ORACLE

Java EE 6 最新機能のご紹介

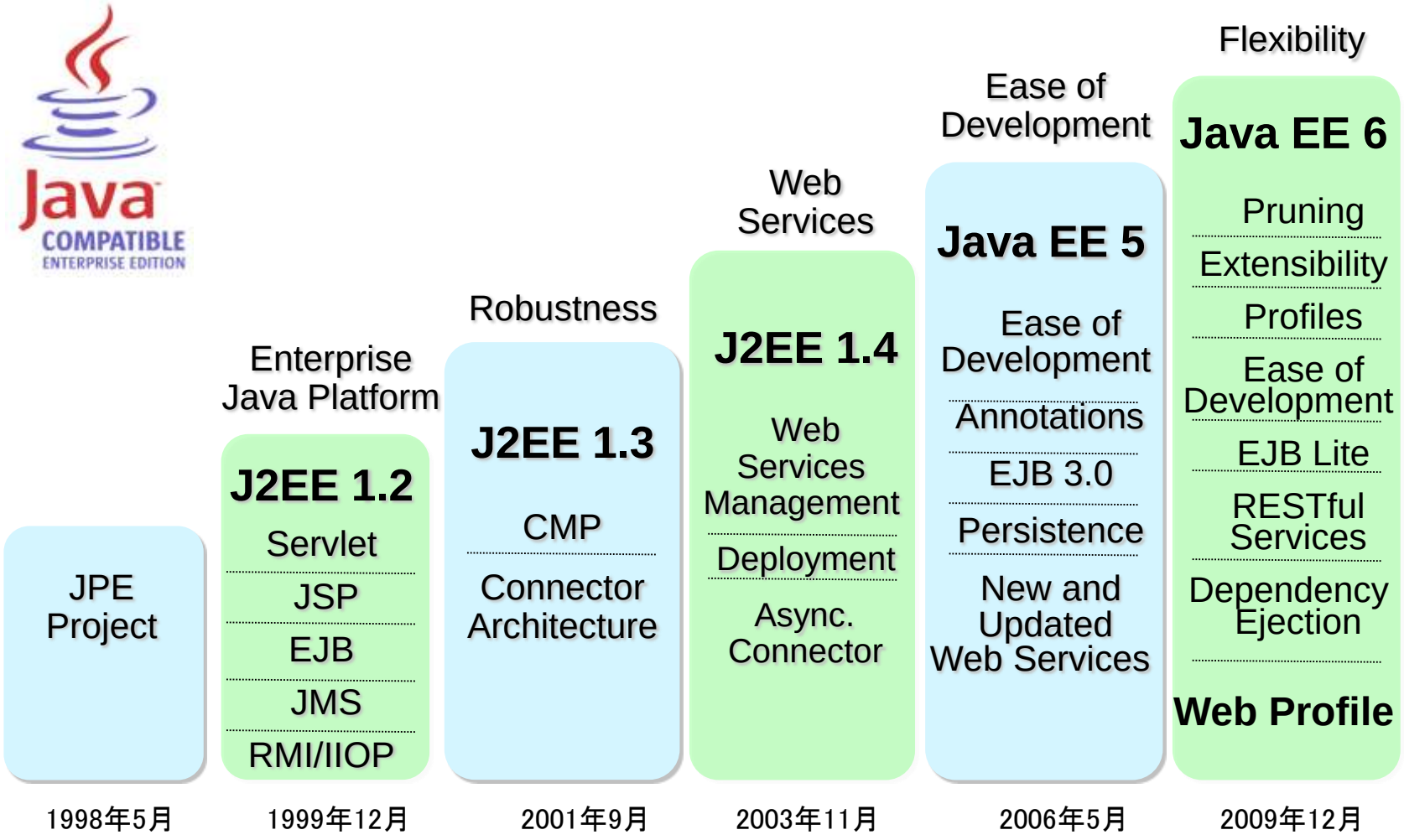
日本オラクル株式会社

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

Java EE 6のテーマ

- 拡張性の向上
  - アプリケーションの簡単な拡張
- 柔軟性の向上/軽量化
  - 目的に応じた必要十分な構成
  - プルーニング
- かんたん開発
  - 新技術の追加
    - DI, CDI, JAX-RS, Bean Validation
  - 更新された技術
    - Servlet 3.0, JPA2.0, EJB 3.1, JSF 2.0 等

