Java EE 再入門

To Java EE 7 from J2EE 1.4

November 2014

@minazou67

はじめに

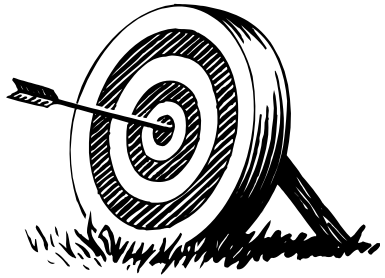
本スライドは、J2EE 1.4 から Java EE 7 の新機能について広く浅く解説し、短時間で Java EE に関する知識を網羅的に学習することを目的としています。

Java を学ぶきっかけになれば幸いです。

- 本スライドでは、Java の基本的な事項 (変数、演算子、制御構文、クラスなど) については取り扱いません。
- 本スライドで紹介している機能が、Java EE の新機能の全てではありません。
- 本スライドに掲載されている会社名、製品名、サービス名、ロゴは、各社・各団体の商標または登録商標です。

Target

- Java の知識が Java EE 5 付近で止まっている人
- Java 使ってるけど、実はよく知らないという人
- Java EE 初心者な人

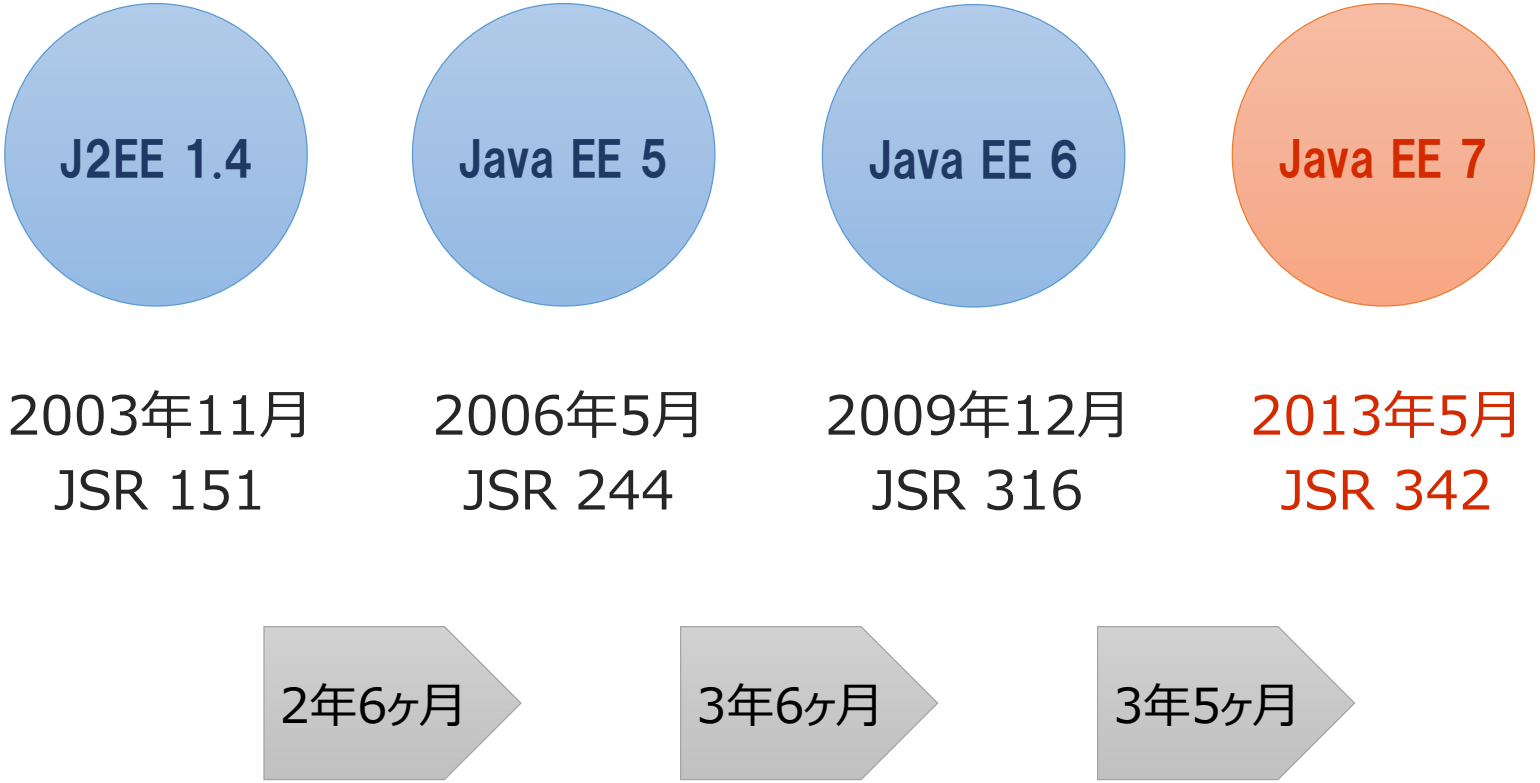


Goal

Java EE の各バージョンの更新内容を再確認することにより、既存の知識を整理すると共に、最新バージョンの Java EE について学び、よりモダンなプログラミングスタイルを身に付けましょう。

Java に関する
あれやこれや

Java EE Release History



Java 関連用語など

「Java SE 再入門」に書いてます。

JCP

JSR

JEP

JRE

JDK

JVM

JNI

JNDI

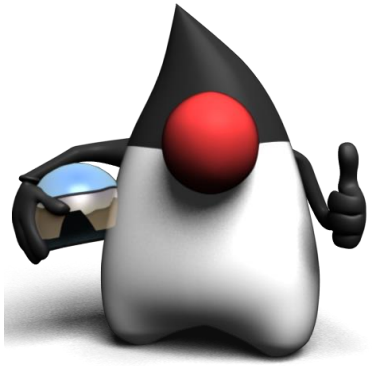
Java Byte Code

Reference Implementation

IDE

Java Platform

J2EE 1.4
(JSR 151)



J2EE 1.4 概要

- 2003年11月24日にリリース
- 主なテーマは、Web サービス
- 20件の Spec
- J2EE 1.3 と同様に、JCP のもとで仕様を考案
- WS-I Basic Profile 1.0 に対応
- Web サービス (SOAP) 機能を標準でサポート
- EJB の機能拡張 (Web サービス対応)
- JavaServer Pages (JSP) のメジャーアップデート
  (Expression Language 式の導入など)

WS-I Basic Profile

- WS-I (Web Services Interoperability Organization) は、Web サービスの相互運用性を図ることを目的としたオープンな団体
- 複数のプラットフォーム、開発言語、アプリケーション上で、Web サービスの相互運用を容易にするための利用法（プロファイル）を推進
- Basic Profile は、SOAP, WSDL, UDDI の基本仕様
- Attachments Profile は添付ファイルの仕様で、Basic Profile 1.1 と組み合わせて利用
- Basic Security Profile はセキュリティの仕様で、Basic Profile の拡張プロファイル

