setReceiverPort メソッド

説明

リクエスト送信先ポート番号を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setReceiverPort(int receiverPort);
```

パラメタ

receiverPort：
　リクエスト送信先ポート番号を指定します。

例外

なし

戻り値

なし

setResult メソッド

説明

監査事象の結果を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時に AuditLogException がスローされます。

形式

```
public void setResult(String result);
```

パラメタ

result：
　監査事象の結果を指定します。

例外

なし

戻り値

なし

setSenderHost メソッド

説明

リクエスト送信元ホストを設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setSenderHost(String senderHost);
```

パラメタ

senderHost：
　リクエスト送信元ホストを指定します。

例外

なし

戻り値

なし

setSenderPort メソッド

説明

リクエスト送信元ポート番号を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setSenderPort(int senderPort);
```

パラメタ

senderPort：

リクエスト送信元ポート番号を指定します。

例外

なし

戻り値

なし

setServiceInstance メソッド

説明

サービスインスタンス名を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setServiceInstance(String serviceInstance);
```

パラメタ

serviceInstance：

サービスインスタンス名を指定します。

例外

なし

戻り値

なし

setSubjectId メソッド

説明

サブジェクト識別情報を設定します。指定できるのは，アカウント識別子（ユーザ ID）だけです。

この項目に値を設定していない場合，または null を設定した場合は，OS のアカウントが設定されます。

OS のアカウントとして設定される情報は，OS ごとに異なります。

Windows の場合
　ユーザ名が設定されます。

UNIX の場合
　実効ユーザ ID が設定されます。

なお，パラメタに，OS のアカウントを指定してはいけません。

形式

```
public void setSubjectId(String subjectId);
```

パラメタ

subjectId：
　サブジェクト識別情報（ユーザ ID）を指定します。

例外

なし

戻り値

なし

setSubjectPoint メソッド

説明

指示元の場所を設定します。

この項目に値を設定していない場合，UserAuditLogger.log メソッド実行時にこの項目は出力されません。この場合，タグも出力されません。

形式

```
public void setSubjectPoint(String subjectPoint);
```

パラメタ

subjectPoint：

　指示元の場所を指定します。

例外

なし

戻り値

なし

8.3 UserAuditLogger クラス

説明

J2EE アプリケーションまたはバッチアプリケーションから監査ログを出力するためのロガークラスです。

UserAuditLogger クラスのパッケージ名は，com.hitachi.software.auditlog です。

形式

```
public class UserAuditLogger
```

メソッド一覧

メソッド名	機能
getLogger メソッド	パラメタに指定した発生コンポーネント名ごとに，UserAuditLogger オブジェクトを取得します。
isEnabled メソッド	監査ログが有効か無効かを判定します。
isLoggable メソッド	パラメタに指定したメッセージ ID から，監査ログの出力要否を判定します。
log メソッド	パラメタに指定した AuditLogRecord オブジェクトを基に，監査ログを出力します。

getLogger メソッド

説明

パラメタに指定した発生コンポーネント名ごとに，UserAuditLogger オブジェクトを取得します。例えば，getLogger(new String("Component"))メソッドを 2 回実行した場合は，2 回とも同じ UserAuditLogger オブジェクトを取得できます。

発生コンポーネント名と UserAuditLogger オブジェクトの対応は，UserAuditLogger クラスの内部で管理されています。内部で管理されている発生コンポーネント名と，パラメタに指定した発生コンポーネント名の判定は，String.equals(component)メソッドで実行されます。内部で管理されている発生コンポーネント名とパラメタに指定した発生コンポーネント名が一致した場合に，対応する UserAuditLogger オブジェクトが返されます。

形式

```
public static UserAuditLogger getLogger(String component);
```

パラメタ

component：

発生コンポーネント名を指定します。

例外

com.hitachi.software.auditlog.AuditLogException：
　初期化に失敗しました。

戻り値

UserAuditLogger：
　監査ログが有効な場合に，UserAuditLogger オブジェクトが返されます。
　パラメタに指定した発生コンポーネント名ごとに，異なるオブジェクトが返されます。

null：
　監査ログが無効な場合に返されます。

isEnabled メソッド

説明

監査ログが有効か無効かを判定します。

なお，監査ログ定義ファイルの auditlog.enabled キーに false が指定されている場合，監査ログは無効になります。

形式

```
public static boolean isEnabled();
```

パラメタ

なし

例外

なし

戻り値

true：
　監査ログが有効な場合に返されます。

false：
　監査ログが無効な場合に返されます。

isLoggable メソッド

説明

パラメタに指定したメッセージ ID から，監査ログの出力要否を判定します。

出力要否は，監査ログ定義ファイルの auditlog.filtered.message.list キーに，指定したメッセージ ID が含まれているかどうかで判定されます。

auditlog.filtered.message.list キーに，指定したメッセージ ID が含まれている場合は，戻り値として false が返され，監査ログを出力できません。auditlog.filtered.message.list キーに，指定したメッセージ ID が含まれていない場合は，戻り値として true が返され，監査ログを出力できます。

なお，このメソッドでは，監査ログが有効か無効かについては判定しません。

形式

```
public boolean isLoggable(String messageId);
```

パラメタ

messageId：

メッセージ ID を指定します。必ず一つのメッセージ ID を指定してください。

例えば，「String messageId = "message-1,message-2";」のように複数のメッセージ ID を指定した場合，一つながりの文字列として認識されます。「String messageId = "message-1";」のように，メッセージ ID を一つだけ指定してください。

例外

なし

戻り値

true：

監査ログを出力できる場合に返されます。パラメタに「null」または空文字を指定した場合も true が返されます。

false：

監査ログを出力できない場合に返されます。

log メソッド

説明

パラメタに指定した AuditLogRecord オブジェクトを基に，監査ログを出力します。出力される監査ログは，AuditLogRecord クラスの set で始まるメソッドで設定した値です。

監査ログのレコードは，出力項目ごとに推奨値が決められています。出力項目ごとの推奨値については，「8.2　AuditLogRecord クラス」を参照してください。ただし，推奨されている文字種以外の文字を設定したり，推奨されている文字数を超過して設定したりした場合も，設定した値がそのまま出力されます。

監査ログの出力先は，監査ログ定義ファイルで指定します。監査ログ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「11.2.1　監査ログ定義ファイル」を参照してください。また，監査ログの出力形式については，マニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「6.4.3　監査ログの出力形式」を参照してください。

すでに監査ログを出力したファイルが存在する場合，出力した監査ログには，そのファイルのアクセス権限が引き継がれます。監査ログを出力したファイルが存在しない場合は，監査ログを出力したファイルに対して，次のアクセス権限が設定されます。

表 8-7　監査ログを出力したファイルのアクセス権限

OS	所有者	設定されるアクセス権限
Windows の場合	出力先のフォルダの設定が引き継がれます。	
UNIX の場合	監査ログを出力したプロセスのユーザおよびプライマリグループ	666※

注※

umask によるマスクが行われます。umask=0022 が設定されている場合，実際には 644 となります。

形式

```
public void log(AuditLogRecord)
  throws AuditLogException;
```

パラメタ

AuditLogRecord：

AuditLogRecord オブジェクトを指定します。

例外

AuditLogException：

監査ログの出力が失敗しました。次の要因が考えられます。

- 監査ログを出力するプロセスの実行ユーザに，監査ログの書き込み権限がない。

- 監査ログを出力する際にディスクフルになった。
- 指定が必要な出力項目が指定されていない。

戻り値

なし

8.4 例外クラス

監査ログ出力で使用する API で発生する例外を表す，例外クラスについて説明します。

例外名

com.hitachi.software.auditlog.AuditLogException

内容

監査ログに関する例外を表すクラスです。

コンストラクタの型は Exception クラスと同じです。また，メソッドについても Exception クラスのすべてのメソッドを継承しています。

コンストラクタ

- AuditLogException()
- AuditLogException(String message)
- AuditLogException(String message, Throwable cause)
- AuditLogException(Throwable cause)

