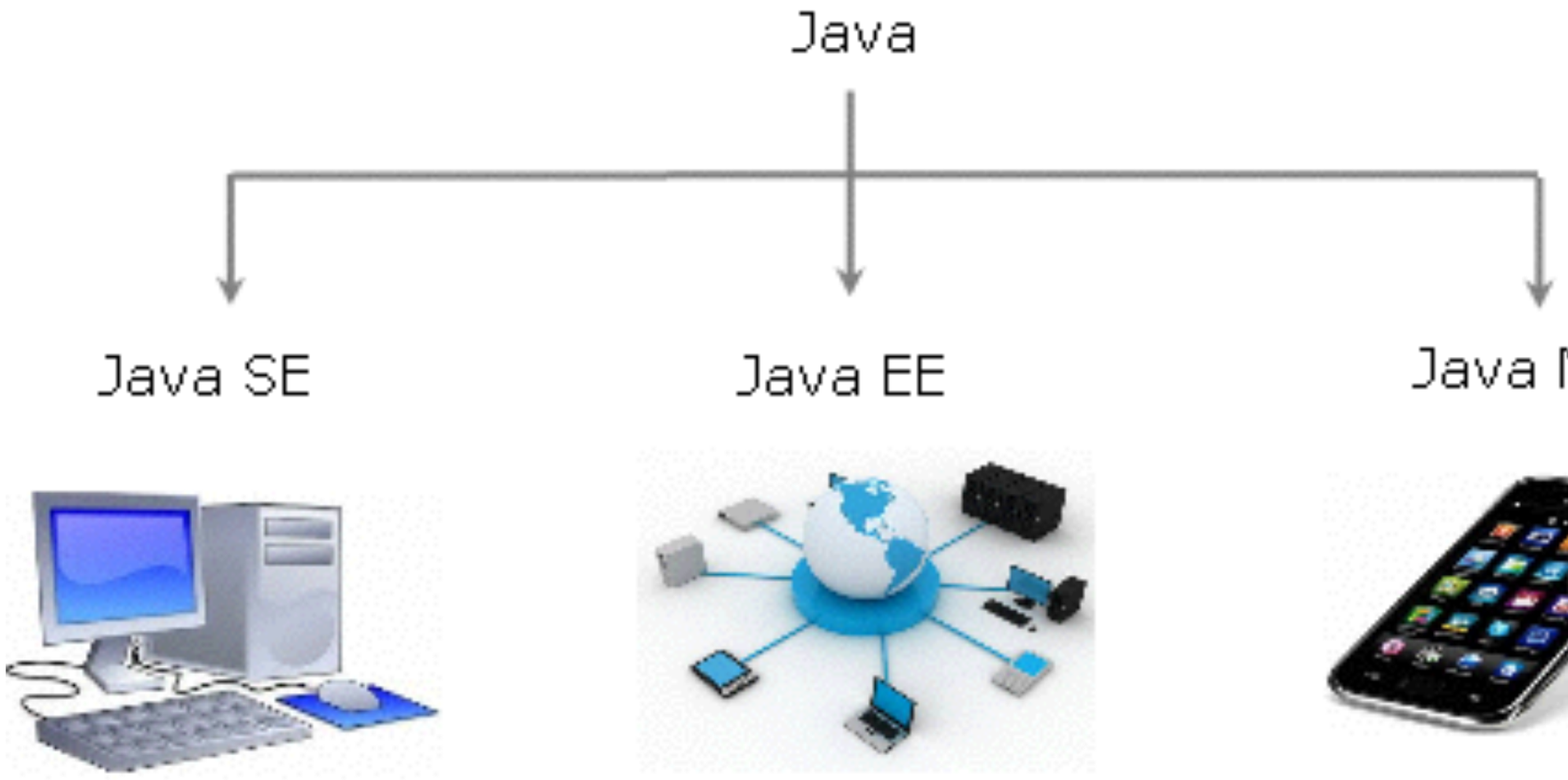
Applications client-serveur avec Java Enterprise Edition

Frederico Alvares
Hervé Grall

Les plateformes Java



Java SE / EE

- Pour manipuler une collection d'objets
  - Pas besoin de ré-créer des classes pour gérer des *hashtables*, des listes, des E/S (ex. : fichiers), des sémaphores, etc.
- Pour implémenter une application web/distribuée?
  - Pas besoin de ré-créer des classes ou modules pour gérer la communication, la persistance de données, les transactions, la concurrence

Java : la gouvernance

- Les plateformes Java évoluent selon un processus bien défini
- Le “Java Community Process” (JCP) décrit le processus d’évolution des plateformes Java
  - Des procédures formelles pour développer les spécifications Java
- Sujet à une demande de spécification - JSR: *Java Specification Request*
  - Avancement disponible sur le site http://www.jcp.org
- Exemple : JSR 372 -> JavaServer Faces 2.3

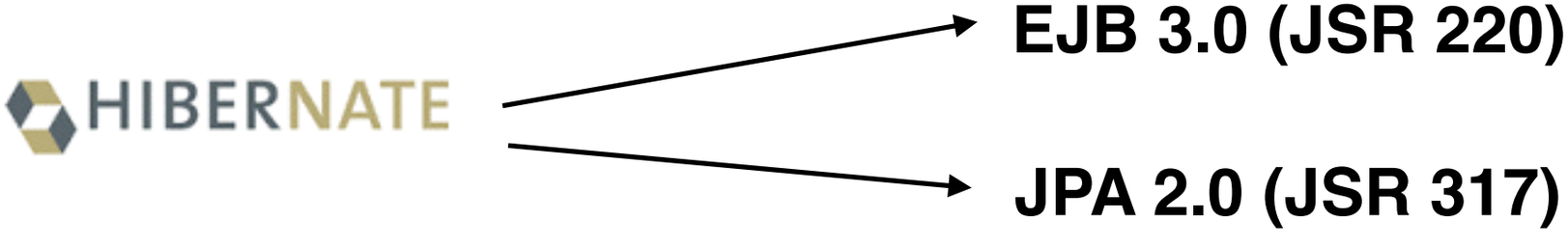
Frameworks Java ≠ composants normalisés

- Exemple : frameworks de présentation

Non normalisé
Pas de JSR

Normalisé
JSR 314 -> JSF 2.0

- Les frameworks influent l'evolution des plateformes

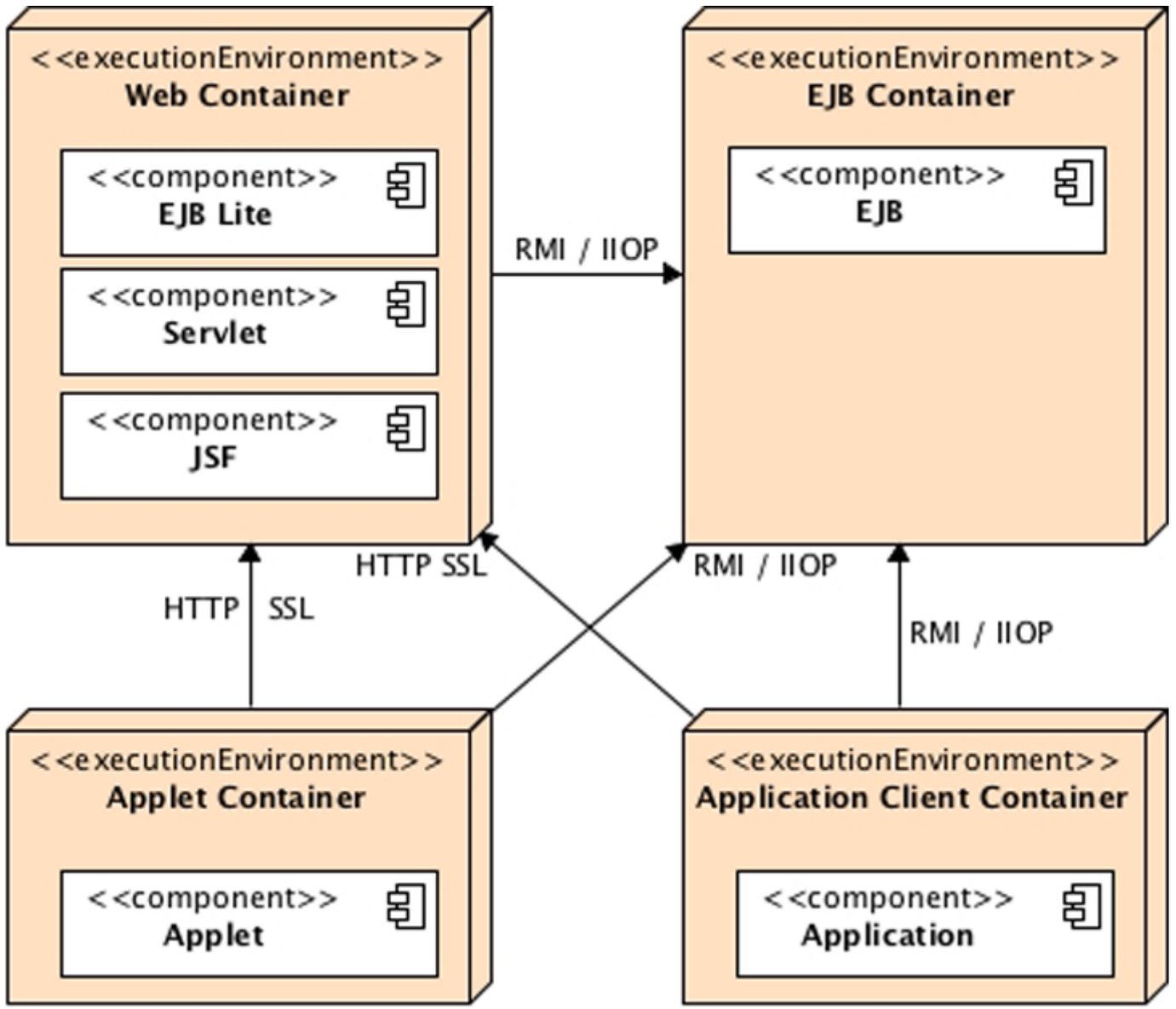


Java EE, c'est quoi ?