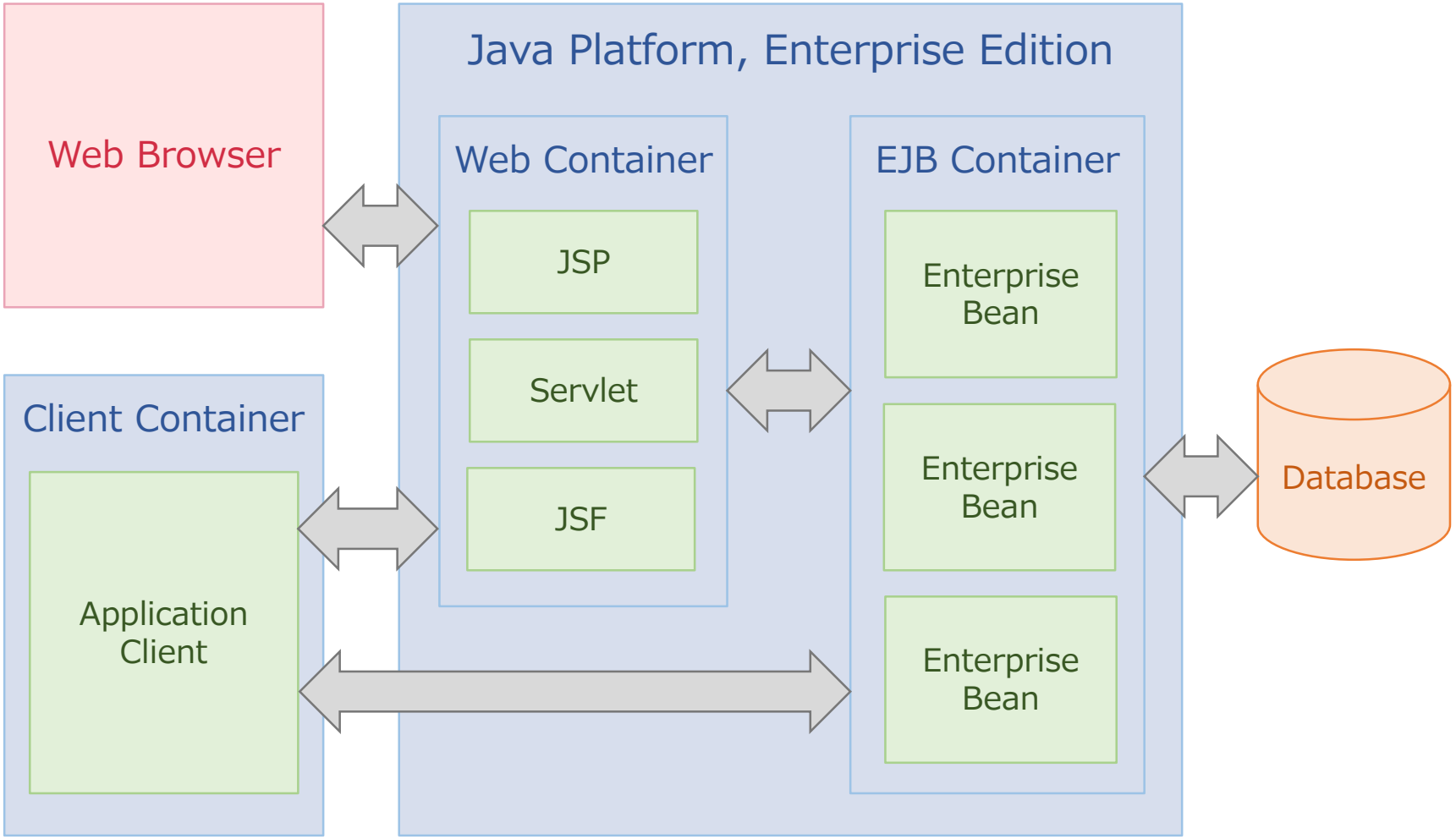
JAX-RPC

- JAX-RPC (Java API for XML-Based RPC) は、XML ベースの RPC (Remote Procedure Call) の標準仕様
- 分散型アプリケーションの構築が可能
- JAX-RPC 1.1 は、WS-I Basic Profile 1.0 に準拠し、SOAP 1.1 をサポート
- SOAP 1.1 over HTTP プロトコルで通信
- XML と Java オブジェクトとをバインディング
- JAX-RPC 1.1 の実装として、Apache Axis が有名

EJB

- EJB (Enterprise JavaBeans) は、分散オブジェクト指向に基づいたサーバコンポーネントモデル仕様
- 分散トランザクション管理やセキュリティ制限、ネーミングサービスなど、ビジネスロジックの実装に有用な機能をまとめて提供
- Java に特化した CORBA のサブセットのようなもの
- EJB コンテナの上で動作
- EJB 2.0 から導入された MDB (Message Driven Bean) は、非同期処理をサポート

EJB の利用イメージ


Web Browser
Java Platform, Enterprise Edition
Web Container
JSP
Servlet
JSF
EJB Container
Enterprise Bean
Enterprise Bean
Enterprise Bean
Client Container
Application Client
Database

EJB 2.1 の問題点

- 仕様が重厚長大で習得が困難
- デプロイメント記述子 (XML) の記述が冗長で複雑
- インターフェースの実装が必要で密結合化
- 不要なコールバック・メソッドの実装が必要で面倒
- データベースへのアクセスが面倒
- 開発生産性が低く、実行性能が悪い
- 環境構築やデプロイ、テストが面倒
- 若者の EJB 離れ

軽量コンテナと ORM

- 重量コンテナ (EJB) の代替として、軽量コンテナと ORM (Object Relational Mapper) が台頭
- 軽量コンテナは、DI (Dependency Injection) コンテナとも呼ばれる
- POJO (Plain Old Java Object) ベースでの開発が可能
- 代表的な軽量コンテナとしては、Spring Framework や Seasar が有名
- POJO と DI による疎結合化により、開発やテストが容易に
- Java EE の次期バージョンへ多大な影響を与えた

Java EE 5
(JSR 244)

Java EE 5 概要

- 2006年5月11日にリリース
- 主なテーマは、Ease of Development (EOD)
- 23件の Spec
- 名称が J2SE から Java EE に変更され、バージョン番号から小数点以下が削除された
- アノテーションを使用した POJO (Plain Old Java Object) ベースの開発が可能に
- Enterprise JavaBeans (EJB) を単純化
- Java Persistence API (JPA) の導入
- Web サービスの新しい仕様である JAX-WS を導入
- 新しい Web フレームワークとして JSF を正式導入