これらのコンストラクタでは，Exception クラスのコンストラクタが呼び出されます。

メソッド

AuditLogException クラスおよび Exception クラスで定義されたメソッドはありません。

Exception クラスが Throwable クラスから継承したメソッドだけが使用できます。

9 性能解析トレースで使用する API

この章では，性能解析トレースのルートアプリケーション情報取得機能で使用する API について説明します。

9.1　性能解析トレースで使用する API の一覧

性能解析トレースで使用する API の一覧を次の表に示します。

表 9-1　性能解析トレースで使用する API の一覧

クラス名	機能
CprfTrace クラス	性能解析トレース関連の機能を提供します。
PrfTrace クラス	性能解析トレース関連の機能を提供します。

9.2　CprfTrace クラス

説明

性能解析トレース関連の機能を提供します。

CprfTrace クラスのパッケージ名は，com.hitachi.software.ejb.application.prf です。

メソッド一覧

メソッド名	機能
getRootApInfo メソッド	現在のスレッドが保持している性能解析トレースのルートアプリケーション情報を文字列表現で返します。

注意事項

このクラスを使用する場合は，次の JAR ファイルをクラスパスに指定してコンパイルします。

- Windows の場合
  <アプリケーションサーバのインストールディレクトリ>¥CC¥lib¥ejbserver.jar
- UNIX の場合
  /opt/Cosminexus/CC/lib/ejbserver.jar

getRootApInfo メソッド

説明

現在のスレッドが保持している性能解析トレースのルートアプリケーション情報を文字列表現で返します。

ルートアプリケーション情報の文字列表現とは，ルートアプリケーション情報を構成する IP アドレス，プロセス ID，および通信番号をスラッシュ(/)で区切ったものです（最大長 45)。

```
例：”10.209.15.130/1234/0x0000000000000001”
```

ルートアプリケーション情報の文字列表現をログファイルなどに記録することによって，任意のタイミングで性能解析トレースファイルとの突き合わせが可能となるため，トラブルシュートに役立てることができます。

性能解析トレースのルートアプリケーション情報取得については，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「7.4　性能解析トレースのルートアプリケーション情報取得のための実装」を参照してください。ルートアプリケーション情報を利用したログ調査については，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「7.7.7　ルートアプリケーション情報を利用したログ調査」を参照してください。

形式

```
public static final String getRootApInfo();
```

パラメタ

なし

例外

なし

戻り値

ルートアプリケーション情報の文字列表現。

次の場合には，null を返します。

- Performance Tracer がインストールされていないなど，現在のスレッドが保持しているルートアプリケーション情報が存在しない場合
- EJB コンテナ外および Web コンテナ外から呼び出された場合
- EJB クライアントから呼び出された場合

9.3 PrfTrace クラス

説明

性能解析トレース関連の機能を提供します。

PrfTrace クラスのパッケージ名は，com.hitachi.software.javaee.util.prf.PrfTrace です。

メソッド一覧

メソッド名	機能
getClientApInfo メソッド	getPrfTrace メソッド実行時に取得した性能解析トレースのクライアントアプリケーション情報を文字列表現で返します。
getPrfTrace メソッド	現在のスレッドが保持しているトレース情報を取得し，取得した情報を保持する PrfTrace のインスタンスを返します。
getRootApInfo メソッド	getPrfTrace メソッド実行時に取得した性能解析トレースのルートアプリケーション情報を文字列表現で返します。

注意事項

アプリケーションで PrfTrace クラスを使用する場合は同時に CprfTrace クラスを使用できません。

getClientApInfo メソッド

説明

getPrfTrace メソッド実行時に取得した性能解析トレースのクライアントアプリケーション情報を文字列表現で返します。

形式

```
public final String getClientApInfo()
```

パラメタ

なし

例外

なし

戻り値

クライアントアプリケーション情報の文字列表現。

クライアントアプリケーション情報を引き継ぐシーケンスの範囲外で実行した場合 null を返します。

getPrfTrace メソッド

説明

現在のスレッドが保持しているトレース情報を取得し，取得した情報を保持する PrfTrace のインスタンスを返します。

形式

```
public static PrfTrace getPrfTrace()
```

パラメタ

なし

例外

なし

戻り値

PrfTrace のインスタンス

getRootApInfo メソッド

説明

getPrfTrace メソッド実行時に取得した性能解析トレースのルートアプリケーション情報を文字列表現で返します。

形式

```
public final String getRootApInfo()
```

パラメタ

なし

例外

なし

戻り値

ルートアプリケーション情報の文字列表現。

ルートアプリケーション情報を引き継ぐシーケンスの範囲外で実行した場合 null を返します。

10 JavaVM で使用する API

この章では，製品の JavaVM（以降，JavaVM と呼びます）で使用する API について説明します。

なお，JavaVM は，Java SE 8 または Java SE 9 に準拠しています。詳細については，マニュアル「アプリケーションサーバ & BPM/ESB 基盤 概説」を参照してください。また，JDK 8 または JDK 9 で使用できる API については，Oracle 社が提供している JDK 8 または JDK 9 のドキュメントを参照してください。

10.1　JavaVM で使用する API の一覧

JavaVM で使用する API の一覧を，次の表に示します。

表 10–1　JavaVM で使用する API の一覧

クラス名	機能
BasicExplicitMemory クラス	Explicit メモリブロックを表すクラスです。なお，このクラスは，ほかの JDK 製品では利用できません。
ExplicitMemory クラス	Explicit メモリブロックを表す抽象クラスです。 派生クラスである BasicExplicitMemory クラスで処理する内容を定義します。なお，このクラスは，ほかの JDK 製品では利用できません。
MemoryArea クラス	Explicit メモリブロックまたは Java ヒープを表す抽象クラスです。なお，このクラスは，ほかの JDK 製品では利用できません。
MemoryInfo クラス	GC のメモリ情報を取得します。なお，このクラスは，ほかの JDK 製品では利用できません。
例外クラス	JavaVM で使用する API で発生する例外を表す例外クラスです。なお，このクラスは，ほかの JDK 製品では利用できません。

パッケージ名の省略

なお，この章では java.lang および MemoryArea パッケージに所属するクラスのクラス名は，完全名ではなくクラス名だけを記載します。

例

java.lang.Object の場合：Object

10.2　BasicExplicitMemory クラス

説明

アプリケーションサーバが生成した Explicit メモリブロックを表すクラスです。ExplicitMemory クラスを継承しています。

BasicExplicitMemory クラスのパッケージは，JP.co.Hitachi.soft.jvm.MemoryArea です。

コンストラクタ・メソッド一覧

コンストラクタ・メソッド名	機能
BasicExplicitMemory コンストラクタ（形式 1）	Explicit メモリブロックを初期化します。
BasicExplicitMemory コンストラクタ（形式 2）	Explicit メモリブロックを初期化して名称を設定します。
getName メソッド	Explicit メモリブロック名称を返却します。

BasicExplicitMemory コンストラクタ（形式 1）

説明

Explicit メモリブロックを初期化します。初期化することで，オブジェクトを Explicit ヒープに配置できます。

Explicit メモリブロックを初期化して，インスタンスの名称を「BasicExplicitMemory-<Explicit メモリブロックの ID>」にします。ただし，Explicit メモリブロックのメモリ領域は確保しません。

次の条件に当てはまる場合は，無効な Explicit メモリブロックにします。

- オプション HitachiUseExplicitMemory が OFF の場合（-XX:+HitachiUseExplicitMemory を指定していない場合）
- Explicit メモリブロックの最大数を超えた場合