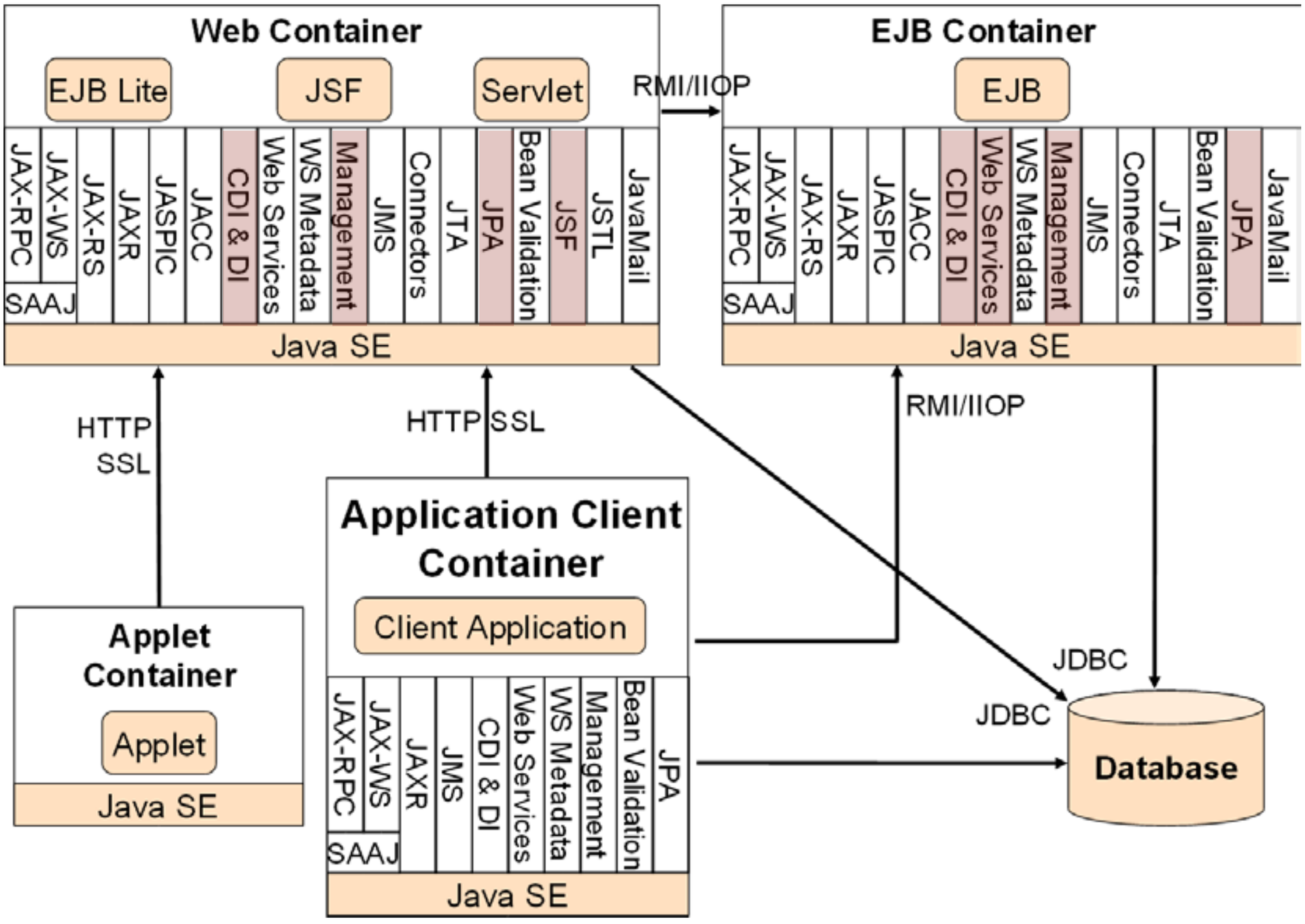
- Une plateforme normalisée
- Orientée composants distribués (réseaux)
- Conforme aux standards : Web (http), XML, SOAP
- Garantissant la portabilité : *Write once, run anywhere*
- Ouverte sur les environnement existants
  - SGBD, Annuaires LDAP, Web services, Intergiciel

Implementations (de référence - RI)