Java EEの歴史



拡張性

Webアプリ開発における拡張性

- プラガビリティの向上
  - web.xmlの変更なしで3rd ライブラリ/フレームワークをプラグイン
  - フレームワーク開発者がフレームワーク内に設定
    - web-fragments.xmlに記述してjar/META-INFに含める
    - Servletにアノテーションを付与
  - これによりアプリ開発者はjarをwarに含めるのみで利用可能

Web Fragment

- jarファイルのMETA-INF/web-fragment.xml

```
<web-fragment
        version=”3.0”
        xmlns="http://java.sun.com/xml/ns/javaee">
    <servlet>
        <servlet-name>welcome</servlet-name>
        <servlet-class>WelcomeServlet</servlet-class>
    </servlet>
    <listener>
        <listener-class>RequestListener</listener-class>
    </listener>
</web-fragment>
```

目的に応じた必要十分な構成

プロファイリング

- Java EEの技術を用途毎に分割提供
- Java EE 6で提供されるプロファイル
  - Webプロファイル(Webの開発に特化)
  - Enterprise Platform(フルJava EE)

Webプロファイル

Web開発に特化した軽量プロファイル

- Java EE 6 リリース時に提供される最初のプロファイル
  - Servlet
  - JSP / EL
  - JSTL
  - JSF
  - Bean Validation
  - EJB Lite
  - JPA
  - JTA
  - DI/CDI
  - Managed Beans
  - Interceptors
  - Common Annotations

プルーニング

古くなった仕様の削減（2段階プロセス）

- 候補の策定後、その次のバージョンで削減
- 次期バージョン(Java EE 7)でオプション化
  - JAX-RPC(->JAX-WS)
  - EJB Entity Beans(->JPA)
  - JAXR
  - JSR-88

Servlet 3.0 – かんたん開発

- 概要
  - J2SE 5.0で追加された言語機能に対応
    - アノテーション
    - 型の安全性(Generics)
  - 使いやすいデフォルト値の定義によるゼロコンフィギュレーション
    - アノテーションによる定義
    - web.xmlはオプション化

Servlet 3.0 – かんたん開発

アノテーション

- アノテーションにより、Servlet、Filter、Listener、Security の定義が可能
  - @WebServlet – Defines a Servlet
  - @WebFilter – Defines a Filter
  - @WebListener – Defines a listener
  - @WebInitParam – Defines an init param
  - @ServletSecurity – security constraints
  - @MultipartConfig – file upload
- web.xmlでオーバーライド

Servlet 3.0 – かんたん開発

Servlet 2.5 example

```
/* Code in Java Class */

package com.foo;
public class MyServlet extends HttpServlet {
    public void doGet(HttpServletRequest req,
                HttpServletResponse res) {
        ...
    }

    ...
}
```

```
<!--Deploymentdescriptorweb.xml-->
<web-app>
    <servlet>
        <servlet-name>MyServlet</servlet-name>
        <servlet-class>
            com.foo.MyServlet
        </servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>MyServlet</servlet-name>
        <url-pattern>/myApp/*</url-pattern>
    </servlet-mapping>
    ...
</web-app>
```

Servlet 3.0 – かんたん開発

Servlet 3.0 example

```
package com.foo;
@WebServlet(name="MyServlet", urlPattern="/myApp/*")
public class MyServlet extends HttpServlet {

    public void doGet(HttpServletRequest req,
                      HttpServletResponse res)
    {
        ...
    }
```