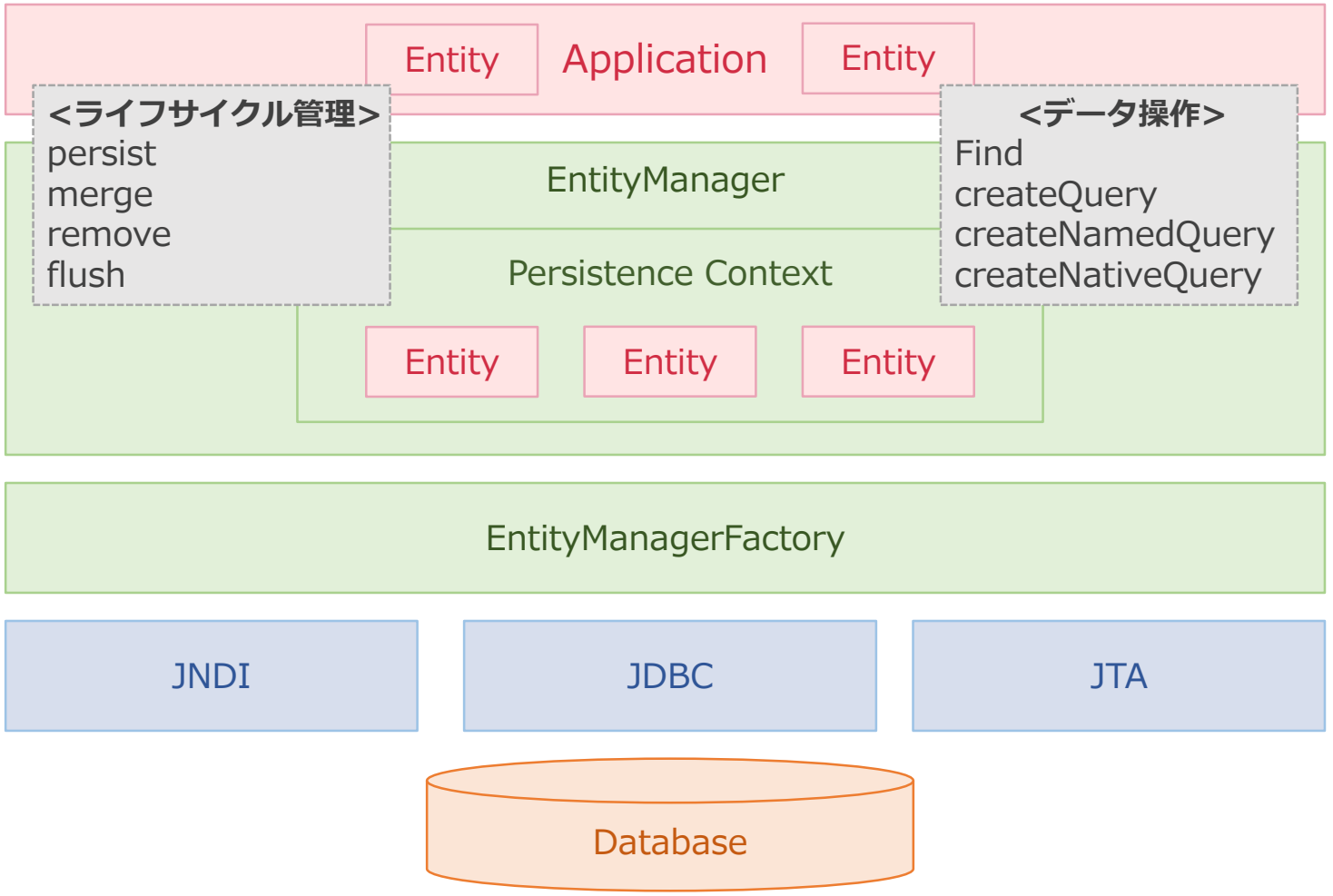
EJB 3.0

- POJO (Plain Old Java Object) ベースでの開発が可能
- コールバック・インターフェイスの実装が不要
- 例外処理の簡素化
- デフォルト値の設定によるコードの簡素化
- JNDI アクセスのカプセル化
- インターセプターの導入により、横断処理が可能
- アノテーションの導入により、XMLファイルを削減
- アノテーションと DI による疎結合化
- Java Persistence API (JPA) の導入によるデータベースアクセスの簡素化

Java Persistence API（JPA）

- Hibernate や JDO (Java Data Objects) の影響を受けた、Java EE 標準の ORM 仕様
- POJO ベースの O/R マッピングが可能
- EJB コンテナは不要で、Java SE 環境でも動作可能
- SQL によく似たクエリ言語である JPQL (Java Persistence Query Language) により、Entity に対する問い合わせが可能
- JPQL により永続化対象のデータベースに非依存
- Java EE 5 の JPA 1.0 は EJB 3.0 の仕様の一部
- Java EE 6 の JPA 2.0 で EJB 3.1 と分離

JPA の Architecture イメージ



JAX-WS

- JAX-WS (Java API for XML-Based Web Services) は、JAX-RPC 1.1 の後継となる Web サービスの仕様
- RPC 指向 (同期) からメッセージ指向 (非同期) へと変化
- JAX-RPC 1.1 と同様に SOAP 1.1 と WSDL 1.1 をサポート
- SOAP 1.2 と WS-I Basic Profile 1.1 もサポート
- アノテーションの使用が可能
- XML と Java オブジェクトとのデータバインディングには、JAXB を採用
- SOAP with Attachments (Sw/A) と MTOM の両方の添付ファイル仕様をサポート

JSF

- JSF (JavaServer Faces) は、 Web アプリケーション用のコンポーネントベースのフレームワーク
- MVC アーキテクチャー・パターンに準拠
- イベント駆動型のプログラミングモデルを採用
- JSF 独自の Tag Library を利用し、JSP ページを作成
- 再利用可能な「UIコンポーネント」を組み合わせて Web ページを構築
- カスタムコンポーネントを容易に作成可能
  (PrimeFaces には 100 以上のコンポーネントやスキンが存在)
- GUI ツールによる画面開発が可能
- 参照実装は、Mojarra

JSP 2.1

- 統合式言語 (Unified EL) の導入
  (JSF の EL 式と JSP の EL 式を統合)
- JSF で用いられていた遅延評価用の EL 式が使用可能
- trimDirectiveWhitespaces 属性が追加され、生成する HTML コードから余分なホワイトスペースを除去可能
- アノテーションによる Resource Injection をサポート
- J2SE 5.0 で追加された列挙型 (enum) をサポート
- カスタム ELResolver の作成が可能
  (Spring, EJB, JNDI などの標準ではサポートされていないオブジェクトの取り扱いが可能)

Java Servlet 2.5

- J2SE 5.0 が必須
- アノテーションのサポート (リソースの注入が可能)
- filter-mapping が複数サーブレット名指定、ワイルドカードに対応
- servlet-mapping が複数 URL パターンの指定に対応
- http-method に HTTP/1.1 の全メソッド名を使用可能
- getContextPath API の追加
- web.xml のルート要素に metadata-complete 属性を追加し、アノテーション処理の実施を制御可能に
- エラーハンドリング後のステータスコードの変更が可能