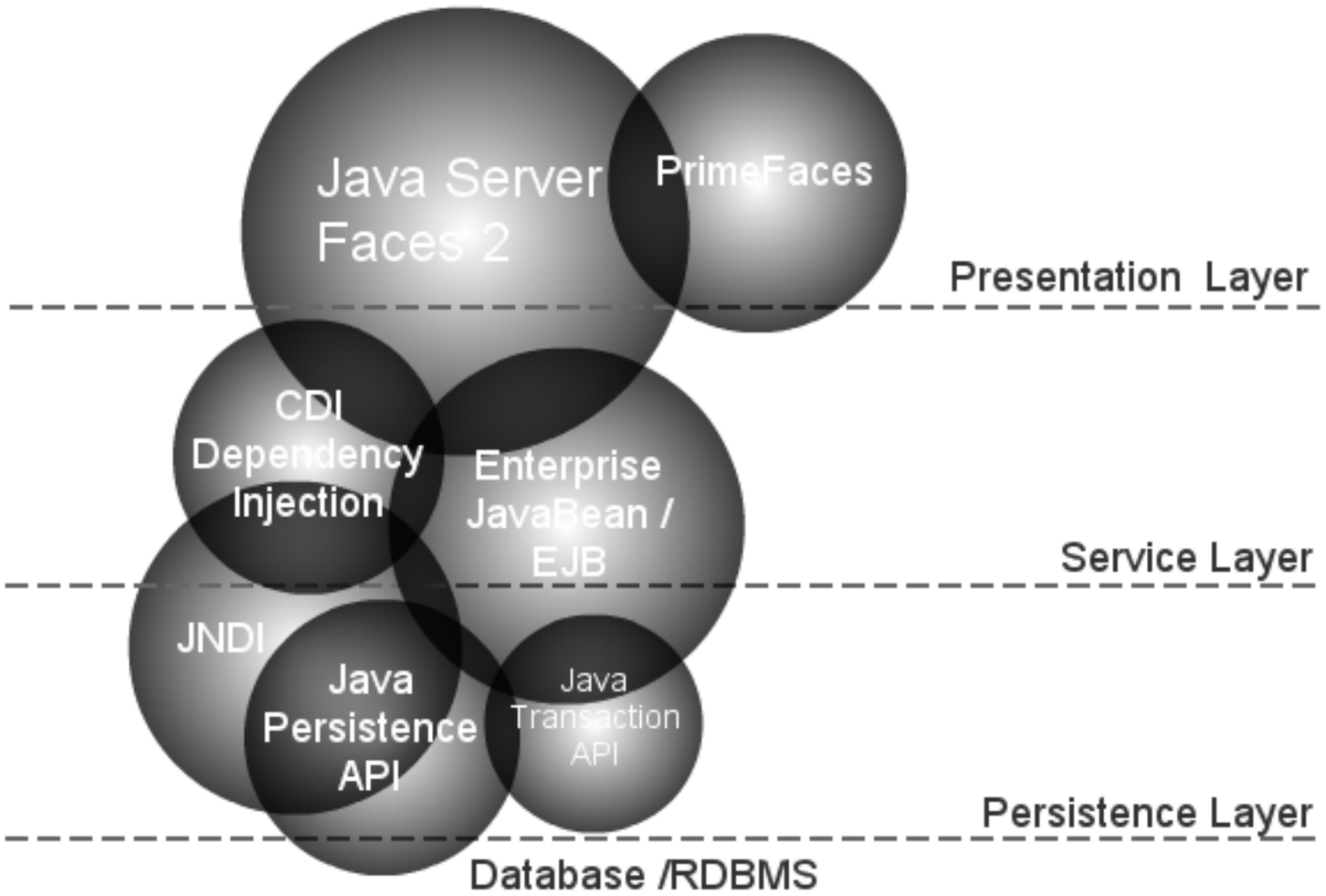
Architecture JEE



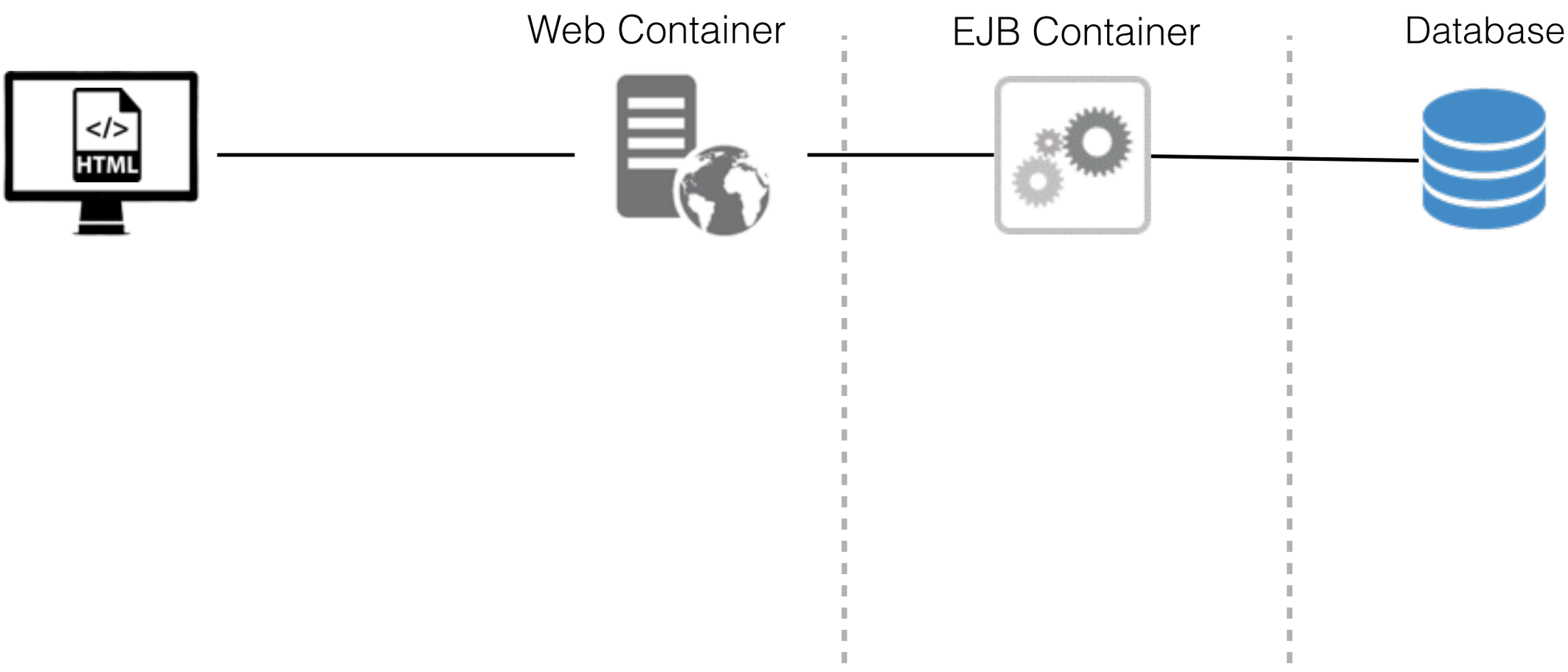
Services fournis par conteneur



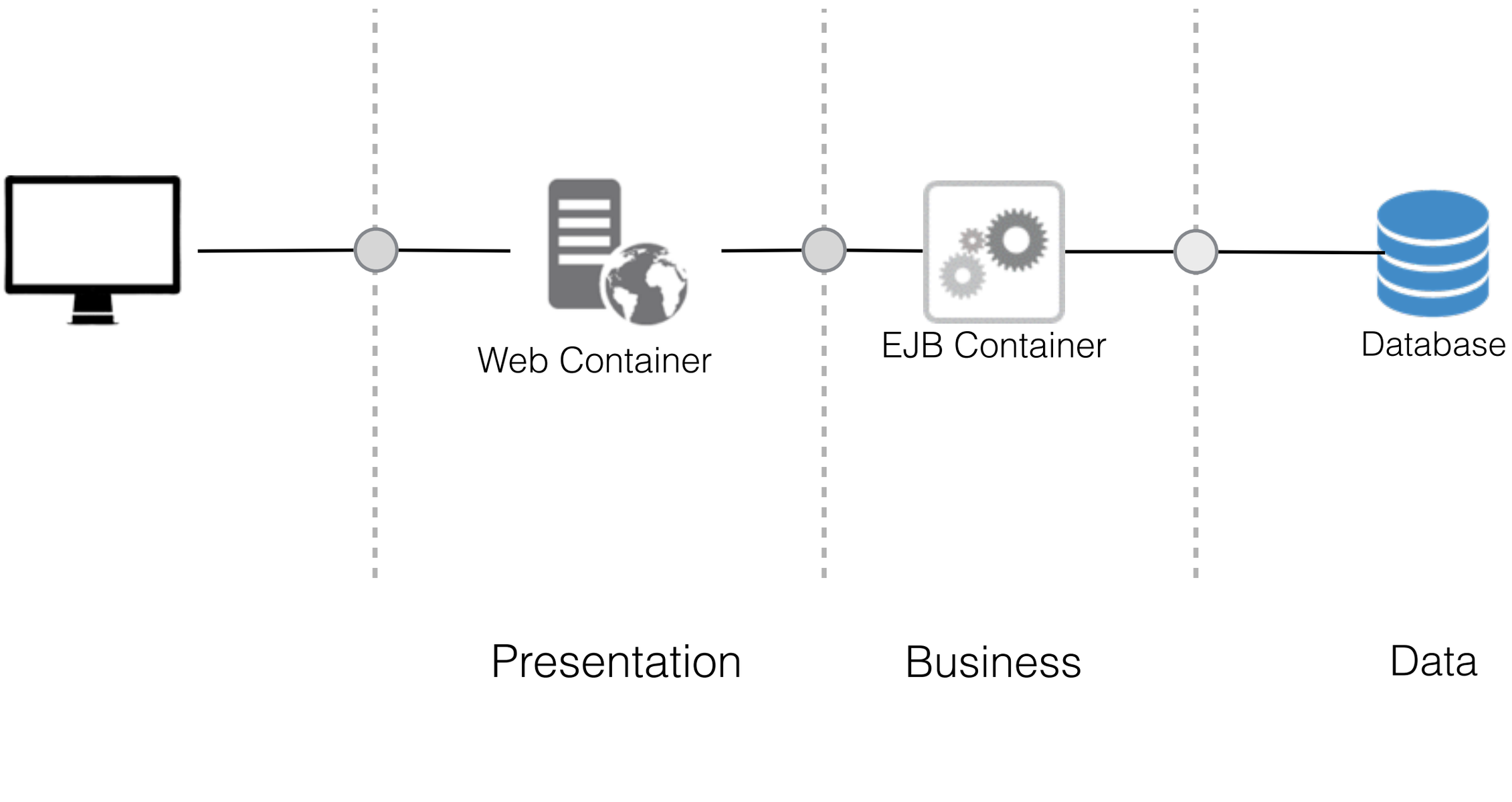
JEE pour l'architecture client-serveur



Architecture client-serveur multi-tier/couche



Architecture client-serveur multi-tier/couche



Modèle de programmation

- Plain Old Java Object (POJO) + Metadata

- EJB, JSF backing beans, SOAP, Rest services

  - Classes Java + annotations (and ou XML)

```
@Path("books")
public class BookResource {
  @Inject
  private EntityManager em;
  @GET
  @Produces({"application/xml",
             "application/json"})
  public List<Book> getAllBooks() {
    Query query =
        em.createNamedQuery("findAllBooks");
    List<Book> books = query.getResultList();
    return books;
  }
}
```