形式

```
public BasicExplicitMemory();
```

パラメタ

なし

例外

なし

戻り値

なし

BasicExplicitMemory コンストラクタ（形式 2）

説明

Explicit メモリブロックを初期化します。また，初期化と同時にこの Explicit メモリブロックの名称を設定します。

インスタンスの名称はパラメタ name になります。ただし，パラメタ name が null の場合は，インスタンスの名称は「BasicExplicitMemory-<Explicit メモリブロックの ID>」になります。

次の条件に当てはまる場合は無効な Explicit メモリブロックにします。

- オプション HitachiUseExplicitMemory が OFF の場合（-XX:+HitachiUseExplicitMemory を指定していない場合）
- Explicit メモリブロックの最大数を超えた場合

形式

```
public BasicExplicitMemory(String name)
```

パラメタ

name：

名称を表す文字列（String）を指定します。

例外

なし

戻り値

なし

getName メソッド

説明

このオブジェクトが表す Explicit メモリブロックの名称を返却します。

形式

```
public String getName();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが表す Explicit メモリブロックの名称をこのオブジェクトの名称を表すインスタンスフィールドから取得し，その参照を返却します。

注意事項

デフォルトで設定されている名前に付いている Explicit メモリブロックの ID の一意性は保証されています。ただし，その ID を持つインスタンスが破棄されたとき，同じ Explicit メモリブロックの ID が再利用される場合があります。この場合，同時に存在することがない異なるインスタンスが同じデフォルト名を持つことがあります。

10.3　ExplicitMemory クラス

説明

Explicit メモリブロックを表す抽象クラスです。BasicExplicitMemory での処理を定義します。

ExplicitMemory クラスのパッケージは，JP.co.Hitachi.soft.jvm.MemoryArea です。

メソッド一覧

メソッド名	機能
countExplicitMemories メソッド	Explicit メモリブロック数を返却します。
freeMemory メソッド	Explicit メモリブロックの利用できるサイズを返却します。
getMemoryUsage メソッド	Explicit ヒープの利用状況を返却します。
isActive メソッド	Explicit メモリブロックが処理できるかどうかを返却します。
isReclaimed メソッド	Explicit メモリブロックが解放予約状態または解放済み状態かどうかを返却します。
newArray メソッド（形式 1）	Explicit メモリブロックに配列オブジェクトを生成します。
newArray メソッド（形式 2）	
newInstance メソッド（形式 1）	Explicit メモリブロックにオブジェクトを生成します。
newInstance メソッド（形式 2）	
newInstance メソッド（形式 3）	
reclaim メソッド（形式 1）	Explicit メモリブロックを解放予約します。
reclaim メソッド（形式 2）	
reclaim メソッド（形式 3）	
reclaim メソッド（形式 4）	
setName メソッド	Explicit メモリブロック名称を name に設定します。
toString メソッド	Explicit メモリブロック名称を返却します。
totalMemory メソッド	Explicit メモリブロックの確保済みサイズを返却します。
usedMemory メソッド	Explicit メモリブロックの使用されているメモリのサイズを返却します。

countExplicitMemories メソッド

説明

Explicit ヒープにある Explicit メモリブロックの個数を返却します。Explicit メモリブロックの状態が解放済み，または無効の場合はカウントしません。

形式

```
public static int countExplicitMemories();
```

パラメタ

なし

例外

なし

戻り値

Explicit ヒープにある Explicit メモリブロックの個数をカウントして，int 型で返却します。

注意事項

- 解放予約済み状態の Explicit メモリブロックをカウントするため，Explicit メモリブロックを明示的に操作しなくても，時間経過によって個数が変化する場合があります。
- ExplicitMemory インスタンスの個数を数えるのではなく，Explicit メモリブロックの実体数を数えるメソッドです。

freeMemory メソッド

説明

Explicit メモリブロックが利用できるメモリサイズを返却します。

形式

```
public long freeMemory();
```

パラメタ

なし

例外

InaccessibleMemoryAreaException：

サポートされていない機能です。

戻り値

共通エラーチェックをして，次のどちらかの値を返却します。共通エラーチェックについては，「10.6 Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

0：

API の処理ができない場合に返却します。

このオブジェクトが表す Explicit メモリブロックが利用できるメモリサイズ（バイト数）：

API の処理ができる場合は，このオブジェクトが表す Explicit メモリブロックが利用できるメモリサイズを long 型で返却します。

getMemoryUsage メソッド

説明

Explicit ヒープの利用状況を返却します。

形式

```
public static java.lang.management.MemoryUsage getMemoryUsage();
```

パラメタ

なし

例外

なし

戻り値

Explicit ヒープの利用状況をフィールドとして保持している java.lang.management.MemoryUsage インスタンスへの参照を，次に示す値で返却します。

init：

Explicit ヒープの初期値です。常に 0 となります。

used：

Explicit ヒープの使用されているメモリサイズ（バイト数）です。

committed：

Explicit ヒープの確保済みサイズ（バイト数）です。

max：

-XX:HitachiExplicitHeapMaxSize で指定した最大 Explicit ヒープサイズの値（バイト数）です。ただし，オプション HitachiUseExplicitMemory が OFF の場合（-XX:+HitachiUseExplicitMemory を指定した場合）は 0 を返却します。

注意事項

MemoryUsage インスタンスが持つ値は getMemoryUsage()を呼び出した時点での値です。
MemoryUsage インスタンスから各フィールドを読み出した時点では，実際の値と異なる場合があります。

isActive メソッド

説明

このオブジェクトが表す Explicit メモリブロックが処理できる状態であるかどうかを返却します。

形式

```
public boolean isActive();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが表す Explicit メモリブロックの状態を次に示す boolean 型で返却します。

true：

処理できる状態です。有効な Explicit メモリブロックで，サブ状態が Enable の場合に返却します。

false：

処理できない状態です。次のどちらかの状態の場合にこの値を返却します。

- 無効になっている Explicit メモリブロックの場合
- 有効な Explicit メモリブロックで，サブ状態が Disable の場合

注意事項

ExplicitMemory は，一度無効状態になった場合，再度有効状態になることはありません。

isReclaimed メソッド

説明

このオブジェクトが表す Explicit メモリブロックが解放予約状態または解放済み状態であるかどうかを返却します。

形式

```
public boolean isReclaimed();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトが表す Explicit メモリブロックの状態を次に示す boolean 型で返却します。

true：
　このオブジェクトが表す Explicit メモリブロックが解放予約状態または解放済み状態の場合に返却します。

false：
　このオブジェクトが表す Explicit メモリブロックが有効な状態の場合に返却します。

newArray メソッド（形式 1）

説明

パラメタ type の表すクラスのパラメタ length 長の配列インスタンスをこのオブジェクトが表す Explicit メモリブロックに直接生成します。

形式

```
public Object newArray(Class type, int length);
```

パラメタ

type：
　直接生成する配列インスタンスのクラスを指定します。

length：
　直接生成する配列インスタンスの長さを指定します。

例外

NullPointerException：
　パラメタ type が null です。

NegativeArraySizeException：
　パラメタ length が 0 未満です。

IllegalArgumentException：
　パラメタ length が 0 以上で，パラメタ type が 255 次元以上の配列クラスまたは Void.TYPE です。

InaccessibleMemoryAreaException：
　サポートされていない機能です。

戻り値

type 型の配列であるこのオブジェクトが示す Explicit メモリブロックに直接生成し，その参照を返却します。配列の長さは length 長です。

共通エラーチェックによって，処理できないと判定された場合は，java.lang.reflect.Array.newInstance(Class<?> componentType,int length)をパラメタ type，パラメタ length で呼び出し，その結果を返却します。共通エラーチェックについては，「10.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

newArray メソッド（形式 2）

説明

dimensions.length 次元の配列インスタンスをこのオブジェクトが表す Explicit メモリブロックに直接生成します。なお，パラメタ type の表すクラスの n 次元目の要素数は dimensions[n-1]です。

形式