Java EE 6
(JSR 316)

Java EE 6 概要

- 2009年12月10日にリリース
- 主なテーマは、柔軟性・拡張性の向上と軽量化
- 28件の Spec
- Web Profile 形式での提供をサポート
- プルーニングの導入
- EJB の機能拡張と EJB Lite の導入
- RESTful Web Services のサポート (JAX-RS)
- JSF のメジャーアップデート
- CDI, Interceptors, Bean Validation の導入
- web.xml のオプション化

Web Profile

Web Profile は、Web アプリケーションに必要な API のみを集めた、Full Profile のサブセットです。

サブセット形式での提供により、アプリケーションサーバの軽量化が可能になりました。

Full Profile

JAXP
JAXM
JAXR
JAX-RPC
JAXB
JAX-WS
JAX-RS
JCA
JMS
EJB
JavaMail
Management
JACC
JASPIC

Web Profile

JSTL/JSP/EL
Servlet
JSF
Managed Beans
Common Annotations
Interceptors
JPA
EJB Lite
CDI/DI
Bean Validation
JTA

プルーニング

- 古くなった仕様を削減し、プラットフォームを軽量化するために導入した仕組み
- 仕様がプルーニングされると、次の Platform リリースでは必須コンポーネントではなくなり、オプション扱いになる可能性がある
- Java EE 6では、以下の仕様をプルーニング

JSR	Technology
JSR 093	JAXR (Java API for XML Registries)
JSR 088	Java EE Application Deployment
JSR 101	JAX-RPC (Java API for XML-Based RPC)
JSR 153	EJB Entity Bean

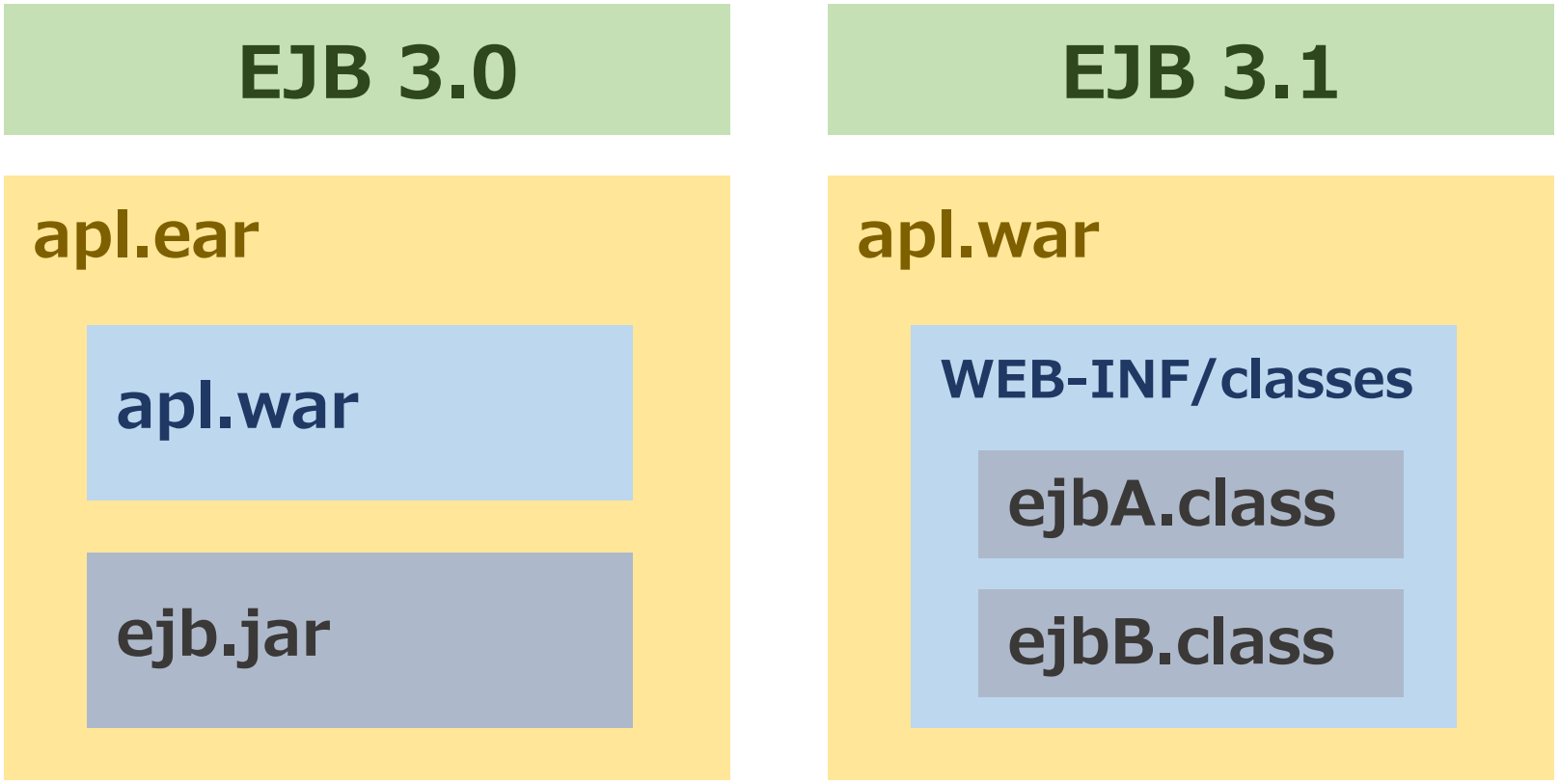
EJB 3.1

- パッケージングの簡略化 (war に直接梱包可能に)
- Singleton Session Beans が追加

  (スレッドセーフな唯一の Bean を簡単に実装可能に)
- タイマーサービスの仕様が大幅に強化

  (アノテーションでカレンダー形式のスケジューリングが可能)
- アノテーションで非同期処理を簡単に実装可能
- ローカルビジネスインタフェースの実装を省略可能
- JNDI 名を標準化し、移植性を向上
- 組み込み可能な EJB コンテナが提供され、Java SE 環境での単体テストを容易に実施可能

EJB パッケージの簡略化



- クラスパス上に配置すれば EJB として認識される
- EJB クラス群を jar 化して WEB-INF/lib に配置することも可能

EJB Lite

- EJB のサブセット仕様
- Full Profile と Web Profile の両形式のアプリケーションサーバで実行可能
- セキュリティが確保された、トランザクション対応のビジネスロジックを手軽に実装可能
- タイマーサービスや MDB (Message-Driven Bean)、非同期メソッド呼び出しは、使用できない
- インターセプターは使用可能