Stateless EJB

Restful Webservice

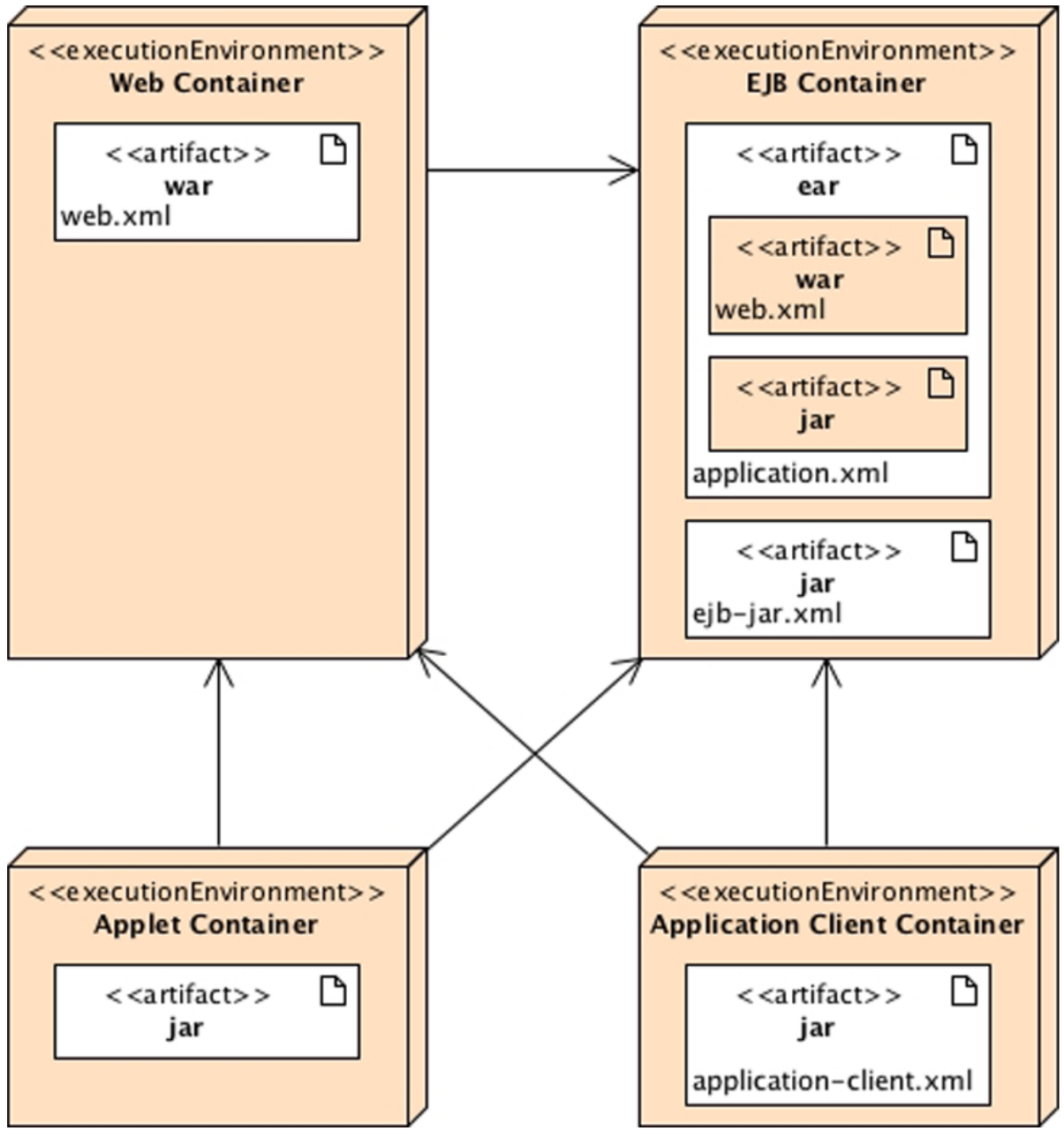
JSF backing bean

Annotations et descripteurs de déploiement

```java
@Stateless
@Remote(ItemRemote.class)
@Local(ItemLocal.class)
public class ItemEJB implements ItemLocal, ItemRemote {
  @PersistenceContext(unitName = "chapter01PU")
  private EntityManager em;

  public Book findBookById(Long id) {
    return em.find(Book.class, id);
  }
}
```

```xml
<ejb-jar ...>
  <enterprise-beans>
    <session>
      <ejb-name>ItemEJB</ejb-name>
      <remote>
         org.agoncal.book.javaee7.ItemRemote
      </remote>
      <local>
         org.agoncal.book.javaee7.ItemLocal
      </local>
      <ejb-class>
         org.agoncal.book.javaee7.ItemEJB
      </ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

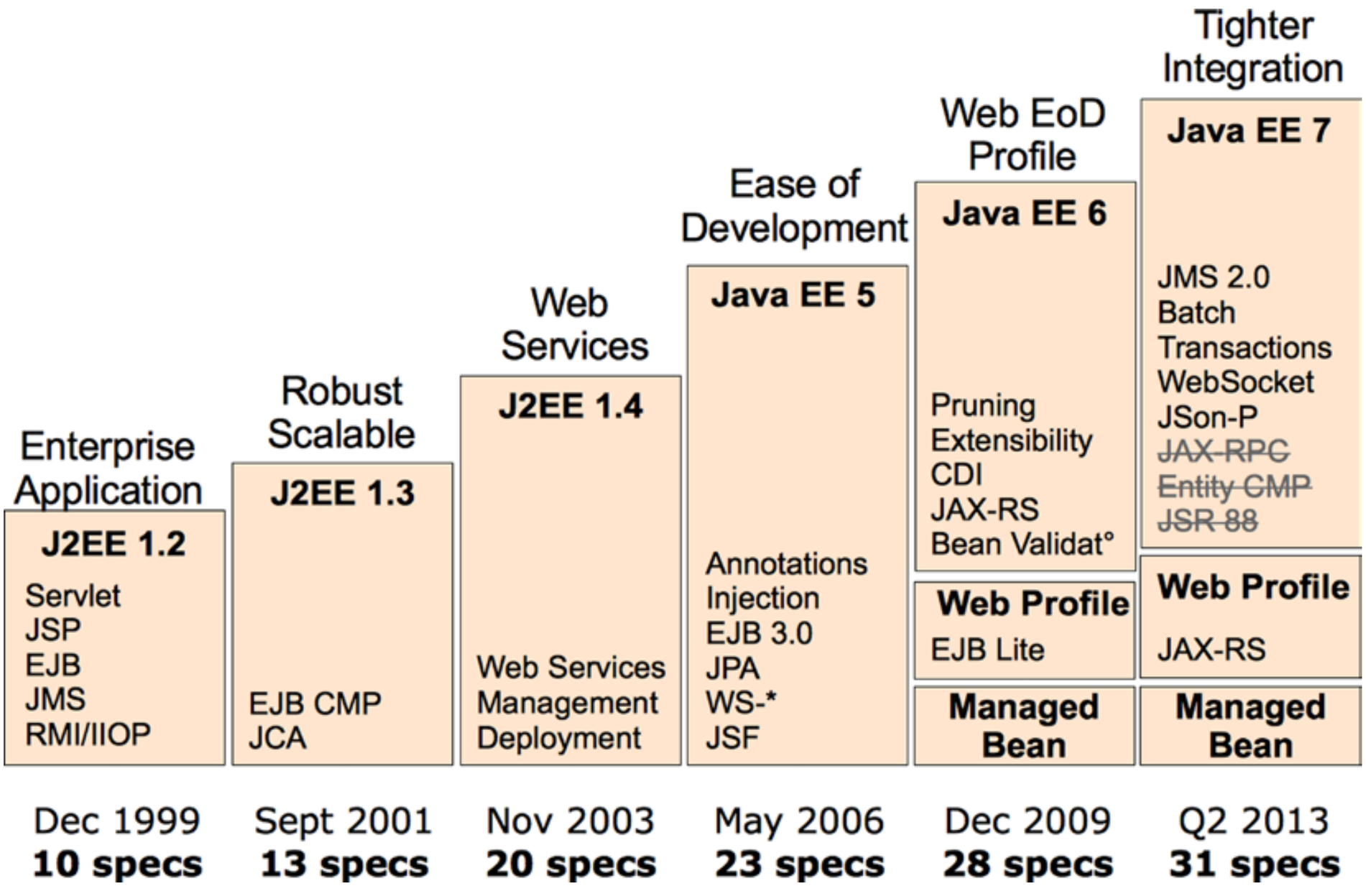
Packaging


<<executionEnvironment>>
Web Container
<<artifact>>
war
web.xml
<<executionEnvironment>>
EJB Container
<<artifact>>
ear
<<artifact>>
war
web.xml
<<artifact>>
jar
application.xml
<<artifact>>
jar
ejb-jar.xml
<<executionEnvironment>>
Applet Container
<<artifact>>
jar
<<executionEnvironment>>
Application Client Container
<<artifact>>
jar
application-client.xml

Annotations et fichiers descripteurs de déploiement

Fichier	Spécification	Chemin
application.xml	Java EE	META-INF
application-client.xml	Java EE	META-INF
beans.xml	CDI	META-INF or WEB-INF
ra.xml	JCA	META-INF or WEB-INF
ejb-jar.xml	EJB	META-INF
faces-config.xml	JSF	META-INF or WEB-INF
persistence.xml	JPA	WEB-INF
validation.xml	Bean Validation	META-INF or WEB-INF
web.xml	Servlet	META-INF
web-fragment.xml	SOAP Web Services	META-INF
webservices.xml	Java EE	META-INF or WEB-INF

Evolution


Enterprise Application
J2EE 1.2
Servlet
JSP
EJB
JMS
RMI/IIOP
Dec 1999
10 specs
Robust Scalable
J2EE 1.3
EJB CMP
JCA
Sept 2001
13 specs
Web Services
J2EE 1.4
Web Services
Management
Deployment
Nov 2003
20 specs
Ease of Development
Java EE 5
Annotations
Injection
EJB 3.0
JPA
WS-*
JSF
May 2006
23 specs
Web EoD Profile
Java EE 6
Pruning
Extensibility
CDI
JAX-RS
Bean Validat°
Web Profile
EJB Lite
Managed Bean
Dec 2009
28 specs
Tighter Integration
Java EE 7
JMS 2.0
Batch
Transactions
WebSocket
JSon-P
~~JAX-RPC~~
~~Entity CMP~~
~~JSR 88~~
Web Profile
JAX-RS
Managed Bean
Q2 2013
31 specs

Ressources

- Livre référence
  - Beginning Java EE 7 - Antônio Gonçalves

- Documentation
  - http://docs.oracle.com/javaee/7/
- Dépôt du module (tutoriels et code sources)
  - https://github.com/fredericoalvares/jee-emn

Plan

- Intro JEE
- Tutoriel 1 : Environnement de développement
- Java Persistance API
- Tutoriel 2
- Pause
- Enterprise JavaBeans
- Tutoriel 3
- JavaServer Faces
- Tutoriel 4

Tutoriel 1

Environnement de développement

Java Persistence API (JPA) et Mapping Objet-Relationnel

Mapping Objet-relationnel

- Vocabulaire des données persistantes / bases de données
  - Tables, clés primaires/étrangères, contraintes d'intégrité, joins, etc.

Mapping

- Vocabulaire en PPO / Java
  - Objets (instances de classes), references à d'autres objets, héritage, classes concrètes, abstraites, interfaces, etc.
  - Objets résident dans la mémoire vive
    - Effacé de temps en temps par le *garbage collector*

Entités et objets

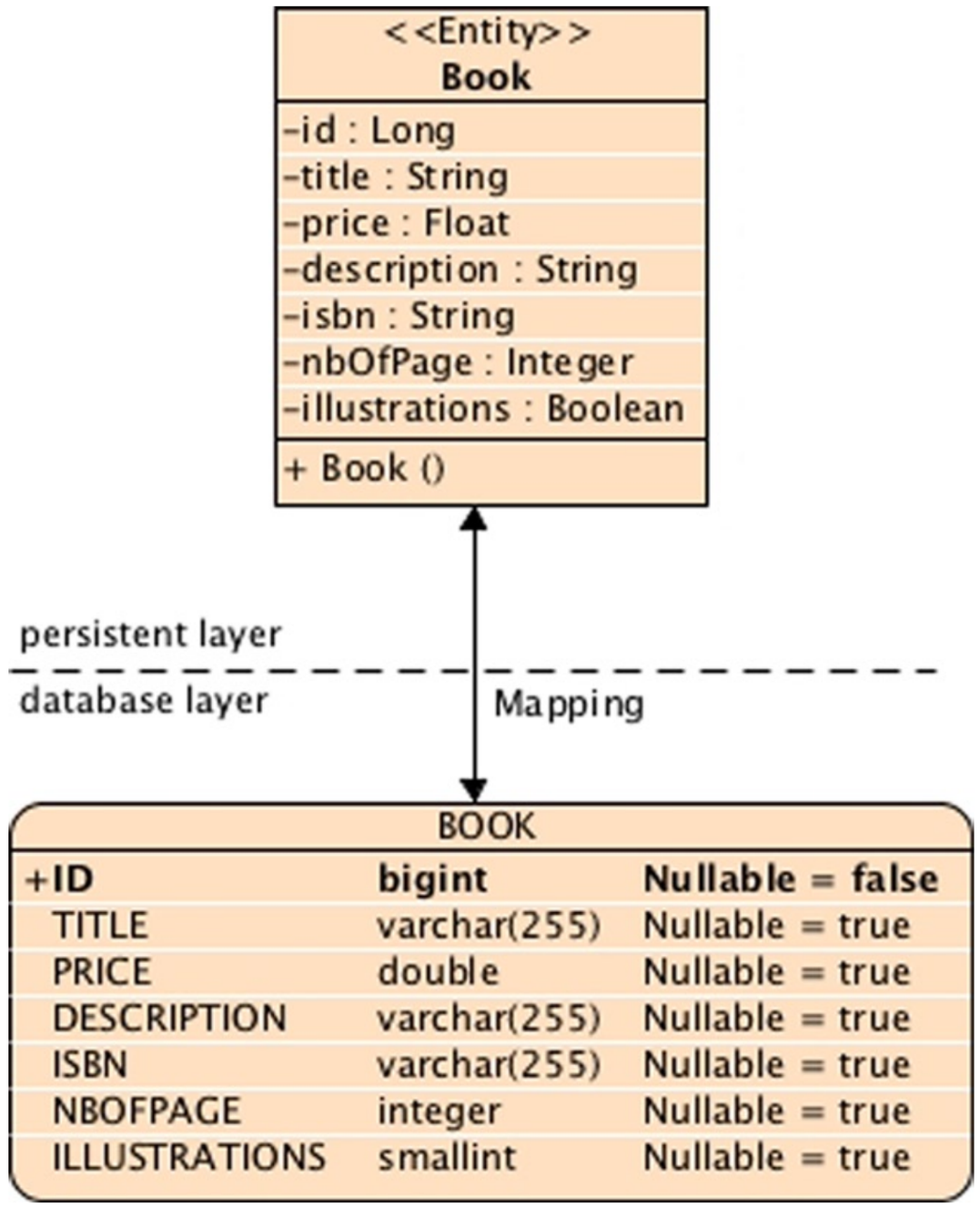
- Objets : vivent seulement dans le mémoire
- Entités
  - Objets qui vivent dans la mémoire pour une courte durée et en permanence dans la BD
  - Peuvent être m*appé* vers une base de donnée
    - Gérés par JPA -> on manipule des entités mais derrière il y a des accès à une base de données

Mapping