```
public Object newArray(Class type, int[] dimensions);
```

パラメタ

type：
　直接生成する配列インスタンスのクラスを指定します。

dimensions：
　直接生成する配列インスタンスの次元数および要素数を指定します。

例外

NullPointerException：

パラメタ dimensions またはパラメタ type のうち，どちらかのパラメタの値または両方のパラメタの値が null です。

NegativeArraySizeException：

パラメタ dimensions が負の値を要素に持っています。

IllegalArgumentException：

次のどれかに当てはまる場合にスローされます。

- パラメタ dimensions の dimensions.length が 0 以下または 255 より大きい値の場合
- パラメタ type の次元数とパラメタ dimensions の dimensions.length との合計が 255 より大きい値の場合
- パラメタ type が Void.TYPE である場合

InaccessibleMemoryAreaException：

サポートされていない機能です。

戻り値

n 次元目の要素数が dimensions[n-1]である，dimensions.length 次元の type 型の配列オブジェクトをこのオブジェクトが表す Explicit メモリブロックに直接生成し，参照を返却します。

共通エラーチェックによって，処理できないと判定された場合は，java.lang.reflect.Array.newInstance(Class<?> componentType,int[] dimensions)をパラメタ type，パラメタ dimensions で呼び出し，その結果を返却します。共通エラーチェックについては，「10.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

newInstance メソッド（形式 1）

説明

パラメタ type の表すクラスのインスタンスをこのオブジェクトが表す Explicit メモリブロックに直接生成します。パラメタで指定したクラスのインスタンスだけを Explicit メモリブロックに生成します。パラメタで指定したクラスのインスタンスのコンストラクタなどによる初期化で生成されるオブジェクトについては，Java ヒープに生成します。type をこのオブジェクトとした場合の Class.newInstance()と類似していますが，一部異なります。

形式

```
public Object newInstance(Class type);
```

パラメタ

type：
　直接生成する配列インスタンスのクラスです。

例外

NullPointerException：
　パラメタ type，またはパラメタ type が表すクラスが null です。

SecurityException：
　SecurityManager があり，かつ次のうちどれかに当てはまる場合にスローされます。

- s.checkMemberAccess(type, Member.PUBLIC)の呼び出しがこのコンストラクタへのアクセスを許可しない。
- 呼び出し側のクラスローダが異なる。
- 現在のクラスローダの上位クラスローダと s.checkPackageAccess()の呼び出しがこのクラスのパッケージへのアクセスを許可しない。

NoSuchMethodException：
　パラメタ type またはパラメタ type が表すクラスに，public のパラメタなしコンストラクタがありません。

ExceptionInInitializerError：
　パラメタ type またはパラメタ type が表すクラスの初期化に失敗しました。

InstantiationException：
　パラメタ type またはパラメタ type が表すクラスが抽象クラスまたはインタフェースです。

InvocationTargetException：
　パラメタ type またはパラメタ type が表すクラスのコンストラクタの実行で例外が発生しました。

IllegalAccessException：
　クラスまたはその nullary コンストラクタにアクセスできません。

InaccessibleMemoryAreaException：
　サポートされていない機能です。

戻り値

このオブジェクトが表す Explicit メモリブロックに生成されたインスタンスへの参照を返却します。

共通エラーチェックで処理できないと判定された場合は，パラメタ type をレシーバとして Class.newInstance()のメソッドを呼び出し，その結果を返却します。共通エラーチェックについては，「10.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

注意事項

パラメタ type には，public クラスを与えることを推奨します。

newInstance メソッド（形式 2）

説明

パラメタ type の表すクラスのインスタンスを Explicit メモリブロックに直接生成します。パラメタ args に指定した値がインスタンスを生成するコンストラクタの引数として渡されます。パラメタで指定したクラスのインスタンスのコンストラクタなどによる初期化で生成されるオブジェクトについては，Java ヒープに生成します。

形式

```
public Object newInstance(Class type, Object... args);
```

パラメタ

type：

直接生成する配列インスタンスのクラスです。

args：

コンストラクタに渡すパラメタです。

例外

NullPointerException：

パラメタ type またはパラメタ args の値のどちらかまたは両方が null です。

SecurityException：

SecurityManager があり，かつ次のうちどれかに当てはまる場合にスローされます。

- s.checkMemberAccess(type,Member.PUBLIC)の呼び出しがこのコンストラクタへのアクセスを許可しない場合
- 呼び出し側のクラスローダが異なる場合
- 現在のクラスローダの上位クラスローダと s.checkPackageAccess()の呼び出しがこのクラスのパッケージへのアクセスを許可しない場合

NoSuchMethodException：

パラメタ args の要素と同じ型のパラメタを持つ public コンストラクタがパラメタ type の表すクラスにありません。

ExceptionInInitializerError：

パラメタ type またはパラメタ type が表すクラスの初期化に失敗しました。

InstantiationException：

パラメタ type またはパラメタ type が表すクラスが抽象クラスまたはインタフェースです。

InvocationTargetException：

パラメタ type またはパラメタ type が表すクラスのコンストラクタの実行で例外が発生しました。

InaccessibleMemoryAreaException：

サポートされていない機能です。

IllegalAccessException：

Constructor オブジェクトが Java 言語アクセス制御を実施するため，基本となるコンストラクタにアクセスできません。

IllegalArgumentException：

次のどれかに当てはまる場合にスローされます。

- 実パラメタ数と仮パラメタ数が異なる場合
- プリミティブ引数のラップ解除変換が失敗した場合
- プリミティブ引数のラップ解除後，パラメタ値を対応する仮パラメタ型に変換できない場合
- コンストラクタが列挙型に関連している場合

戻り値

このオブジェクトが表す Explicit メモリブロックに生成されたインスタンスへの参照を返却します。

共通エラーチェックをして，処理できないと判定した場合，type.getConstructor(arg_types※)として，java.lang.reflect.Constructor インスタンスを取得します。この場合，java.lang.reflect.Constructor をレシーバ，パラメタ args をパラメタとして，java.lang.reflect.Constructor.newInstance(Object... initargs)のメソッドを呼び出し，その結果を返却します。共通エラーチェックについては，「10.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

注※

arg_types は，パラメタ args の各要素をこのオブジェクトとして Object.getClass()を呼び出した結果を要素とする Class 配列です。

注意事項

パラメタ type には，public クラスを与えることを推奨します。

プリミティブ型を引数とするコンストラクタを呼び出すことはできません。プリミティブ型を引数とするコンストラクタを呼び出す場合は，newInstance メソッド（形式 3）を使用します。次に newInstance メソッド（形式 3）を使用したコード例を示します。

```
import JP.co.Hitachi.soft.jvm.MemoryArea.*;
import java.lang.reflect.*;
public class test1 {
  public static void main(String[] args) throws Exception {
    ExplicitMemory em = new BasicExplicitMemory();
    TheClass obj = null;

    Constructor cons = TheClass.class.getConstructor(new Class[]{int.class});
    obj = (TheClass)em.newInstance(cons,1);            // 実行成功

    obj = (TheClass)em.newInstance(TheClass.class,1);  // NoSuchMethodExceptionをスロー
  }
}

public class TheClass {
  public TheClass(int i){}
}
```

newInstance メソッド（形式 3）

説明

パラメタ cons が表すコンストラクタをパラメタ args で実行し，このオブジェクトが表す Explicit メモリブロックに直接生成します。パラメタで指定したクラスのインスタンスだけを Explicit メモリブロックに生成します。パラメタで指定したクラスのインスタンスのコンストラクタなどによる初期化で生成されるオブジェクトについては，Java ヒープに生成します。

形式