JAX-RS

- JAX-RS (Java API for RESTful Web Services) は、REST スタイルの Web サービスを提供するための仕様
- JAX-WS に代わる新しい Web サービスの仕様
- アノテーションベースで宣言的に実装可能
- リソースの要求を URI のパスで定義
- 参照実装は、Jersey

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path(value="/welcome")
public class WelcomeResource {
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String welcome() {
        return "welcome";
    }
}
```

JSF 2.0

- 新たなテンプレート言語である Facelets(xhtml) を導入
- Facelets の導入により、コンポーネントの再利用やデザイナとの分業が容易に
- Apache Tiles のようなテンプレート機能の追加
- Bean Validation (JSR-303) との統合
  (コンポーネント毎のエラー表示が可能)
- CDI (JSR-299) との統合
- 暗黙的なナビゲーションの導入
  (ナビゲーション定義用の XML ファイルが不要に)
- Ajax 対応 (JavaScript の知識は不必要)

CDI

- CDI (Contexts and Dependency Injection)は、コンテキスト (スコープ) を持った依存性注入の仕様
- バラバラだった DI の方法を全てのレイヤで統一
- スコープの種類は、Application, Session, Conversation, Request, Dependent
- 型に基づくタイプセーフなインジェクションが可能
- @Qualifier アノテーションで静的な型解決が可能
- @Produces アノテーションで動的な型解決が可能
- Java EE 6 の CDI 1.0 では beans.xml は必須
- 参照実装は、JBoss の Weld

JAX-RS + CDI の実装例

```
import javax.ejb.Stateless;
import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Stateless
@Path(value="/cdi")
public class SampleResource {
    @Inject
    private SampleService service = null;

    @GET
    @Path("/find/{id}")
    @Produces(MediaType.TEXT_PLAIN)
    public String update(@PathParam("id") long id) {
        service.find(id);
        return "Find! [" + id + "]";
    }
}
```

Interceptors

- Interceptors は、CDI で AOP (Aspect Oriented Programming) を実現するための仕様
- ビジネスロジックなどの Bean の呼び出しをインターセプトし、横断的な関心事を分離
- ログ出力や独自の認証チェックなどに使用可能
- アノテーションと Interceptor クラスを作成し、対象の Bean にアノテートすることで AOP が可能
- beans.xml に Interceptor を定義して有効化
- 複数の Interceptor をリスト定義した場合は、リストの上から順に適用される

Interceptors の実装例

```
@Inherited
@InterceptorBinding
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE, ElementType.METHOD})
public @interface Logable {
}
```

```
@Interceptor
@Logable
public class LoggingInterceptor implements Serializable {
    @AroundInvoke
    public Object invoke(InvocationContext ic) throws Exception {
        Logger logger = Logger.getLogger(ic.getTarget().getClass().getSuperclass().getName());
        logger.info(ic.getMethod().getName() + " start.");
        try {
            return ic.proceed();
        } finally {
            logger.info(ic.getMethod().getName() + " end.");
        }
    }
}
```

```
@Logable
@RequestScoped
public class SampleServiceImpl implements SampleService {
}
```

Bean Validation

- Bean Validationは、JavaBeans の値の妥当性をチェックするための仕様
- 検証のためのメタデータをアノテーションで指定
- Spring MVC や JSF, JAX-RS, JPA などで使用可能
- 標準で null、最小、最大、サイズ、正規表現、真偽値などのチェックが可能
- エラーメッセージは、properties ファイルやアノテーションの message 属性で変更可能
- 独自に作成したアノテーションによる検証も可能
- 参照実装は、Hibernate Validator

Bean Validation の実装例

```
public class SampleBean {
    @NotNull
    private String notNull;
    // Min・Max
    @Min(-10)
    int intMin;
    @Max(10)
    int intMax;
    // Decimal Min・Max
    @DecimalMin("-10.5")
    @DecimalMax("10.5")
    String stringMinMax;
    // Size
    @NotNull
    @Size(max = 10, message = "{max} 文字以内である必要があります。")
    String stringSize;
    @Size(min = 0, max = 10)
    List<Integer> listSize;
    // Pattern
    @Pattern(regexp = "[0-9]+")
    private String stringPattern;
    // getter/setter
}
```

Java Servlet 3.0

- アノテーションを利用した EoD
  (Servlet, InitParam, Filter, Listener, Security などのアノテーションを提供し web.xml は不要に)
- Web-Fragment 機能により web.xml の分割が可能
  (jar 内の web.xml をデプロイ時にマージ可能)
- Web リソースのモジュール化
  (WEB-INF/lib に 静的コンテンツや JSP を纏めた jar を配置可能)
- 非同期処理のサポート (アノテーションで簡単に実装可能)
- 認証用 API (login, authenticate, logout) の追加
- マルチパート対応 (ファイルのアップロード処理が簡単に)
- HttpOnlyCookie, JSESSIONIDの変更, URL Rewriting の無効化のサポート

Java EE 7
(JSR 342)

Java EE 7 概要

- 2013年5月28日にリリース
- 主なテーマは、簡素化の強化と HTML5 サポート
- 30件の Spec (+3 Spec のオプション)
- WebSocket と JSON のサポート
- JAX-RS の仕様を拡張
- JavaServer Faces (JSF) の HTML5 対応
- Expression Language (EL) のメジャーアップデート
- Batch フレームワークの提供
- Concurrency Utilities (並行処理ユーティリティ) の提供
- Java Message Service API (JMS) の改善

Java API for WebSocket

- Java API for WebSocket は、Java で WebSocket 通信するための API 仕様
- WebSocket は、RFC 6455 で定義された TCP ベースのプロトコルで、サーバー・クライアント間での双方向通信が可能なプロトコル
- WebSocket で通信することで、通信コスト削減、サーバープッシュが容易などのメリットがある
- WebSocket の利用には、モダンなブラウザとアプリケーションサーバーが必要
- アノテーション・ベースのプログラミングをサポート
- Java SE 環境でも利用可能

WebSocket のサーバ側の実装例