```java
@Entity
public class Book {
  @Id @GeneratedValue
  private Long id;
  private String title;
  private Float price;
  private String description;
  private String isbn;
  private Integer nbOfPage;
  private Boolean illustrations;

  public Book() {
  }
  // Getters, setters
}
```

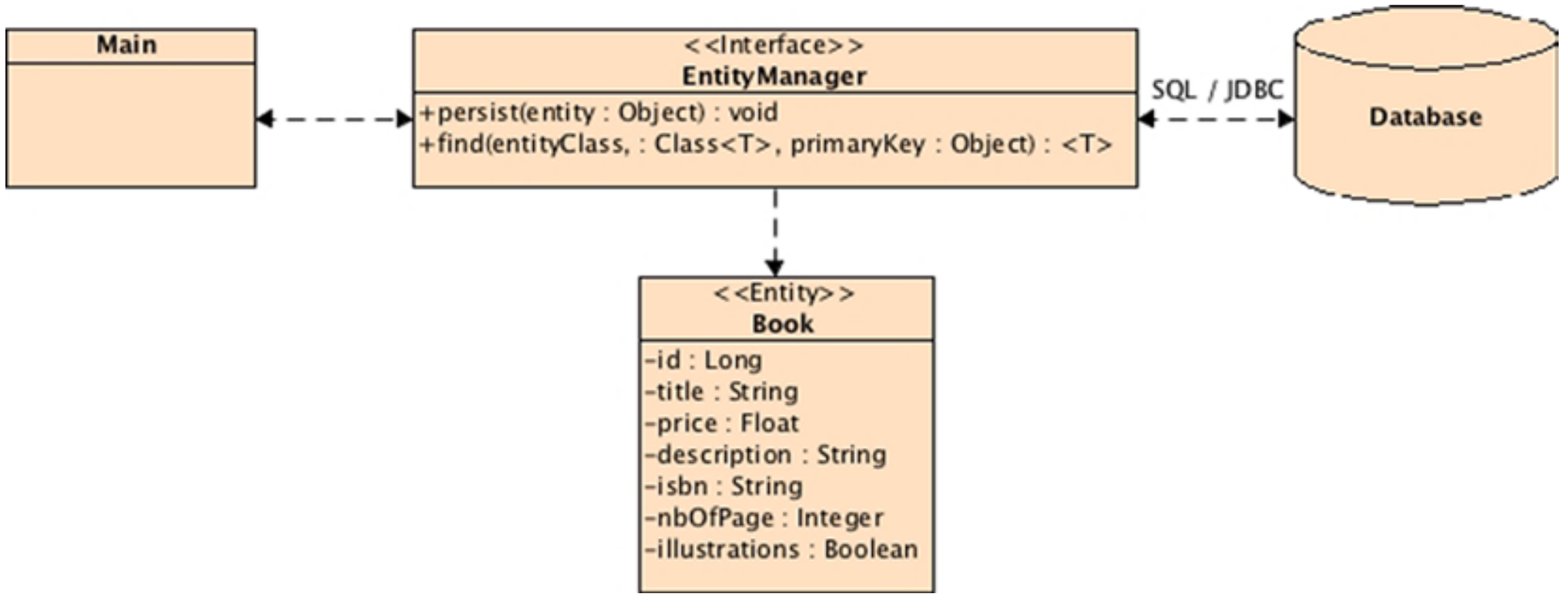
- Annotation ou description dans un fichier descripteur XML
- @Id indique la clé primaire
- Constructeur public sans argument
- La classe ni ses méthodes ne peut pas être final
- @Table pour préciser une table avec un nom différent
- @Column pour préciser une colonne avec un nom différent

Synchronisation entité - table



Interroger les entités

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("bookstore");
EntityManager em = emf.createEntityManager();
em.persist(book);
```



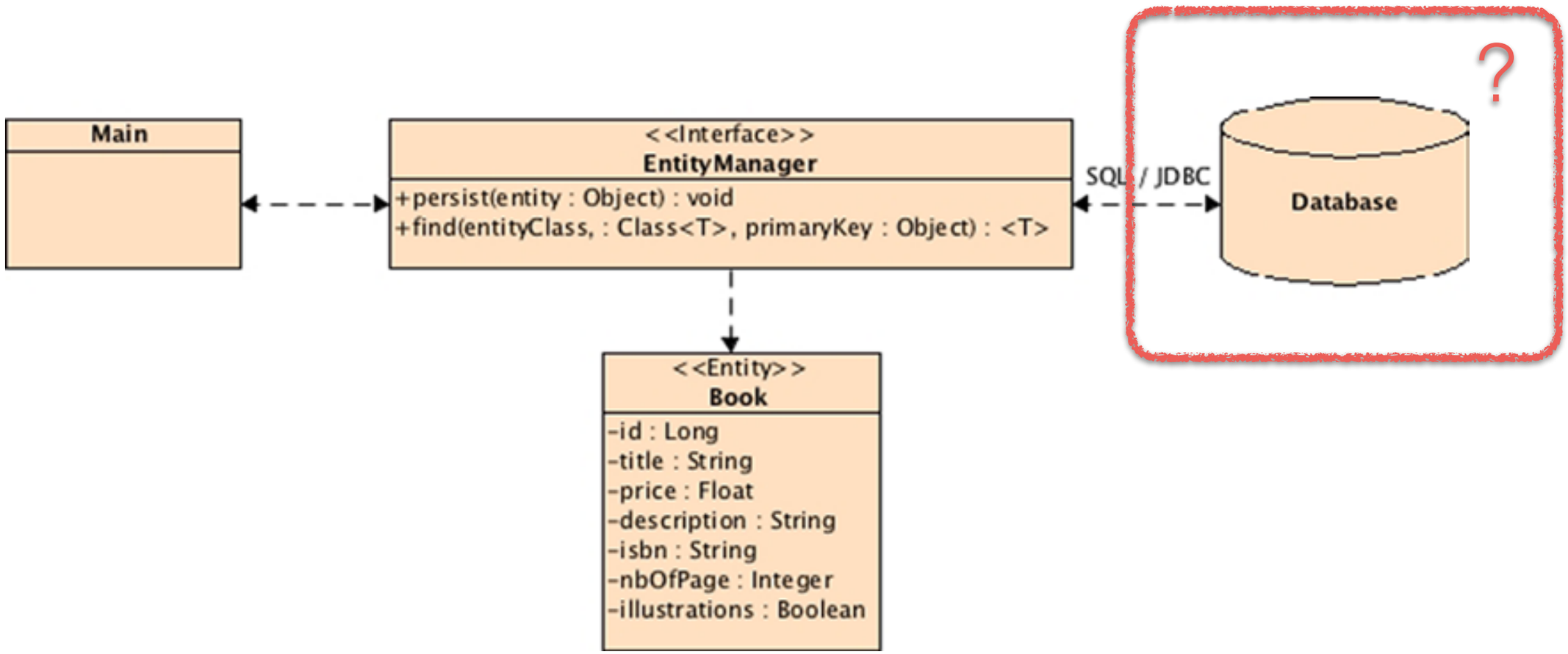
Requêtes SQL nommées

```
@Entity
@NamedQuery(name = "findBookH2G2", query = "SELECT b FROM Book b WHERE
b.title ='H2G2'")
public class Book {
...
}
```

Main.java

```
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("bookstore");
EntityManager em = emf.createEntityManager();

book = em.createNamedQuery("findBookH2G2",
                           Book.class).getSingleResult();
```

Comment JPA sait où se trouve la BD?