```
public Object newInstance(java.lang.reflect.Constructor cons, Object... args);
```

パラメタ

cons：

直接生成する配列インスタンスのコンストラクタを指定します。

args：

コンストラクタに渡すパラメタを指定します。

例外

NullPointerException：

パラメタ cons またはパラメタ args の値のどちらかまたは両方が null です。

ExceptionInInitializerError：

パラメタ cons が表すコンストラクタでクラスの初期化に失敗しました。

InstantiationException：

パラメタ cons が表すコンストラクタが抽象クラスです。

IllegalArgumentException：

パラメタ cons が示すコンストラクタのパラメタとパラメタ args が一致しません。

InvocationTargetException：

パラメタ cons またはパラメタ args が表すコンストラクタの実行で例外が発生しました。

InaccessibleMemoryAreaException：

サポートされていない機能です。

戻り値

このオブジェクトが表す Explicit メモリブロックに生成されたインスタンスへの参照を返却します。

共通エラーチェックで処理できないと判定された場合，
java.lang.reflect.Constructor.newInstance(Object... initargs)のパラメタ cons をこのオブジェクト，パラメタ args をパラメタとして呼び出し，その結果を返却します。共通エラーチェックについては，「10.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

注意事項

パラメタ cons には，public クラスのコンストラクタを与えることを推奨します。

reclaim メソッド（形式 1）

説明

パラメタ areas のすべての要素に対して解放予約をします。

パラメタ areas が null 以外の場合に，パラメタ areas のすべての要素に対して，要素をパラメタとして ExplicitMemory.reclaim(ExplicitMemory area)を呼び出した場合と同じ処理をします。処理する要素の順序は未定義とします。ある要素に対する処理で例外が発生した場合は，その例外をスローします。例外のスローまでに未処理だった要素に対する処理は実行しません。

形式

```
public static void reclaim(ExplicitMemory... areas);
```

パラメタ

areas：

要素に解放予約をする Explicit メモリブロックを持つ配列を指定します。

例外

NullPointerException：
　パラメタ areas が null です。

InaccessibleMemoryAreaException：
　サポートされていない機能です。

戻り値

なし

注意事項

解放予約をするだけで，解放処理はしません。

オプション HitachiExplicitMemoryAutoReclaim が ON の場合（-XX:+HitachiExplicitMemoryAutoReclaim を指定している場合），自動解放されたあとの Explicit メモリブロックは，明示管理ヒープ自動配置設定ファイルで生成された Explicit メモリブロックと同じ動作をします。この動作をさせたくない場合は，オプション HitachiExplicitMemoryAutoReclaim を OFF（-XX:+HitachiExplicitMemoryAutoReclaim を指定しない）にしてください。

reclaim メソッド（形式 2）

説明

パラメタ area が null 以外の値の場合に共通エラーチェックで処理できると判定されたとき，パラメタ area に排他処理を実施してから，パラメタ area の表す Explicit メモリブロックを解放予約します。

次の条件に当てはまる場合は，処理しません。

- パラメタ area が null の場合
- 共通エラーチェックで処理できないと判定された場合

形式

```
public static void reclaim(ExplicitMemory area);
```

パラメタ

area：
　解放予約をする Explicit メモリブロックを指定します。

例外

InaccessibleMemoryAreaException：
　サポートされていない機能です。

戻り値

なし

注意事項

解放予約をするだけで，解放処理はしません。

オプション HitachiExplicitMemoryAutoReclaim が ON の場合（-XX:+HitachiExplicitMemoryAutoReclaim を指定している場合），自動解放されたあとの Explicit メモリブロックは，明示管理ヒープ自動配置設定ファイルで生成された Explicit メモリブロックと同じ動作をします。この動作をさせたくない場合は，オプション HitachiExplicitMemoryAutoReclaim を OFF（-XX:+HitachiExplicitMemoryAutoReclaim を指定しない）にしてください。

reclaim メソッド（形式 3）

説明

パラメタ area0，area1 の表す Explicit メモリブロックを解放予約します。

パラメタ area0，パラメタ area1 それぞれをパラメタとして ExplicitMemory.reclaim(ExplicitMemory area)を呼び出した場合と同じ処理をします。パラメタ area0，パラメタ area1 の処理順序は未定義とします。一方のパラメタに対する処理で，例外が発生した場合は，その例外をスローします。もう一方のパラメタが未処理である場合，処理はしません。

形式

```
public static void reclaim(ExplicitMemory area0, ExplicitMemory area1);
```

パラメタ

area0：
　解放予約をする Explicit メモリブロックその 1 を指定します。

area1：
　解放予約をする Explicit メモリブロックその 2 を指定します。

例外

InaccessibleMemoryAreaException：
　サポートされていない機能です。

戻り値

なし

注意事項

解放予約をするだけで，解放処理はしません。

オプション HitachiExplicitMemoryAutoReclaim が ON の場合（-XX:+HitachiExplicitMemoryAutoReclaim を指定している場合），自動解放されたあとの Explicit メモリブロックは，明示管理ヒープ自動配置設定ファイルで生成された Explicit メモリブロックと同じ動作をします。この動作をさせたくない場合は，オプション HitachiExplicitMemoryAutoReclaim を OFF（-XX:+HitachiExplicitMemoryAutoReclaim を指定しない）にしてください。

reclaim メソッド（形式 4）

説明

パラメタ areas のすべての要素に対して解放予約をします。

パラメタ areas が null 以外の場合は，パラメタ areas のすべての要素に対して，要素をパラメタとして ExplicitMemory.reclaim(ExplicitMemory area)を呼び出した場合と同じ処理をします。処理する要素の順序は未定義とします。ある要素に対する処理で例外が発生した場合は，その例外をスローします。例外のスローまでに未処理だった要素に対する処理はしません。

形式

```
public static void reclaim(Iterable<ExplicitMemory> areas);
```

パラメタ

areas：
　解放予約をする Explicit メモリブロックのイテレータを指定します。

例外

NullPointerException：
　パラメタ areas の値が null です。

InaccessibleMemoryAreaException：

サポートされていない機能です。

戻り値

なし

注意事項

解放予約をするだけで，解放処理はしません。

オプション HitachiExplicitMemoryAutoReclaim が ON の場合（-XX:+HitachiExplicitMemoryAutoReclaim を指定している場合），自動解放されたあとの Explicit メモリブロックは，明示管理ヒープ自動配置設定ファイルで生成された Explicit メモリブロックと同じ動作をします。この動作をさせたくない場合は，オプション HitachiExplicitMemoryAutoReclaim を OFF（-XX:+HitachiExplicitMemoryAutoReclaim を指定しない）にしてください。

setName メソッド

説明

Explicit メモリブロックに名称を設定します。このオブジェクトが表す Explicit メモリブロックの名称として，このオブジェクトの名称を表すインスタンスフィールドにパラメタ name を設定します。

この名称は主にデバッグ用に設定します。設定した値はイベントログまたはスレッドダンプで表示します。

形式

```
public void setName(String name);
```

パラメタ

name：

設定する名称を表す String を指定します。

例外

NullPointerException：

パラメタ name の値が null です。

戻り値

なし

注意事項

複数の ExplicitMemory に同じ名前を設定できるため，名称に一意性はありません。

toString メソッド

説明

このオブジェクトの文字列表現を返却します。this.getName()を呼び出し，その結果を返却します。

形式

```
public String toString();
```

パラメタ

なし

例外

なし

戻り値

このオブジェクトの文字列表現を表す String 型オブジェクトへの参照を返却します。

totalMemory メソッド

説明

Explicit メモリブロックの確保済み総サイズを返却します。

形式