```
@ServerEndpoint("/websocket")
public class SampleWebSocketEndpoint {
    private static final Logger LOG =
        Logger.getLogger(SampleWebSocketEndpoint.class.getName());
    @OnOpen
    public void onOpen(final Session session) {
        LOG.info("On open");
    }
    @OnMessage
    public void onMessage(final String message, final Session session) {
        LOG.info("On message [" + message + "]");
        try {
            session.getBasicRemote().sendObject("Received [" + message + "]");
        } catch (IOException | EncodeException e) {
            LOG.severe(e.getMessage());
        }
    }
    @OnClose
    public void onClose(final Session session) {
        LOG.info("On close");
    }
}
```

JSON-P

- JSON-P (Java API for JSON Processing) は、JSON をパース・生成・問い合わせするための API 仕様
- Streaming API と Object Model API の2種類の API が存在
- Streaming API は低レベルな API で、イベントベースの JsonParser と JsonGenerator を提供
- Object Model API は、JSON データをメモリ上にツリー構造で表現することにより、DOM のような操作が可能だが、Streaming API よりも遅く、メモリをより多く消費する
- Java オブジェクトと JSON のバインディング機能は、仕様に含まれていない

JAX-RS 2.0

- JAX-RS 1.1 の仕様を拡張
- クライアント API を提供
- リクエストとレスポンスに対するフィルターをサポート
- Message Body の加工を目的とするインターセプターをサポート
- CDI をサポート (EJB との併用が不要)
- Bean Validation をサポート
  (アノテーションによるパラメーター検証が可能)
- 非同期処理をサポート

JSF 2.2

- HTML5 サポート (HTML5 タグ・属性への対応)
- パススルー機能の導入 (未定義のタグ属性をスルー)
- Faces フローの導入
  (画面フロー定義と対話スコープの提供)
- Resource Library Contracts の導入
  (テンプレートやCSSを動的に切り替え可能に)
- Stateless Views の導入
  (状態を保持しないことによるパフォーマンスの向上)
- ファイルアップロード機能の提供
- セキュリティトークン方式による CSRF 対策

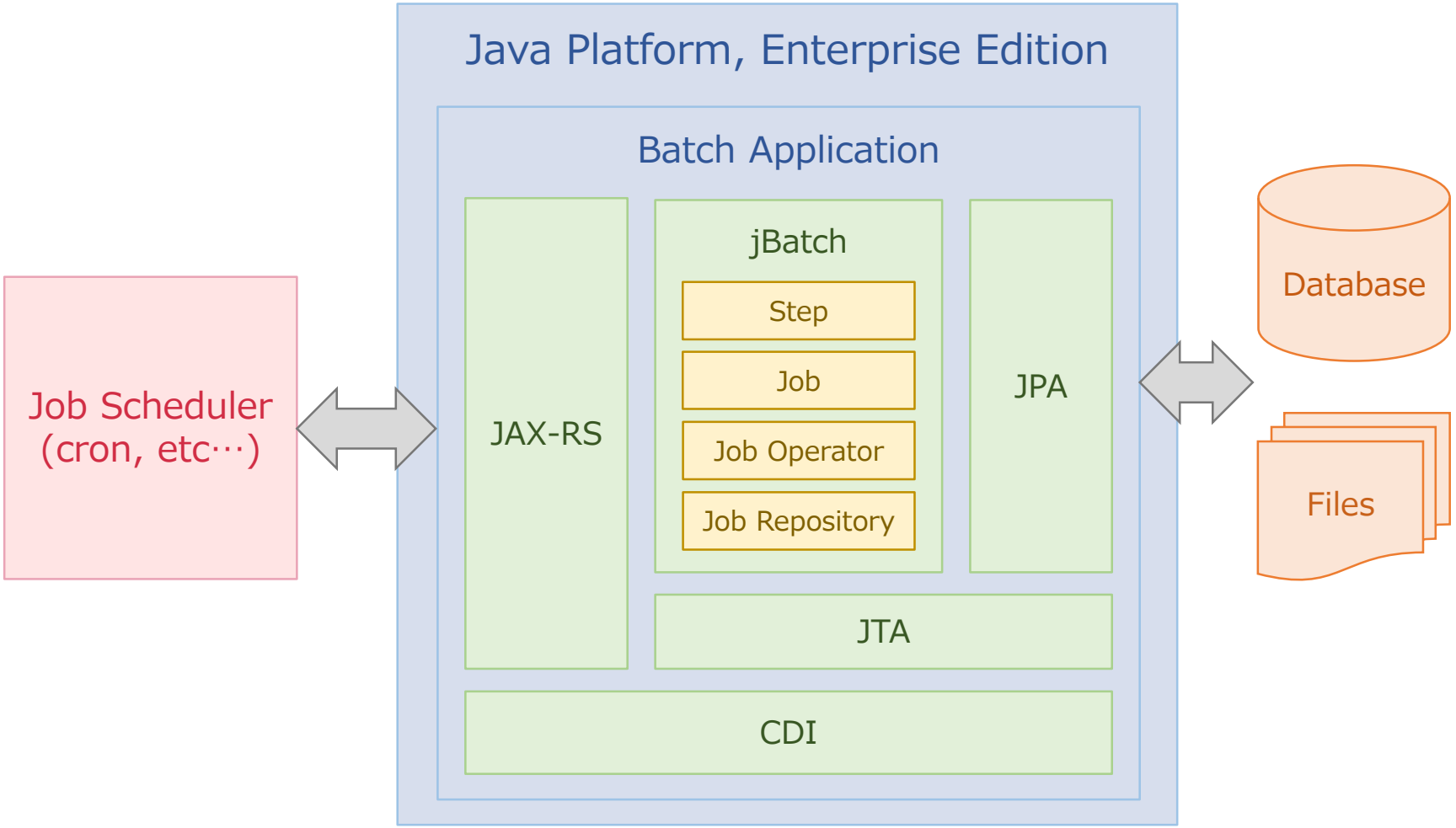
EL 3.0

- JSP の仕様から切り離され、個別の JSR として独立
- ラムダ式の使用が可能
- コレクション操作のための EL stream API を提供
- Static なフィールドとメソッドに直接アクセス可能
- スタンドアロンでの利用が可能になる ELProcessor クラスを提供 (JSP・JSF 以外からも利用が可能に)
- セミコロンオペレータ「;」で複数処理の記述が可能
- アサインメントオペレータ「=」で値の代入が可能
- 文字列連結オペレータ「+=」で文字連結が可能
- 独自の型コンバーターの利用が可能

Batch Applications for the Java Platform

- Java バッチフレームワークの標準仕様
- 仕様のベースは Spring Batch
- ジョブスケジューラー機能はスコープ外で、cron などから起動される想定
- ジョブやステップを XML で定義
- チェックポイントによる分割コミットやエラーハンドリング (Skip・Retry・Restart) 機能を提供
- アノテーション・ベースのプログラミングをサポート
- Java SE 環境でも利用可能

Batch Application のイメージ



JTA 1.2

- JTA (Java Transaction API) は、分散トランザクションをサポートしたトランザクション管理のための仕様
- コンテナ管理のトランザクション (CMT) と、Bean 管理のトランザクション (BMT) の2種類の管理方法が存在
- CMT では、トランザクション境界をコンテナが管理
- BTM では、トランザクション境界をコードで管理
- JTA 1.1 までは、EJB コンテナが必須
- JTA 1.2 (Java EE 7) では、@Transactional アノテーションを使うことで EJB コンテナが不要
  (CDI 管理の Bean を CMT 可能)

Java Servlet 3.1

- ノンブロッキングの I/O のサポート
  (ReadListener, WriteListener インターフェースの提供)
- プロトコルのアップグレード対応
  (upgrade API による WebSocket 対応)
- セキュリティ機能の強化
  (changeSessionId API によるログイン後のセッション ID の変更や、deny-uncovered-http-methods によるアクセスメソッドの制限など)