Servlet 3.0 – その他新機能

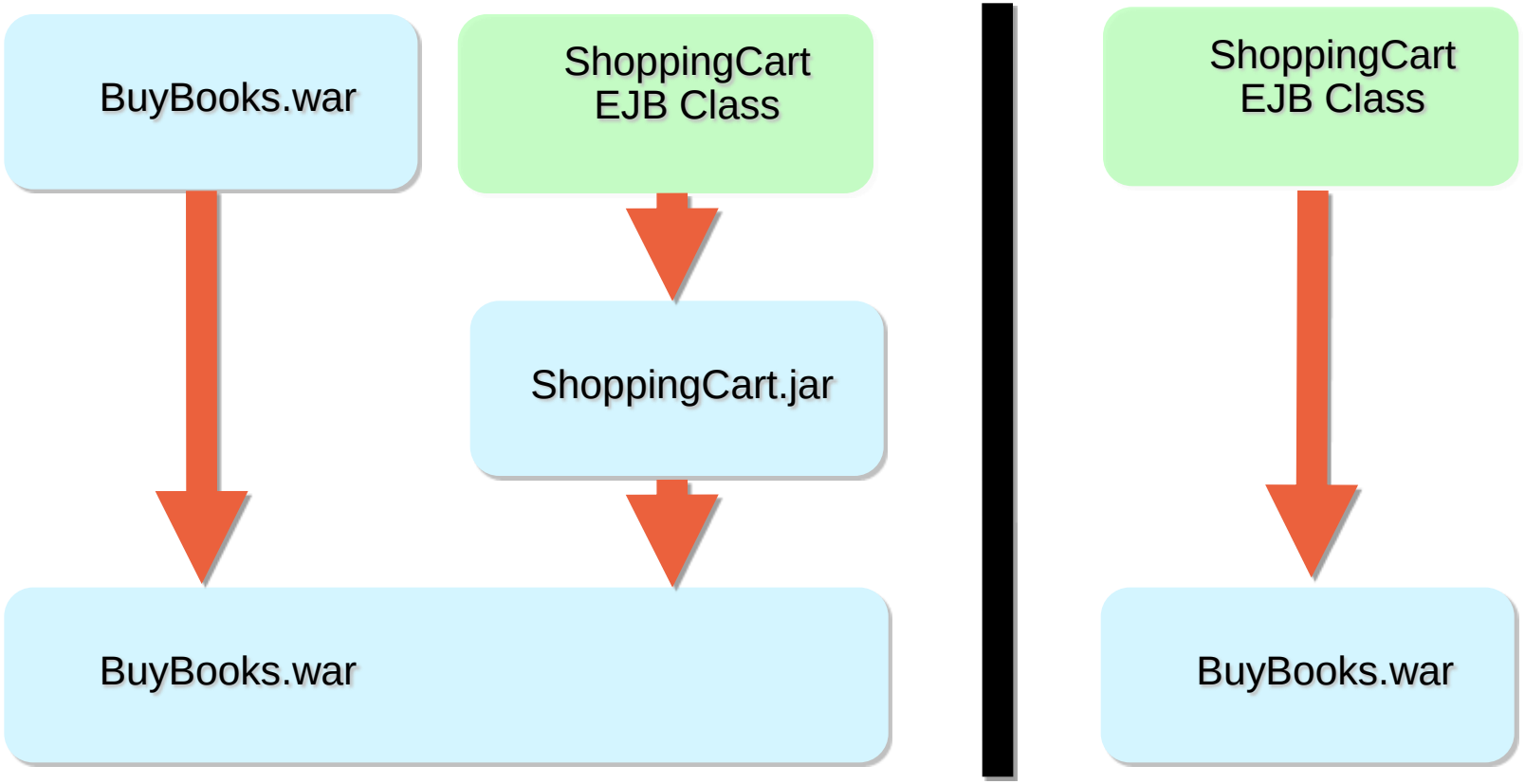
- マルチパート対応
  - ファイルアップロードに対応
  - @MultipartConfig
- 非同期処理のサポート
  - @WebServlet(asyncSupported=true)
  - コンテナによるスレッド管理
  - e.g. Comet, chat, push apps
- セキュリティの拡張
  - プログラミングによる認証、ログイン、ログアウトのサポート
    - HttpServletRequest.authenticate
    - HttpservletRequest.login
    - HttpServletRequest.logout
  - @ServletSecurity

EJB 3.1 – 特徴

- パッケージングの簡略化
- EJB 3.1 “Lite” の提供
- その他の新機能
  - ローカルビジネスインタフェースのオプション化
  - 移植可能なGlobal JNDI名
  - Java SEに組み込み可能なEJBコンテナ
  - Singleton Session Beansの追加
  - タイマーサービス
  - 非同期処理

EJB 3.1 – シンプルなパッケージング

- warに直接梱包可能



EJB 3.1 “Lite”の提供

- Full EJB 3.1機能のサブセットを提供

- Lite
  - ローカルセッションBeans
  - CMT/BMT
  - Declarative Security
  - Interceptors
- Full = Lite +
  - Message-Driven Beans
  - Web Service Endpoint
  - 2.x/3.x Remote view
  - RMI-IIOP Interoperability
  - Timer Service
  - Async method call
  - 2.x Local view
  - CMP/BMP Entity