```
public long totalMemory();
```

パラメタ

なし

例外

InaccessibleMemoryAreaException：

サポートされていない機能です。

戻り値

共通エラーチェックをして，次のどちらかの値を返却します。

0：
　API の処理ができない場合に返却します。

このオブジェクトが表す Explicit メモリブロックの確保済みの総メモリサイズ（バイト数）：
　API の処理ができる場合は，このオブジェクトが表す Explicit メモリブロックが利用できるメモリサイズを long 型で返却します。

共通エラーチェックについては，「10.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

usedMemory メソッド

説明

Explicit メモリブロックの使用されているメモリのサイズを返却します。

形式

```
public long usedMemory();
```

パラメタ

なし

例外

InaccessibleMemoryAreaException：
　サポートされていない機能です。

戻り値

このオブジェクトが表す Explicit メモリブロックの使用されているメモリサイズ（バイト数）を long 型で返却します。

共通エラーチェックをして処理できないと判定された場合は，0 を返却します。共通エラーチェックについては，「10.6　Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）」を参照してください。

10.4 MemoryArea クラス

説明

Explicit メモリブロックまたは Java ヒープを表す抽象クラスです。MemoryArea クラスのパッケージは，JP.co.Hitachi.soft.jvm.MemoryArea です。

MemoryArea クラスに含まれるメソッドは，抽象メソッドのため処理がありません。各メソッドの処理については，派生クラスの同一シグネチャメソッドを参照してください。各メソッドの形式と参照先を次の表に示します。

メソッドの形式・同一シグネチャメソッドの一覧

メソッド名	形式	同一シグネチャメソッド
freeMemory メソッド	public abstract long freeMemory();	freeMemory メソッド
getName メソッド	public abstract String getName();	getName メソッド
newArray メソッド（形式 1）	public abstract Object newArray(Class type, int number);	newArray メソッド（形式 1）
newArray メソッド（形式 2）	public abstract Object newArray(Class type, int[] dimensions);	newArray メソッド（形式 2）
newInstance メソッド（形式 1）	public abstract Object newInstance(Class type);	newInstance メソッド（形式 1）
newInstance メソッド（形式 2）	public abstract Object newInstance(Class type, Object... args);	newInstance メソッド（形式 2）
newInstance メソッド（形式 3）	public abstract Object newInstance(java.lang.reflect.Constructor cons, Object... args);	newInstance メソッド（形式 3）
setName メソッド	public abstract void setName(String name);	setName メソッド
toString メソッド	public abstract String toString();	toString メソッド
totalMemory メソッド	public abstract long totalMemory();	totalMemory メソッド
usedMemory メソッド	public abstract long usedMemory();	usedMemory メソッド

10.5 MemoryInfo クラス

説明

Java プログラムから直接 GC のメモリ情報を取得できます。

例えば，現在使用中のサイズは次の式で求められます。

```
getXXXTotalMemory()-getXXXFreeMemory()
```

MemoryInfo クラスのパッケージは，JP.co.Hitachi.soft.jvm です。

メソッド一覧

メソッド名	機能
getEdenFreeMemory メソッド	Eden 領域の空きサイズを取得します。
getEdenMaxMemory メソッド	Eden 領域の最大使用サイズを取得します。
getEdenTotalMemory メソッド	Eden 領域の使用可能サイズを取得します。
getMetaspaceFreeMemory メソッド	Metaspace 領域の空きサイズを取得します。
getMetaspaceMaxMemory メソッド	Metaspace 領域の最大使用サイズを取得します。
getMetaspaceTotalMemory メソッド	Metaspace 領域の使用可能サイズを取得します。
getSurvivorFreeMemory メソッド	Survivor 領域の空きサイズを取得します。
getSurvivorMaxMemory メソッド	Survivor 領域の最大使用サイズを取得します。
getSurvivorTotalMemory メソッド	Survivor 領域の使用可能サイズを取得します。
getTenuredFreeMemory メソッド	Tenured 領域の空きサイズを取得します。
getTenuredMaxMemory メソッド	Tenured 領域の最大使用サイズを取得します。
getTenuredTotalMemory メソッド	Tenured 領域の使用可能サイズを取得します。

使用例

メモリ情報を取得する際のメソッドの使用例を次に示します。

Metaspace 領域の空きサイズを求める場合

```
free_memory = JP.co.Hitachi.soft.jvm.MemoryInfo.getMetaspaceFreeMemory()
```

現在使用中の Eden 領域を求める場合

```
use_memory = JP.co.Hitachi.soft.jvm.MemoryInfo.getEdenTotalMemory()-JP.co.Hitachi.soft
.jvm.MemoryInfo.getEdenFreeMemory()
```

注意事項

Cosminexus Developer's Kit for Java 09-70-08 以降では，JP.co.Hitachi.soft.jvm.MemoryInfo の API の返却値が java.lang.Long.MAX_VALUE を超える場合，java.lang.Long.MAX_VALUE を返却します

getEdenFreeMemory メソッド

説明

Eden 領域の空きサイズを取得します。

形式

```
getEdenFreeMemory();
```

パラメタ

なし

例外

なし

戻り値

Eden 領域の空きサイズ（バイト数）を long 型で返却します。

getEdenMaxMemory メソッド

説明

Eden 領域の最大使用サイズを取得します。

形式

```
getEdenMaxMemory();
```

パラメタ

なし

例外

なし

戻り値

Eden 領域の最大使用サイズ（バイト数）を long 型で返却します。

注意事項

G1GC を使用している場合，getEdenMaxMemory()，getSurvivorMaxMemory()を呼んだときの返却値は-1 となります。また，getTenuredMaxMemory()を呼んだときの返却値は Java ヒープ領域の最大サイズとなります。

getEdenTotalMemory メソッド

説明

Eden 領域の使用可能サイズを取得します。

形式

```
getEdenTotalMemory();
```

パラメタ

なし

例外

なし

戻り値

Eden 領域の使用可能サイズ（バイト数）を long 型で返却します。

getMetaspaceFreeMemory メソッド

説明

Metaspace 領域の空きサイズを取得します。

形式

```
getMetaspaceFreeMemory();
```

パラメタ

なし

例外

なし

戻り値

Metaspace 領域の空きサイズ（バイト数）を long 型で返却します。

getMetaspaceMaxMemory メソッド

説明

Metaspace 領域の最大使用サイズを取得します。

形式

```
getMetaspaceMaxMemory();
```

パラメタ

なし

例外

なし

戻り値

Metaspace 領域の最大使用サイズ（バイト数）を long 型で返却します。

getMetaspaceTotalMemory メソッド

説明

Metaspace 領域の使用可能サイズを取得します。

形式

```
getMetaspaceTotalMemory();
```

パラメタ

なし

例外

なし

戻り値

Metaspace 領域の使用可能サイズ（バイト数）を long 型で返却します。

getSurvivorFreeMemory メソッド

説明

Survivor 領域の空きサイズを取得します。

形式

```
getSurvivorFreeMemory();
```

パラメタ

なし

例外

なし

戻り値

Survivor 領域の空きサイズ（バイト数）を long 型で返却します。

getSurvivorMaxMemory メソッド

説明

Survivor 領域の最大使用サイズを取得します。

形式

```
getSurvivorMaxMemory();
```

パラメタ

なし

例外

なし

戻り値

Survivor 領域の最大使用サイズ（バイト数）を long 型で返却します。

注意事項

G1GC を使用している場合，getEdenMaxMemory()，getSurvivorMaxMemory()を呼んだときの返却値は-1 となります。また，getTenuredMaxMemory()を呼んだときの返却値は Java ヒープ領域の最大サイズとなります。

getSurvivorTotalMemory メソッド

説明

Survivor 領域の使用可能サイズを取得します。

形式

```
getSurvivorTotalMemory();
```

パラメタ

なし

例外

なし

戻り値

Survivor 領域の使用可能サイズ（バイト数）を long 型で返却します。

getTenuredFreeMemory メソッド

説明

Tenured 領域の空きサイズを取得します。

形式