おまけ

Web Service のトレンド

SOA
(Service Oriented Architecture)

Microservice

SOAP
(XML)

REST
(JSON)

JAX-RPC
(SOAP 1.1)

JAX-WS
(SOAP 1.2)

JAX-RS

JAXB
(Java-XML)

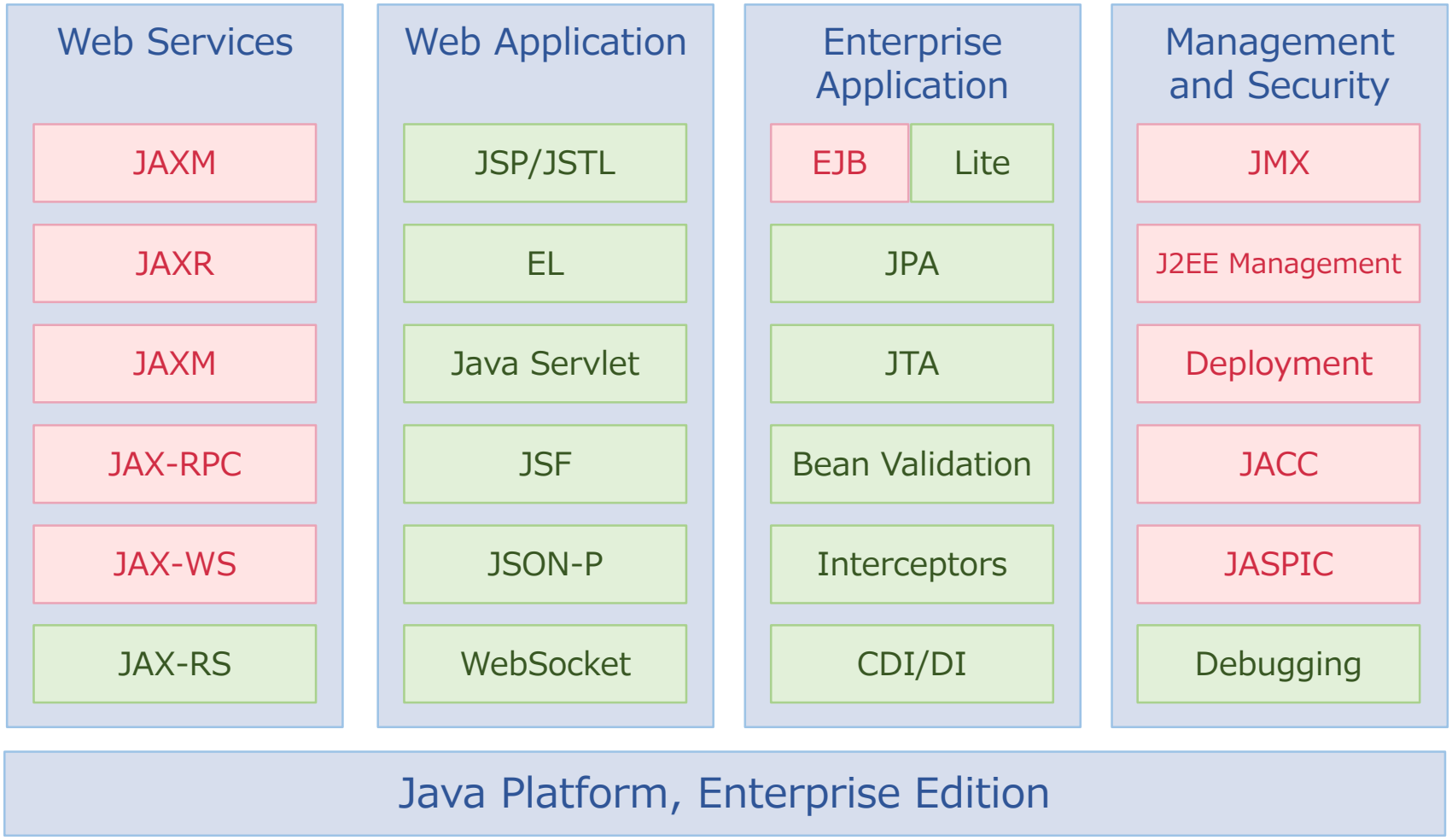
JSON-B
(Java-JSON)

Application Server の Java EE サポート状況

Application Server 名称	サポート状況
Oracle WebLogic Server 12c	Java EE 6, Java EE 7 (一部のみ)
IBM WebSphere Application Server V.8.5	Java EE 6
Red Hat JBoss EAP 6.3	Java EE 6
FUJITSU Interstage Application Server V11.0	Java EE 6
Hitachi uCosminexus Application Server V9	Java EE 6
NEC WebOTX Application Server V9.2	Java EE 6
Oracle GlassFish Server 3.1.2	Java EE 6
GlassFish 4.1 Open Source Edition	Java EE 7
WildFly 8 (旧 JBoss Application Server)	Java EE 7
Apache TomEE 1.7.1	Java EE 6 Web Profile

(2014年10月23日時点)

Java EE 7 全体イメージ



Matrix of the Java EE version and JSR
(Web Services Technologies)

Technology	JSR	EE 1.4	EE 5	EE 6	EE 7
Java API for XML Processing (JAXP)	063	1.2	-	-	-
	206	-	1.3 (SE)	1.4 (SE)	1.4 (SE)
Java APIs for XML Messaging (JAXM)	067	1.2	1.3	1.3	1.3
Java API for XML Registries (JAXR)	093	1.0	1.0	1.0	1.0 (Op)
Java API for XML-Based RPC (JAX-RPC)	101	1.1	1.1	1.1	1.1 (Op)
Implementing Enterprise Web Services	109	1.1	1.2	1.3	1.3
Streaming API for XML (StAX)	173	-	1.0	1.0 (SE)	1.0 (SE)
Web Services Metadata for the Java Platform	181	-	2.0	2.1	2.1
Java Architecture for XML Binding (JAXB)	222	-	2.0	2.2	2.2 (SE)
Java API for XML-Based Web Services (JAX-WS)	224	-	2.0	2.2	2.2
Java API for RESTful Web Services (JAX-RS)	311	-	-	1.1	-
	339	-	-	-	**2.0**

(太字は Web Profile 対象)

Matrix of the Java EE version and JSR (Web Application Technologies)

Technology	JSR	EE 1.4	EE 5	EE 6	EE 7
Standard Tag Library for JavaServer Pages (JSTL)	052	1.1	1.2	**1.2**	**1.2**
JavaServer Pages (JSP)	152	2.0	-	-	-
	245	-	2.1	**2.2**	**2.3**
Expression Language (EL)	245	-	-	**2.2**	-
	341	-	-	-	**3.0**
Java Servlet	154	2.4	2.5	-	-
	315	-	-	**3.0**	-
	340	-	-	-	**3.1**
JavaServer Faces (JSF)	127	1.1	-	-	-
	252	-	1.2	-	-
	314	-	-	**2.0**	-
	344	-	-	-	**2.2**
Java API for JSON Processing (JSON-P)	353	-	-	-	**1.0**
Java API for WebSocket	356	-	-	-	**1.0**

(太字は Web Profile 対象)

Matrix of the Java EE version and JSR
(Enterprise Application Technologies)

Technology	JSR	EE 1.4	EE 5	EE 6	EE 7
Managed Beans	316	-	-	**1.0**	-
	342	-	-	-	**1.0**
Concurrency Utilities for Java EE	236	-	-	-	1.0
Common Annotations for the Java Platform	250	-	1.0	**1.1**	**1.2**
Interceptors	318	-	-	**1.1**	**1.2**
Java EE Connector Architecture (JCA)	112	1.5	1.5	-	-
	322	-	-	1.6	1.7
Java Persistence API (JPA)	220	-	1.0	-	-
	317	-	-	**2.0**	-
	338	-	-	-	**2.1**
Java Message Service API (JMS)	914	1.1	1.1	1.1	-