Unité de persistence

descripteur persistence.xml à placer dans :
src/main/resources/META-INF

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
                        http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
         version="2.1">
    <persistence-unit name="bookstore-hsqldb">
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
        <class>fr.emn.gsi2015.bookstore.persistence.Book</class>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="drop-
and-create" />
            <property name="javax.persistence.jdbc.driver" value="org.hsqldb.jdbcDriver" />
            <property name="javax.persistence.jdbc.url" value="jdbc:hsqldb:hsql://localhost:
9001/bookStoreDB;shutdown=true" />
            <property name="javax.persistence.jdbc.user" value="SA" />
            <property name="javax.persistence.jdbc.password" value="" />
            <property name="javax.persistence.sql-load-script-source" value="insert.sql" />
        </properties>
    </persistence-unit>
</persistence>
```

Tutoriel 2

Java Persistence API

Enterprise Java Beans - EJB

Enterprise Java Beans

c’est quoi?

- Composant côté serveur qui encapsule la logique métier de l’application
- Simplifie le développement
  - Services fournis par le conteneur
  - Composants portables -> Reuse

Services d'un conteneur

- Communication distante
- Injection de dépendances
- Gestion de l'état
- *Pooling*
- Cycle de vie
- Messages
- Gestion de transaction
- Intercepteurs transversaux
- Sécurité
- Gestion de la concurrence
- Appels de méthodes asynchrones
- *Pooling*

Types de EJB

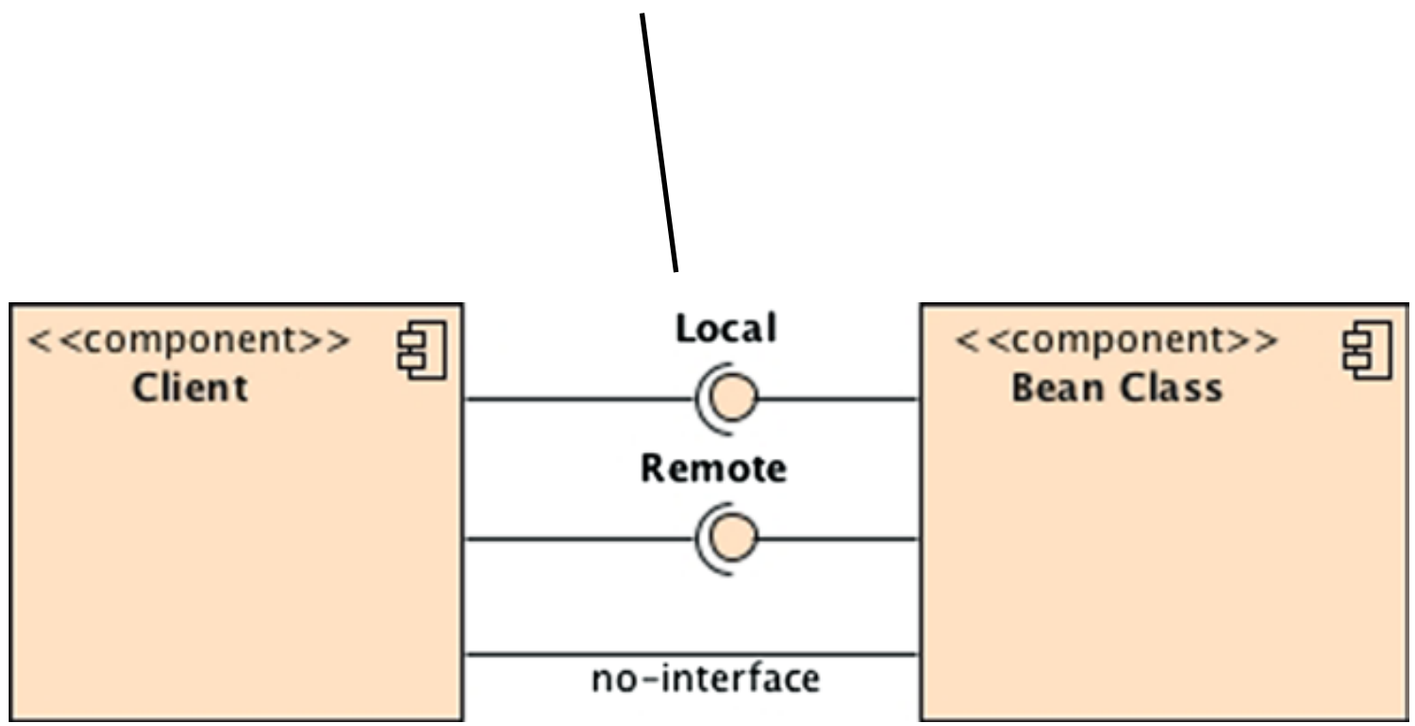
- Session beans
  - Accomplit une tâche pour un client
- Message-Driven Beans :
  - un écouteur pour un type de messagerie particulière (ex. : comme l'API Java Message Service)

Types de Bean de Session

- Stateless : aucun état conventionnel entre les méthodes
- Stateful : contient l'état conventionnel qui doit être mémorisé pour un client donné
- Singleton : une seule instance pour tous les clients

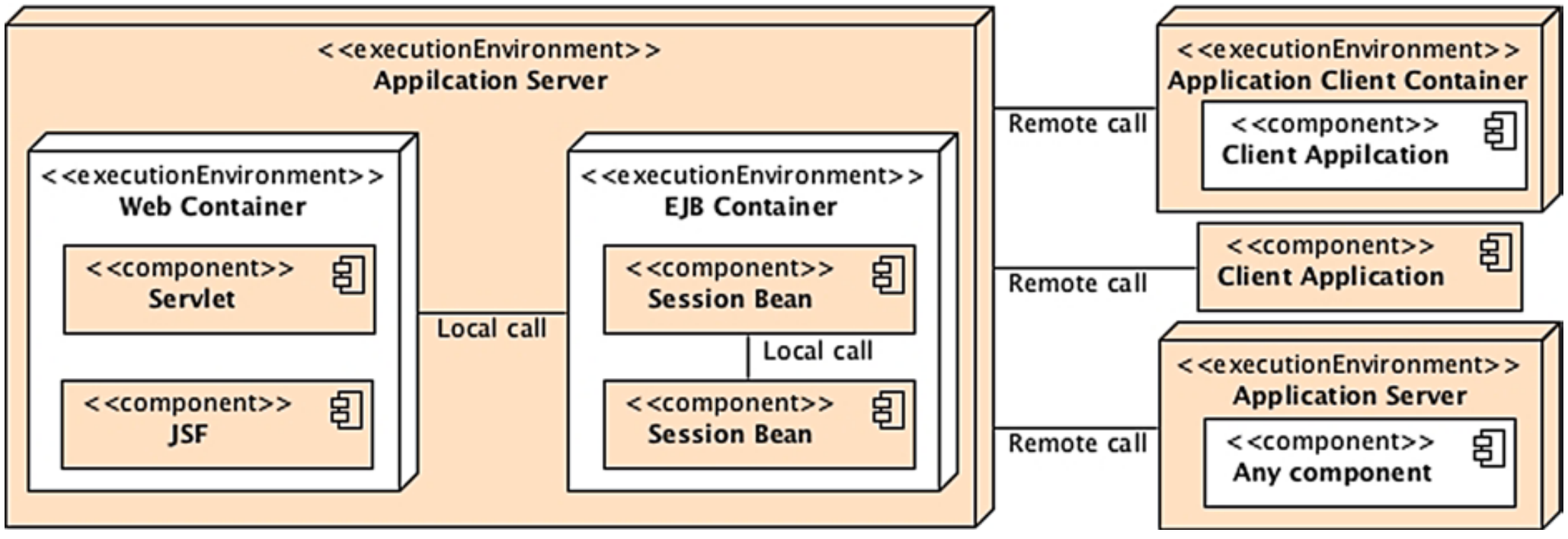
Anatomie d'un bean

Interface métier



Implémentation des méthodes métiers

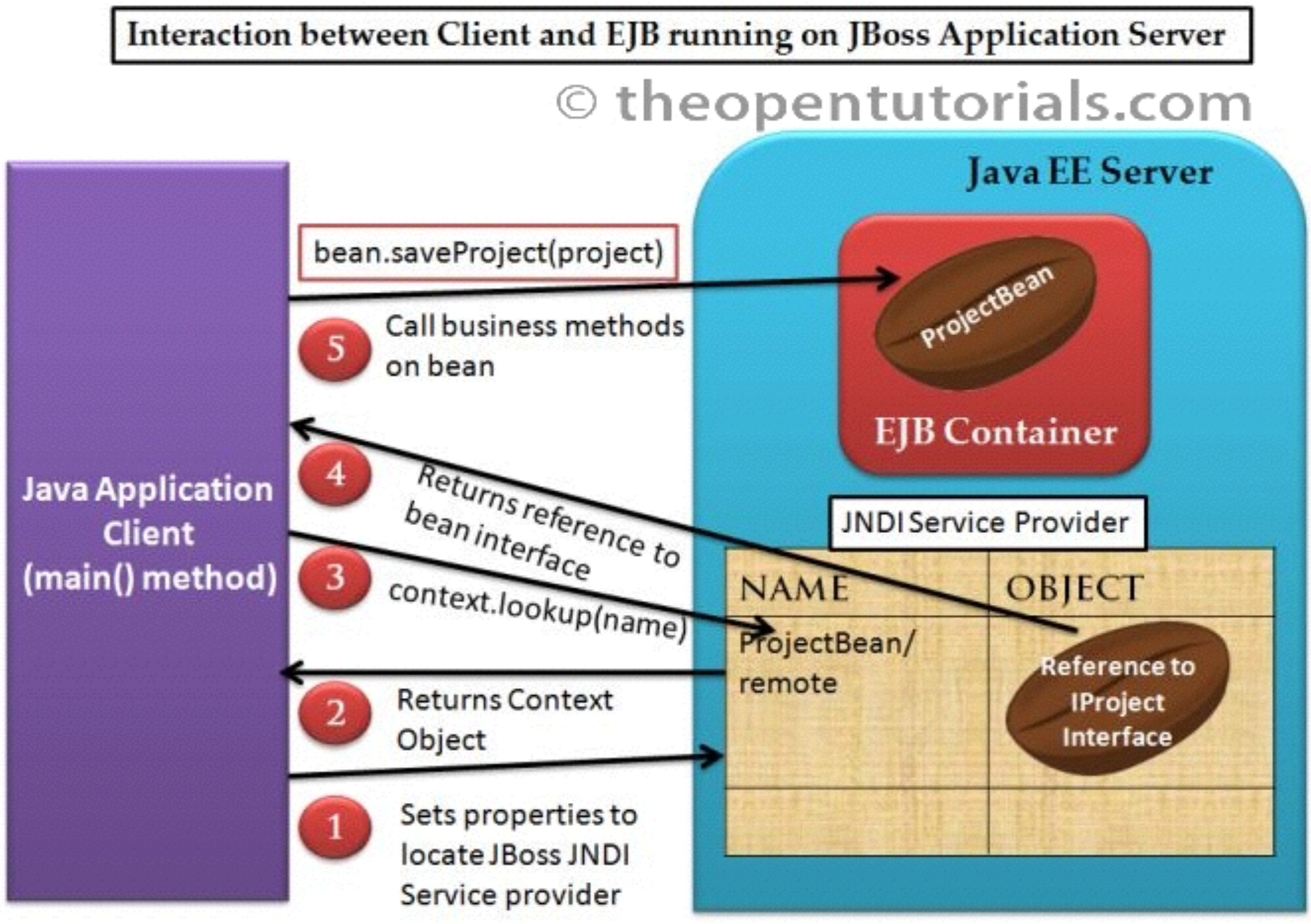
Interface Locale / Distante



Comment trouver un bean?

- JNDI : Java Naming and Directory Interface
  - Une API pour un service d'annuaire permettant un client de trouver des données et des objets

java:<scope>[/<app-name>]/<module-name>/<bean-name>[!<fully-qualified-interface-name>]

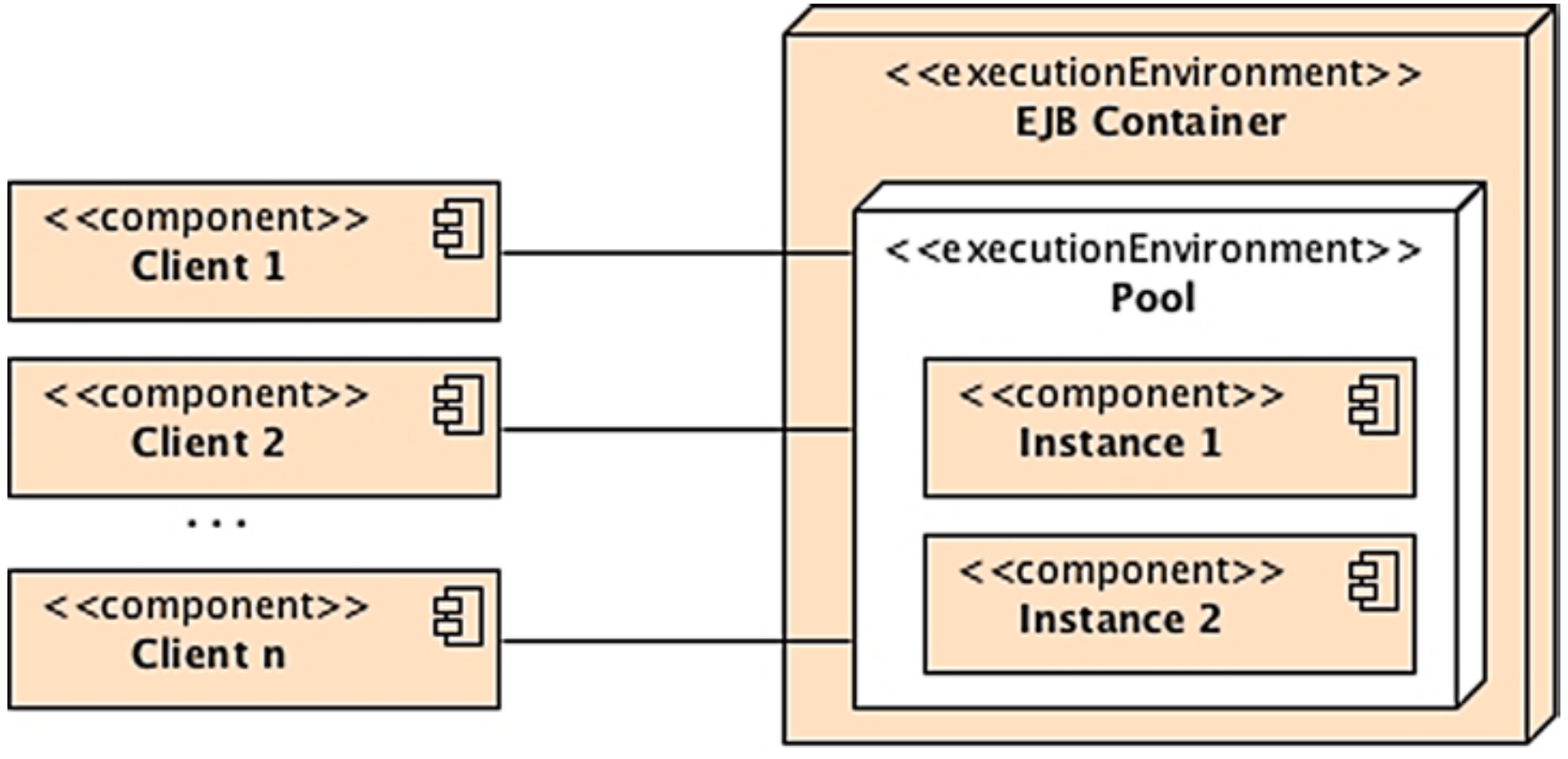

Interaction between Client and EJB running on JBoss Application Server
© theopentutorials.com
Java EE Server
Java Application Client (main() method)
bean.saveProject(project)
5
Call business methods on bean
4
Returns reference to bean interface
3
context.lookup(name)
2
Returns Context Object
1
Sets properties to locate JBoss JNDI Service provider
ProjectBean
EJB Container
JNDI Service Provider
NAME
OBJECT
ProjectBean/ remote
Reference to IProject Interface

Interface Local / Distante