```
getTenuredFreeMemory();
```

パラメタ

なし

例外

なし

戻り値

Tenured 領域の空きサイズ（バイト数）を long 型で返却します。

getTenuredMaxMemory メソッド

説明

Tenured 領域の最大使用サイズを取得します。

形式

```
getTenuredMaxMemory();
```

パラメタ

なし

例外

なし

戻り値

Tenured 領域の最大使用サイズ（バイト数）を long 型で返却します。

注意事項

G1GC を使用している場合，getEdenMaxMemory()，getSurvivorMaxMemory()を呼んだときの返却値は-1 となります。また，getTenuredMaxMemory()を呼んだときの返却値は Java ヒープ領域の最大サイズとなります。

getTenuredTotalMemory メソッド

説明

Tenured 領域の使用可能サイズを取得します。

形式

```
getTenuredTotalMemory();
```

パラメタ

なし

例外

なし

戻り値

Tenured 領域の使用可能サイズ（バイト数）を long 型で返却します。

10.6 Explicit メモリブロックを制御する処理のエラーチェック（共通エラーチェック）

有効な Explicit メモリブロックを示していない場合，Explicit ヒープを操作する多くの API は処理できなくなります。そこで，各 API 共通のエラーチェックルーチンを定義して，API の処理ができるかどうか判断します。共通エラーチェックは，各 API の処理対象である Explicit メモリブロックの状態によって，API を処理できるかどうかを判断します。共通エラーチェックで返却される値は次のとおりです。

true：

API の処理を続けることができると判断します。処理対象である Explicit メモリブロックの状態が有効な場合に返却されます。

false：

API を処理できないと判断します。処理対象である Explicit メモリブロックの状態が無効な場合に返却されます。

InaccessibleMemoryAreaException（例外クラス）：

サポートしていない機能を実行しようとした場合にスローされます。
InaccessibleMemoryAreaException クラスの詳細は，「10.7　例外クラス」を参照してください。
なお，処理対象の Explicit メモリブロックが次の状態である場合にスローされます。

- 解放済みの場合
- 解放予約済み，または自動解放明示予約済みの場合
- 有効，無効，解放済み，および解放予約済み以外の状態の場合

10.7　例外クラス

JavaVM で使用する API で発生する例外を表す，例外クラスについて説明します。

例外クラスの一覧を次に示します。

表 10–2　JavaVM で使用する API で発生する例外クラス

例外クラス名	説明
JP.co.Hitachi.soft.jvm.MemoryArea.MemoryManagementException	JP.co.Hitachi.soft.jvm.MemoryArea パッケージで定義する例外クラスの基底クラスです。Java プログラムで，生成またはスローすることは想定しません。 派生クラスは，InaccessibleMemoryAreaException クラスです。
JP.co.Hitachi.soft.jvm.MemoryArea.InaccessibleMemoryAreaException	MemoryArea クラスのインスタンスに対して，サポートしていない機能を実行しようとした場合にスローします。 例えば，削除予約済みまたは削除済みの ExplicitMemory クラスのインスタンスに対する削除操作が該当します。 Java プログラムで，生成またはスローすることは想定しません。 基底クラスは，MemoryManagementException クラスです。

11 アプリケーション開発時に使用できるプロパティ

この章では，アプリケーションを開発するときに使用できるプロパティについて説明します。

11.1　バッチアプリケーションで使用できるプロパティ

ここでは，バッチアプリケーションを開発するときに使用できるプロパティについて説明します。

ejbserver.batch.currentdir プロパティ

説明

バッチ実行コマンド（cjexecjob）を実行したカレントディレクトリの絶対パスを取得します。

バッチアプリケーション内で使用してください。

使用例

使用例を示します。