	343	-	-	-	2.0

(太字は Web Profile 対象)

Matrix of the Java EE version and JSR
(Enterprise Application Technologies)

Technology	JSR	EE 1.4	EE 5	EE 6	EE 7
Enterprise JavaBeans (EJB)	153	2.1	-	-	-
	220	-	3.0	-	-
	318	-	-	**3.1**	-
	345	-	-	-	**3.2**
Contexts and Dependency Injection for Java (CDI)	299	-	-	**1.0**	-
	346	-	-	-	**1.1**
Dependency Injection for Java (DI)	330	-	-	**1.0**	**1.0**
Bean Validation	303	-	-	**1.0**	-
	349	-	-	-	**1.1**
Batch Applications for the Java Platform	352	-	-	-	1.0
Java Transaction API (JTA)	907	1.0	1.1	**1.1**	**1.2**
JavaMail API	919	1.3	1.4	1.4	1.5
JavaBeans Activation Framework (JAF)	925	1.0	1.1	1.1 (SE)	1.1 (SE)

(太字は Web Profile 対象)

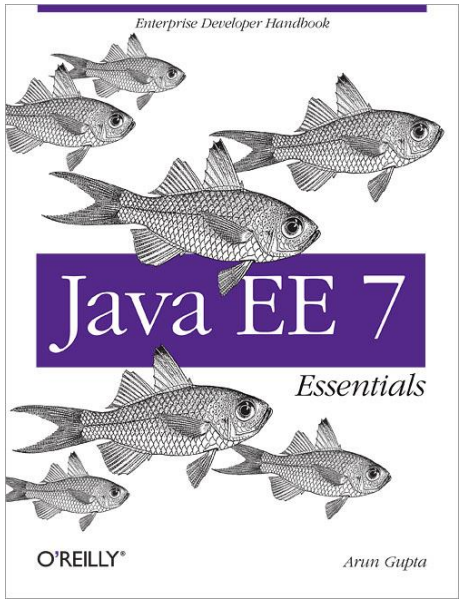
Matrix of the Java EE version and JSR (Management and Security Technologies)

Technology	JSR	EE 1.4	EE 5	EE 6	EE 7
Java Management Extensions (JMX)	003	1.2	1.4 (SE)	1.4 (SE)	2.0 (SE)
Debugging Support for Other Languages	045	1.0	1.0	**1.0**	**1.0**
J2EE Management	077	1.0	1.1	1.1	1.1
Java EE Application Deployment	088	1.1	1.2	1.2	1.2 (Op)
Java Authorization Contract for Containers (JACC)	115	1.0	1.1	1.4	1.5
Java Authentication Service Provider Interface for Containers (JASPIC)	196	-	-	1.0	1.1

(太字は Web Profile 対象)

Java EE 関連のお勧め書籍

タイトル：Beginning Java EE 6
出版社　：翔泳社
著者　　：Antonio Goncalves (著)
　　日本オラクル株式会社 (監訳)
発売日　：2012/03/08
ページ数：608ページ
価格　　：4,536円 (税込)



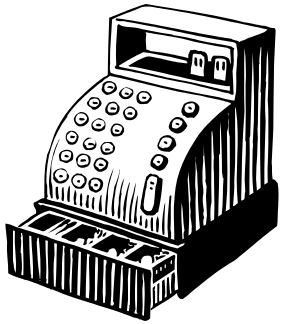
タイトル：Java EE 7 Essentials
出版社　：Oreilly & Associates Inc
著者　　：Arun Gupta (著)
発売日　：2013/9/6
ページ数：343ページ
価格　　：5,631円 (税込)
　　：2,040円 (税込) [Kindle 版]

Java EE 8

JSR 366 として正式にスタートし、2016年 Q3 にリリース予定。以下のような機能を提供予定。

JSR	仕様
JSR 107	JCACHE - Java Temporary Caching API キャッシュ機構を提供する標準 API
JSR 365	Contexts and Dependency Injection for Java 2.0 CDI のモジュール化 (Java SE 上でのコンテナ起動など)
JSR 367	Java API for JSON Binding (JSON-B) Java オブジェクトと JSON のバインディング
JSR 369	Java Servlet 4.0 HTTP/2 対応 (Request/Response の多重化や Server Push)
JSR 370	Java API for RESTful Web Services (JAX-RS 2.1) Server-Sent Event (SSE) のサポート
JSR 371	Model-View-Controller (MVC 1.0) アクションベースの MVC フレームワーク (テンプレートは対象外)

まとめ



Java EE の良いところ

- フルスタックな標準技術
- IDE のサポートが比較的優秀
- 知識の差分アップデートが可能
- アプリケーションサーバがライブラリを提供してくれるため、デプロイ用アーカイブのファイルサイズの肥大化が抑制でき、配備時間や起動時間の短縮が可能
- 有償サポートによる脆弱性対応
- プログラマの確保が比較的容易

Java EE の良くないところ

- コードが冗長になりがち
- OSS の後追いで進化のスピードが遅い
- 仕様が重厚長大で全体を把握するのが大変
- サーバの起動に時間が掛かる (過去の話)
- Java EE 準拠のサーバ間でもアプリケーションの互換性を維持できない場合がある
- 日本語の書籍が少ない
- 若者の Java EE 離れが顕著

最後に

世の中には多数のライブラリやフレームワークが存在しますが、システム特性や開発規模、メンバー構成などにより最適なアーキテクチャは異なります。

Java EE は、規模が大きく、ライフサイクルの長いアプリケーションの開発に適していると言われており、現時点でも主要な選択肢の一つです。

Modern Java
is the best Java.

おわり