EJB 3.1 – その他の新機能

- Local Business Interfaceのオプション化
  - インタフェースの定義が不要
- 移植可能なGlobal JNDI名
  - ベンダー固有のJNDI名を標準化されたGlobal JNDI名に統一
  - 移植性の向上
- Singleton Session Beansの追加
  - 並列アクセス処理対応
  - Startup / Shutdownコールバック機能の追加
- タイマーサービス
  - 自動タイマー生成/カレンダーベースタイマー
- 非同期処理

RESTful ウェブサービス

JAX-RS 1.1

- すでに広く採用されている
- ハイレベルHTTP API
- アノテーションベースのプログラミングモデル
- 必要に応じてプログラム可能なAPIも利用可能

JAX-RS リソースクラス

- @Path アノテーションにより識別

```
@Path(“widgets/{id}”)
@Produces(“application/widgets+xml”)
public class WidgetResource {
        public WidgetResource(@PathParam(“id”)
                                        String id) { … }
        @GET
        Widget getWidget() { … }
}
```

Standard Validation API

Bean Validation 1.0

- JSF、JPAと統合
- アノテーションによる制約の表現
  - @NotNull
  - @Size(max=40) String address;
- カスタム・バリデーター
  - e.g.Emailバリデーターをカスタムで作成する
    - Custom validator classの作成
    - Custom validator methodのBeanへの追加
  - @Email String recipient;

Persistence

JPA 2.0

- エンティティではないCollectionフィールドの永続化
  - @ElementCollection
  - @CollectionTable
- JPQL 拡張
  - e.g. CASE WHEN, NULLIF, COALESCE
- 動的クエリ生成のためのクライテリアAPI
- 悲観的ロック

Criteria APIサンプル

```
EntityManager em = ... ;
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Employee> cq =
    cb.createQuery(Employee.class);
Root<Employee> emp = cq.from(Employee.class);
cq.select(emp);
cq.where(cb.equal(emp.get(Employee_.lastName), "Smith"));
TypedQuery<Employee> query = em.createQuery(cq);
List<Employee> rows = query.getResultList();
```

JavaServer Faces 2.0 – 特徴

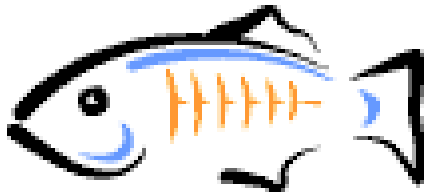
- Faceletsの採用
- アノテーション
  - @ManagedBean/@RequestScope/@SessionScope
- faces-config.xmlオプション化
  - ManagedBeanのアノテーション化
  - JSFナビゲーションを改良
    - ボタン/リンク名とXHTMLファイル名のマッチング
  - その他の定義にはfaces-config.xmlが必要
- 標準リソースフォルダ（css/js/images etc）
  - Resourcesフォルダ、warのルートもしくはMETE-INF配下
    - /resources/scripts/, /resources/css/, /resources/img/
- その他: Ajax対応/ブックマーク可能なURL

Dependency Injection

CDI 1.0

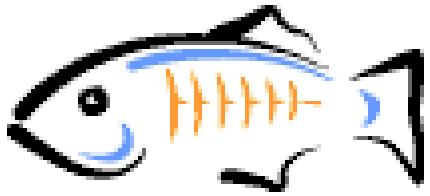
- 新たな@Inject アノテーション
  - @Inject @LoggedIn User user;
- Injection メタモデル
- どんなBeanもInject対象
  - EJB session beans
  - Plain classes with @ManagedBean
  - CDIがモジュール内で見つけたクラス
- デフォルトで無効、有効化する場合は、beans.xmlを配置
  - META-INF/、WEB-INF/に配置

Demo



Demo内容

- Oracle Enterplize pack for Eclipse + GlassFish の設定
- シンプルなJava EE 6 アプリケーションの作成（Servlet, EJB）
- Java Persistence API 2を利用したDB読み込み
- Java Server Faces 2 – Faceletsの利用
- JAX-RSによるRESTfulウェブサービスの作成



Summary

Java EE6 Platform

- かんたん開発がより強力に
  - 今までできたことはよりかんたんに、
  - できなかったこともかんたんにできるように
- より拡張性が向上
- より柔軟に

GlassFish + OEPE

- Eclipse上でJava EE 6アプリケーションの開発が可能
- OEPE 11.1.1.6 downlaod

  http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/index.html

ORACLE

SOFTWARE. HARDWARE. COMPLETE.

ORACLE


ORACLE®