```
File f = new File(System.getProperty(”ejbserver.batch.currentdir”) + System.getProperty(”file.separator”) + ”DataFile.txt”);
```

付録

付録 A　Java ヒープメモリのリークを起こしやすい JavaAPI クラス

JavaAPI クラスを使用した場合，JavaAPI クラスのインスタンスが Java ヒープメモリ上に作成されます。しかし，表 A-1 および表 A-2 に示す JavaAPI クラスのメソッドを使用して，表の条件に該当する場合，JavaAPI クラス以外のインスタンスも Java ヒープメモリ上に作成されます。

作成されたユーザ作成クラスのインスタンスは，JavaAPI クラスが削除されるまで Java ヒープメモリ上に残ります。このため，ユーザが意識しない間に Java ヒープメモリの使用量が増え，Java ヒープメモリがリークするおそれがあるので注意してください。

ただし，表 A-2 の JavaAPI クラスについては，ユーザ作成クラスのインスタンスを削除することもできます。

Java ヒープメモリがリークしやすいクラス一覧について，インスタンスの削除方法がある場合とない場合に分けて表に示します。表 A-2 については，ユーザ作成のインスタンスの削除方法も示します。

参考

Java ヒープメモリのリークを起こしやすい JavaAPI クラスや条件，およびユーザ作成クラスのインスタンス削除方法については，ご使用のアプリケーションサーバのバージョンによって変わる場合があります。

表 A-1　Java ヒープメモリがリークしやすいクラス一覧（削除方法がない場合）

JavaAPI クラス名	Java ヒープメモリにユーザ作成クラスのインスタンスを作成する条件
java.lang.ClassLoader	defineClass()メソッドの引数に指定した ProtectionDomain がユーザ作成クラスと関連づけられている場合。
java.net.URL	URL クラスインスタンスを生成する際，URLStreamHandlerFactory が作成する URLStreamHander クラスがユーザ作成クラスの場合。
java.text.DateFormat	DateFormat クラスを継承したユーザ作成クラスのコンストラクタで super()を呼び出した場合。
java.util.logging.Level	Level クラスを継承したユーザ作成クラスのコンストラクタで super()を呼び出した場合。
java.util.logging.LogManager	addLogger()メソッドの引数に Logger クラスを継承したユーザ作成クラスを指定した場合。
java.util.logging.Logger	Logger クラスを継承したユーザ作成クラスで getAnonymousLogger()メソッドや setParent()メソッドを呼び出した場合。
javax.accessibility.AccessibleBundle	toDisplayString()メソッドの引数に指定したリソースバンドル名に対応するユーザ実装のクラスで，そのユーザ実装のクラス内で保持しておくキーと値のペアの値に，ユーザクラスのオブジェクトを設定している場合。
javax.print.attribute.standard.MediaSize	MediaSize クラスを継承したユーザ作成クラスのコンストラクタで super()を呼び出した場合。

JavaAPI クラス名	Java ヒープメモリにユーザ作成クラスのインスタンスを作成する条件
java.rmi.server.UnicastRemoteObject	exportObject()メソッドの引数(RMIClientSocketFactory,RMIServerSocketFactory)にユーザ作成クラスを指定した場合。

表 A-2　Java ヒープメモリがリークしやすいクラス一覧（削除方法がある場合）

JavaAPI クラス名	Java ヒープメモリにユーザ作成クラスのインスタンスを作成する条件	ユーザ作成クラスのインスタンス削除方法
java.beans.Introspector	getBeanInfo()メソッドの引数(beanClass)にユーザ作成クラスを指定した場合。	flushCaches()メソッドやflushFromCaches()メソッドを実行する。
java.beans.PropertyEditorManager	registerEditor()メソッドの引数(editorClass)にユーザ作成クラスを指定した場合。	registerEditor()メソッドのeditorClass に null を指定する。
java.util.logging.Logger	addHandler()メソッドの引数に Handler クラスを継承したユーザ作成クラスを指定した場合。	removeHandler()メソッドを実行する。
javax.imageio.ImageReader	addIIOReadWarningListener()メソッドの引数に IIOReadWarningListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOReadWarningListener()メソッドを実行する。
	addIIOReadProgressListener()メソッドの引数に IIOReadProgressListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOReadProgressListener()メソッドを実行する。
	addIIOReadUpdateListener()メソッドの引数にIIOReadUpdateListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOReadUpdateListener()メソッドを実行する。
javax.imageio.ImageWriter	addIIOWriteProgressListener()メソッドの引数に IIOWriteProgressListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOWriteProgressListener()メソッドを実行する。
	addIIOWriteWarningListener()メソッドの引数に IIOWriteWarningListener クラスを継承したユーザ作成クラスを指定した場合。	removeIIOWriteWarningListener()メソッドを実行する。
javax.naming.InitialContext	addToEnvironment()メソッドの引数(propVal)にユーザ作成クラスを指定した場合。	removeFromEnvironment()メソッドを実行する。
javax.naming.spi.NamingManager	getContinuationContext()メソッドの引数にユーザ作成クラスを指定した場合。	得られた Context の close()を実装して super()を実行する。
javax.print.attribute.HashAttributeSet	add()メソッドの引数にユーザ作成クラスを指定した場合。	remove(Attribute)メソッドやremove(Class)メソッドを実行する。

付録 B　JavaVM 内部で暗黙にスレッドを生成する JavaAPI クラス

Java SE の API を使用してスレッドの生成が起きるのは，通常は java.lang.Thread クラスや java.util.concurrent パッケージのスレッド関係のクラスを使用した場合です。

しかし，Java SE API の一部には暗黙にスレッドを生成するものがあります。これらの API を使用した場合，意図しないでスレッド数が増加しＣヒープ領域を消費することがあるため，注意が必要です。ここでは，Java SE 6.0 の JavaVM 内部でスレッドを生成する API や機能を示します。

付録 B.1　スレッド生成処理一覧

ここでは，次の API や機能が JavaVM 内部で生成するスレッドについて説明します。

- GUI 関連の API
- JMX 関連の API
- JNDI 関連の API
- RMI 関連の API
- そのほかの API や機能

(1)　GUI 関連の API

GUI 関連の API で，次のスレッドを生成します。

特定の条件なし

AWT または Swing の GUI 機能を使用すると，スレッドを生成することがあります。生成するスレッドは Java プロセスで最大六つです。

java.awt.EventQueue クラス

EventQueue インスタンスごとに，一つのスレッドを生成することがあります。

java.awt.FileDialog クラス

show()メソッドを呼び出すと，スレッドを一つ生成します。この呼び出しによるスレッドは FileDialog インスタンスごとに最大一つです。

java.awt.image.renderable.RenderableImageProducer クラス

startProduction()メソッドを呼び出すとスレッドを一つ生成します。

java.awt.print.PrinterJob クラス

次のメソッドを呼び出すと，スレッドを一つ生成します。

- print()（2 種両方）
- printDialog()（2 種両方）
- pageDialog()（2 種両方）

java.awt.TrayIcon クラス

UNIX 環境で java.awt.TrayIcon インスタンスを生成すると，インスタンスごとに，一つのスレッドを生成します。

javax.swing.JEditorPane クラス

JEditorPane インスタンスごとに，一つのスレッドを生成することがあります。

javax.swing.JFileChooser クラス

javax.swing.JFileChooser インスタンスごとに，一つのスレッドを生成することがあります。

javax.swing.JTable クラス

print()（5 種類すべて）を呼び出すと，スレッドを一つ生成します。

javax.swing.Timer

start()または restart()メソッドを呼び出すと，スレッドを一つ生成します。この呼び出しによるスレッドは Timer インスタンスごとに最大一つです。

javax.swing.text.LayoutQueue クラス

addTask()メソッドを呼び出すと，スレッドを一つ生成します。この呼び出しによるスレッドは LayoutQueue インスタンスごとに最大一つです。

javax.swing.text.JTextComponent クラス

print()（3 種類すべて）を呼び出すと，スレッドを一つ生成することがあります。

javax.swing.text.AsyncBoxView クラス

次のメソッドを呼び出すと，API 内部で LayoutQueue インスタンスを作成し，その延長でスレッドを生成することがあります。

- preferenceChanged()
- replace()
- setSize()

javax.swing.text.html.FormView クラス

submitData()メソッドを呼び出すと，スレッドを一つ生成します。

Applet の使用

UNIX 環境で Applet を使用して警告アイコンが表示されると，スレッドを一つ生成します。

インプットメソッドの使用

java.awt.im や java.awt.im.spi で提供されるインプットメソッドを使用するとスレッドを生成することがあります。このスレッドは Java プロセスで最大一つです。

(2) JMX 関連の API

JMX 関連の API で，次のスレッドを生成します。

SNMP(Simple Network Management Protocol)によるリソース監視

SNMP を使用してリソースを監視・管理している場合，最大で九つのスレッドを生成します。

javax.management.remote.rmi. RMIConnection インターフェース

RMIConnection インターフェースを実装したクラスのインスタンスごとに，一つのスレッドを生成することがあります。

javax.management.remote.rmi.RMIConnector クラス

connect()（2 種両方）メソッドを呼び出すと，確立した接続一つごとにスレッドを二つ生成します。

(3) JNDI 関連の API

JNDI 関連の API で，次のスレッドを生成します。

特定の条件なし

ネーミングやディレクトリの操作で，JNDI コンテキスト一つごとにスレッドを二つ生成します。

javax.naming.event.EventContext インターフェース

EventContext インターフェースを実装したクラスの addNamingListene()（2 種両方）メソッドを呼び出すと，一つのスレッドを生成することがあります。この呼び出しによるスレッドはディレクトリコンテキストごとに最大一つです。

(4) RMI 関連の API

RMI 関連の API で，次のスレッドを生成します。

RMI サーバ側

JavaVM 内部で次のスレッドを生成します。

- 特定の条件なしに最大六つのスレッドを生成します。このスレッドは JavaVM プロセスの終了まで維持されます。
- RMI クライアントから接続待機向けに，リモートオブジェクトをエクスポートした TCP ポート数分のスレッドを生成します。
- RMI クライアントからのメソッド呼び出しがあると，その制御のため一つのスレッドを生成します。
- java.rmi.server.Unreferenced インターフェースを実装したリモートオブジェクトの unreferenced()メソッド呼び出し向けに，一つのスレッドを生成します。

RMI クライアント側

接続している RMI サーバと同数のスレッドを生成します。

(5) そのほかの API や機能

そのほかの API や機能で，次のスレッドを生成します。

HTTP/HTTPS 通信

HTTP/HTTPS 通信をすると Keep Alive の制御のためスレッドを二つ生成します。このスレッドは Java プロセスで最大二つです。

DNS 通信

Windows 環境で DNS 通信をすると，スレッドを一つ生成することがあります。このスレッドは Java プロセスで最大一つです。

ファイナライザの明示実行

java.lang.System クラスや java.lang.Runtime クラスの runFinalization()を呼び出すとスレッドを一つ生成します。

外部プロセスの作成

UNIX 環境で java.lang.ProcessBuilder クラスなどによって外部プロセスを作成すると，スレッドを一つ生成します。このスレッドは java.lang.Process インスタンスごとに最大一つです。

java.nio.channels.Selector クラス

Windows 環境で Selector クラスを使用すると，Selector インスタンスへのチャネルの登録数が 1,024 増えるごとに，スレッドを一つ生成します。

java.util.prefs.Preferences クラス

Preferences クラスを使用すると，スレッドを一つ生成することがあります。このスレッドは Java プロセスで最大一つです。

javax.print.PrintServiceLookup クラス

javax.print.PrintServiceLookup を使用すると，スレッドを一つ生成します。このスレッドは Java プロセスで最大一つです。

sun.security.pkcs11.SunPKCS11 クラス

sun.security.pkcs11.SunPKCS11 インスタンスごとに，スレッドを一つ生成することがあります。

索引

記号

A

B

C

E

F

G

T

U

W

あ

か

せ

は

ゆ

れ