```
@Local
public interface BookLocalService {
  List<Book> findBooks();
  Book createBook(Book book);
}
@Remote
public interface BookRemoteService {
  List<Book> findBooks();
}
@Stateless
public class ItemEJB implements BookLocalService,
BookRemoteService {
// ...
}
```

Bean de session *stateless*

- Chaque requête est attribué à une instance du pool
- Une fois la requête traité, l'instance retourne à au pool

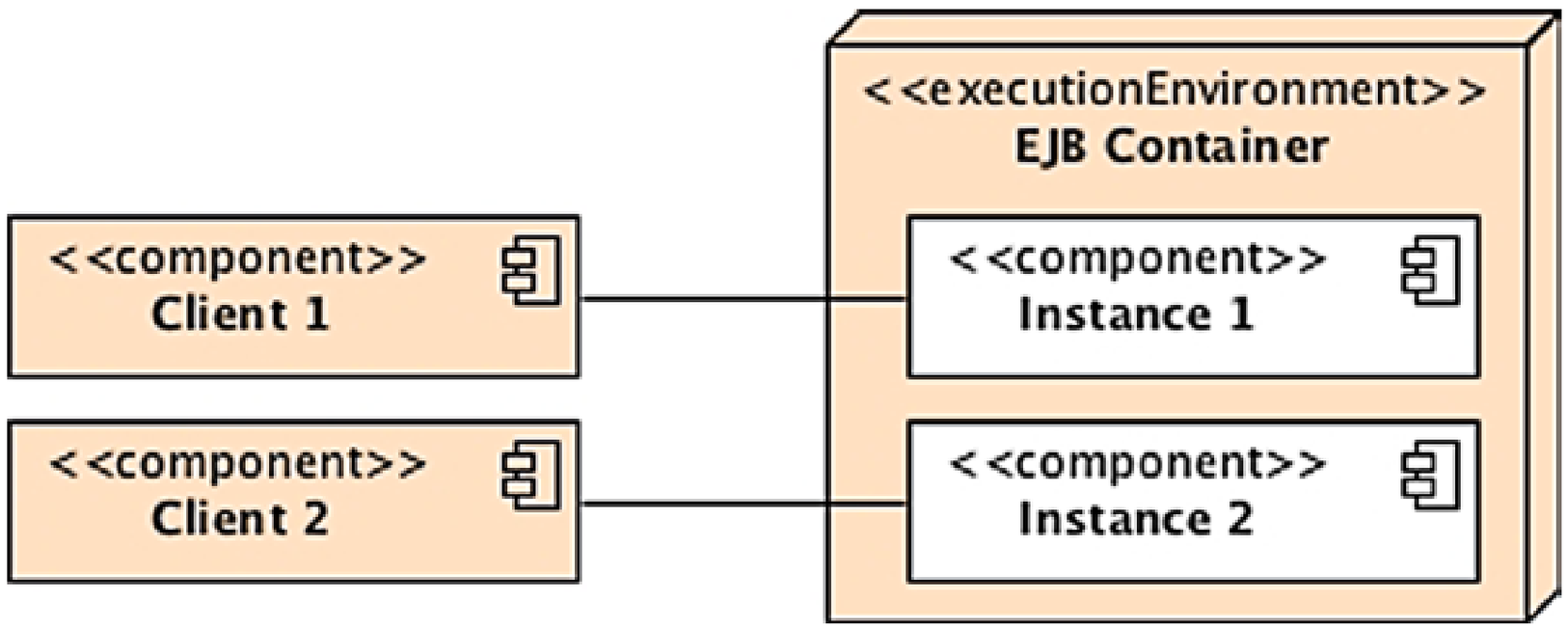


Exemple

```java
@Stateless
public class BookEJB implements BookLocalService {

    @PersistenceContext(unitName = "bookstore-hsqldb")
    private EntityManager em;

    public List<Book> findBooks() {
        TypedQuery<Book> query = em.createNamedQuery("findAllBooks", Book.class);
        return query.getResultList();
    }


    public Book findBookById(Long id) {
        return em.find(Book.class, id);
    }

    public Book createBook(Book book) {
        em.persist(book);
        return book;
    }
}
```

Bean de session *stateful*

- Une instance par client, une fois la session terminé, l'objet est détruit
- Gestion de la mémoire vive
  - Passivation : mémoire -> disc dur
  - Activation : disc dur -> mémoire

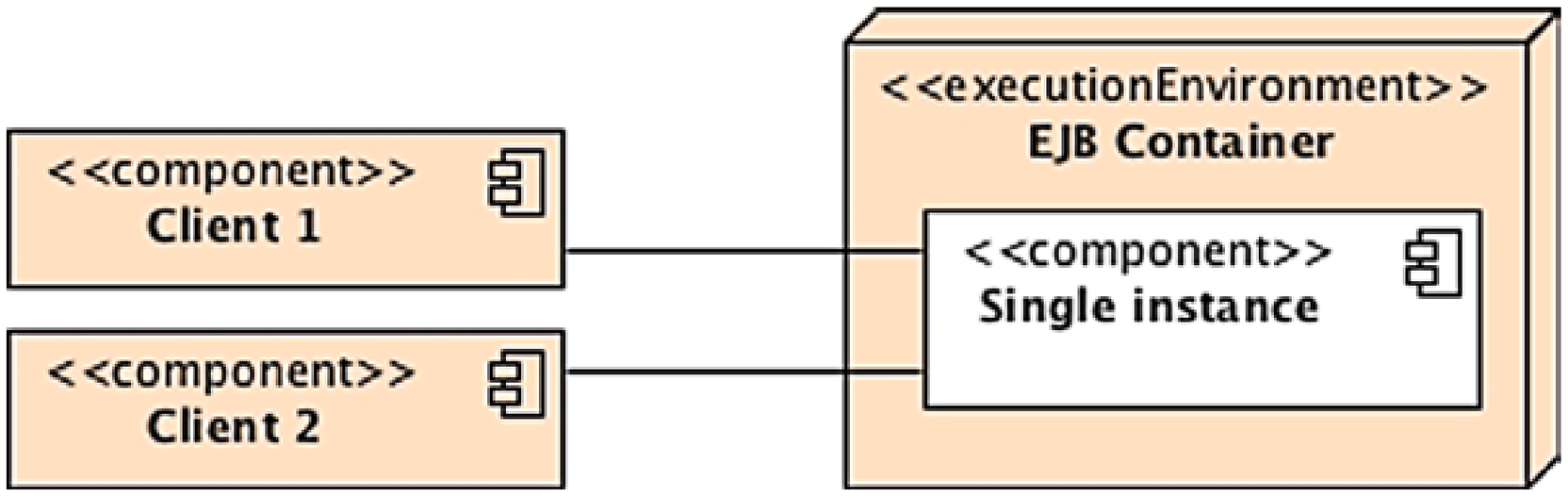


Exemple

```java
@Stateful
@StatefulTimeout(value = 30, unit = TimeUnit.SECONDS)
public class ShoppingCartEJB implements ShoppingCartService {

    private List<Book> cartItems = new ArrayList<Book>();

    public void addItem(Book item) {
        if (!cartItems.contains(item))
            cartItems.add(item);
    }

    public List<Book> getItems() {
        return cartItems;
    }

    public void removeItem(Book item) { ... }

    public Integer getNumberOfItems() { ... }

    public Float getTotal() { ... }

    public void empty() { ... }

    @Remove
    public void checkout() {
        cartItems.clear();
    }
}
```

Beans de session *singleton*

- Une seule instance pour tous les clients toute la durée de vie de l'application



Exemple

```
@Singleton
public class LastViewsEJB implements LastViewsService {

    private List<Book> books = new LinkedList<Book>();

    public List<Book> getLastViewed() {
       return books;
    }

    public void addLastViewed(Book book) { ... }
}
```

Concurrence

- *Container-Managed Concurrency (CMC)*
  - Méta-données (annotations ou xml)
    - Accès contrôlé par le conteneur
- *Bean-Managed Concurrency (BMC)*
  - Gestion délégué au bean

Exemple de CMC
Verrou au niveau du bean

```
@Singleton
@Lock(LockType.WRITE)
@AccessTimeout(value = 20, unit = TimeUnit.SECONDS)
public class LastViewsEJB implements LastViewsService {

    private List<Book> books = new LinkedList<Book>();

    public List<Book> getLastViewed() {
       return books;
    }

    public void addLastViewed(Book book) { ... }
}
```

Exemple de CMC
Types de verrou

```
@Singleton
public class LastViewsEJB implements LastViewsService {

    private List<Book> books = new LinkedList<Book>();

    @Lock(LockType.READ)
    public List<Book> getLastViewed() {
       return books;
    }

    @Lock(LockType.WRITE)
    @AccessTimeout(value = 20, unit = TimeUnit.SECONDS)
    public void addLastViewed(Book book) { ... }
}
```

Exemple de BMC

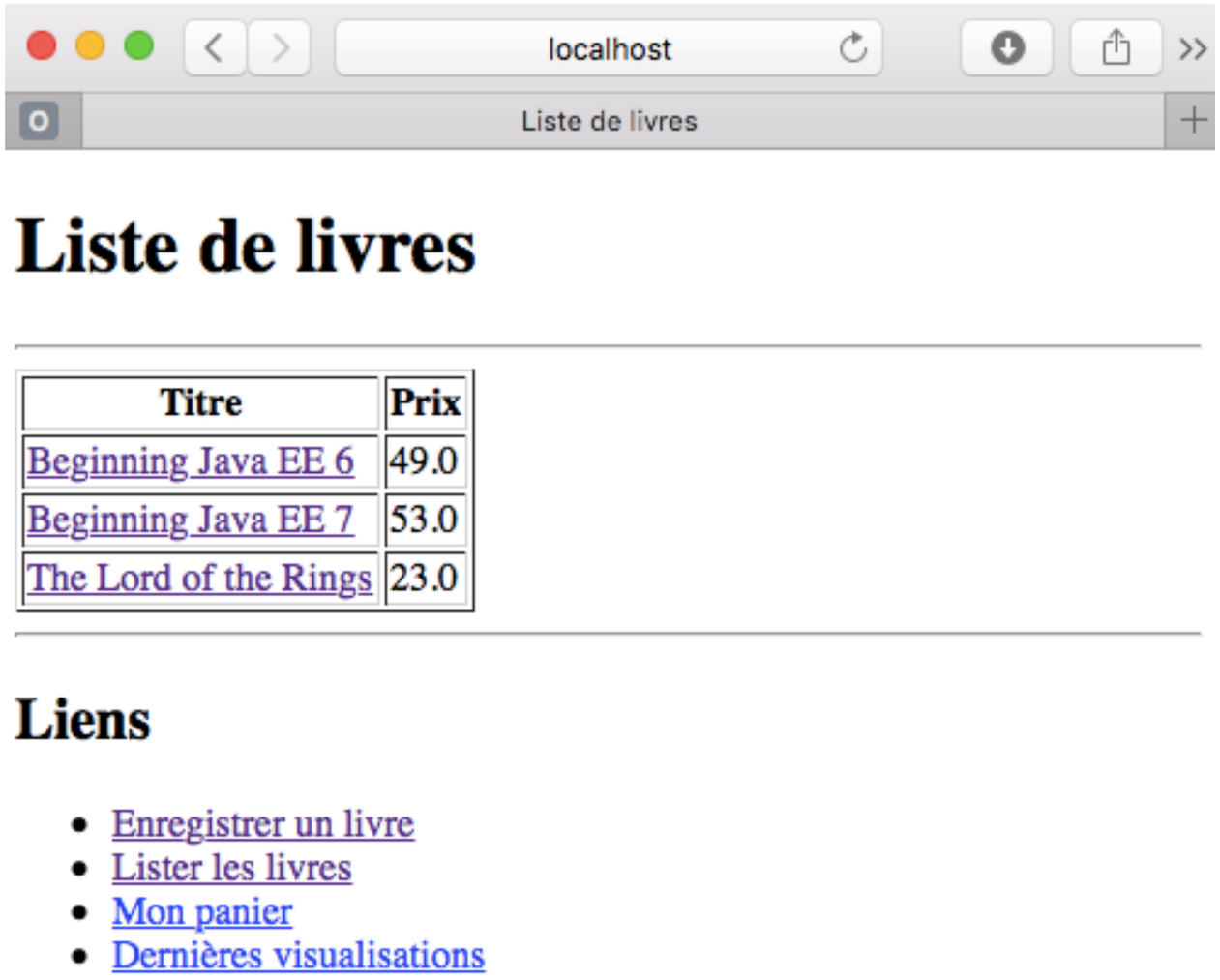
```
@Singleton
public class LastViewsEJB implements LastViewsService {

    private List<Book> books = new LinkedList<Book>();

    public List<Book> getLastViewed() {
        return books;
    }

    public synchronized void addLastViewed(Book book){
        ...
    }
}
```

Tutoriel 3

Enterprise Java Beans

Java Server Faces - JSF

Requête Page Web (HTTP)



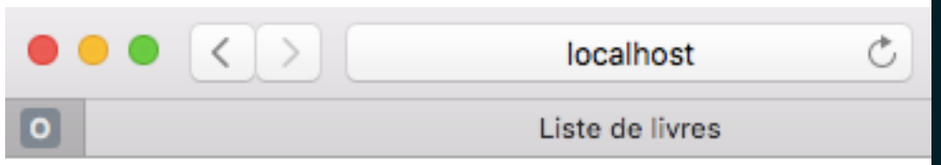
Accès
à une page

requête HTTP

réponse HTTP

HTML rendu

Requête Page Web (HTTP)



Liste de livres

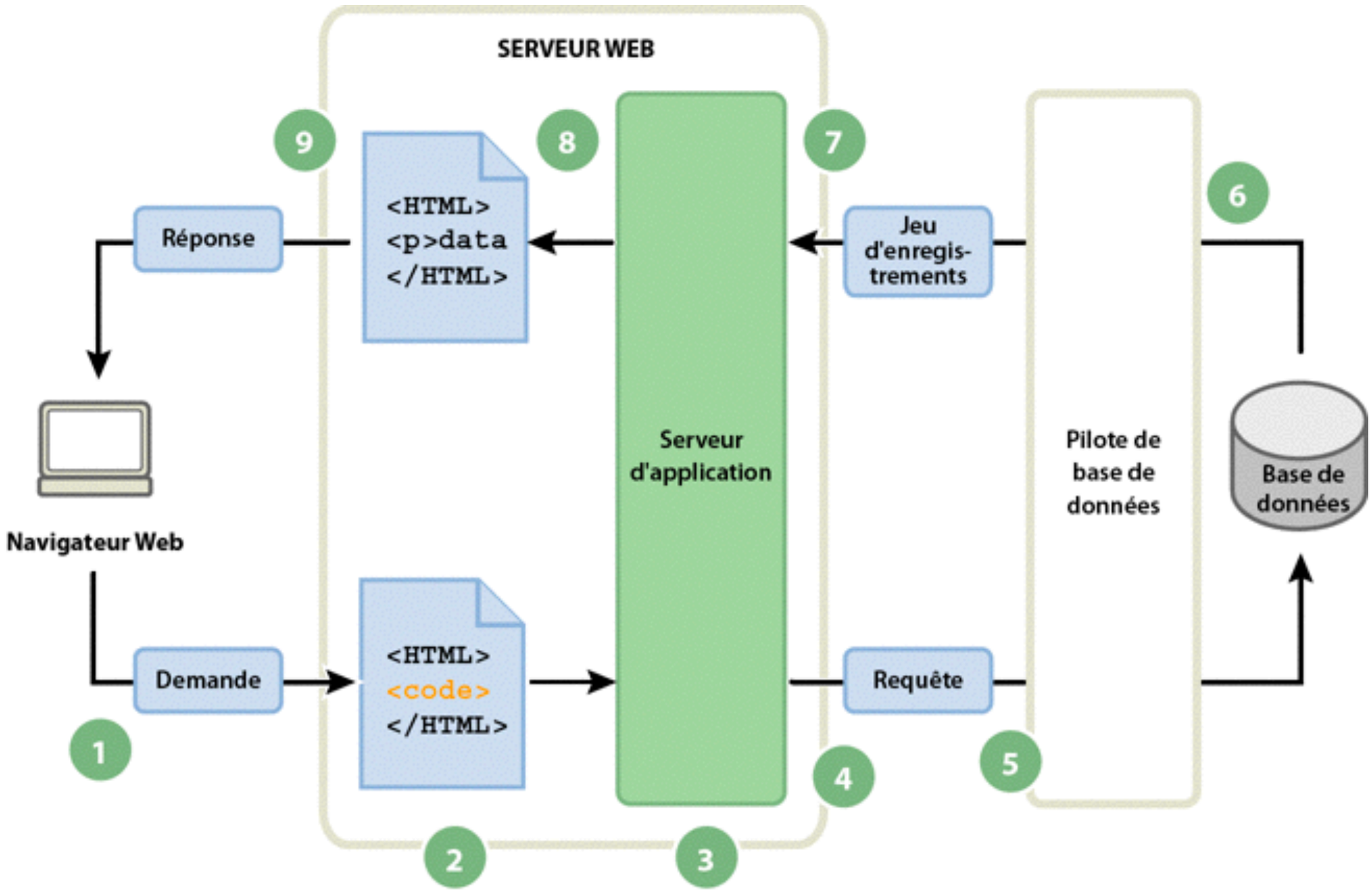
Titre	Prix
Beginning Java EE 6	49.0
Beginning Java EE 7	53.0
The Lord of the Rings	23.0

Liens

- Enregistrer un livre
- Lister les livres
- Mon panier
- Dernières visualisations

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Liste de livres</title></head><body>
</head>
<body>
 <h1>Liste de livres</h1>
 <hr />
 <table border="1">
  <tr>
    <th scope="col">Titre</th>
    <th scope="col">Prix</th>
  </tr>
  <tr>
    <td><a href="/bookstore/viewBook.faces?idBook=1000">Beginning Java EE 6</a></td>
    <td>49.0</td>
  </tr>
  <tr>
    <td><a href="/bookstore/viewBook.faces?idBook=1001">Beginning Java EE 7</a></td>
    <td>53.0</td>
  </tr>
  <tr>
    <td><a href="/bookstore/viewBook.faces?idBook=1010">The Lord of the Rings</a></td>
    <td>23.0</td>
  </tr>
 </table>
 <hr />
 <h2>Liens</h2>
 <ul>
  <li><a href="newBook.faces">Enregistrer un livre</a></li>
  <li><a href="listBooks.faces">Lister les livres</a></li>
  <li><a href="shoppingCart.faces">Mon panier</a><br /></li>
  <li><a href="listViews.faces">Dernières visualisations</a></li>
 </ul>
 </body>
</html>
```

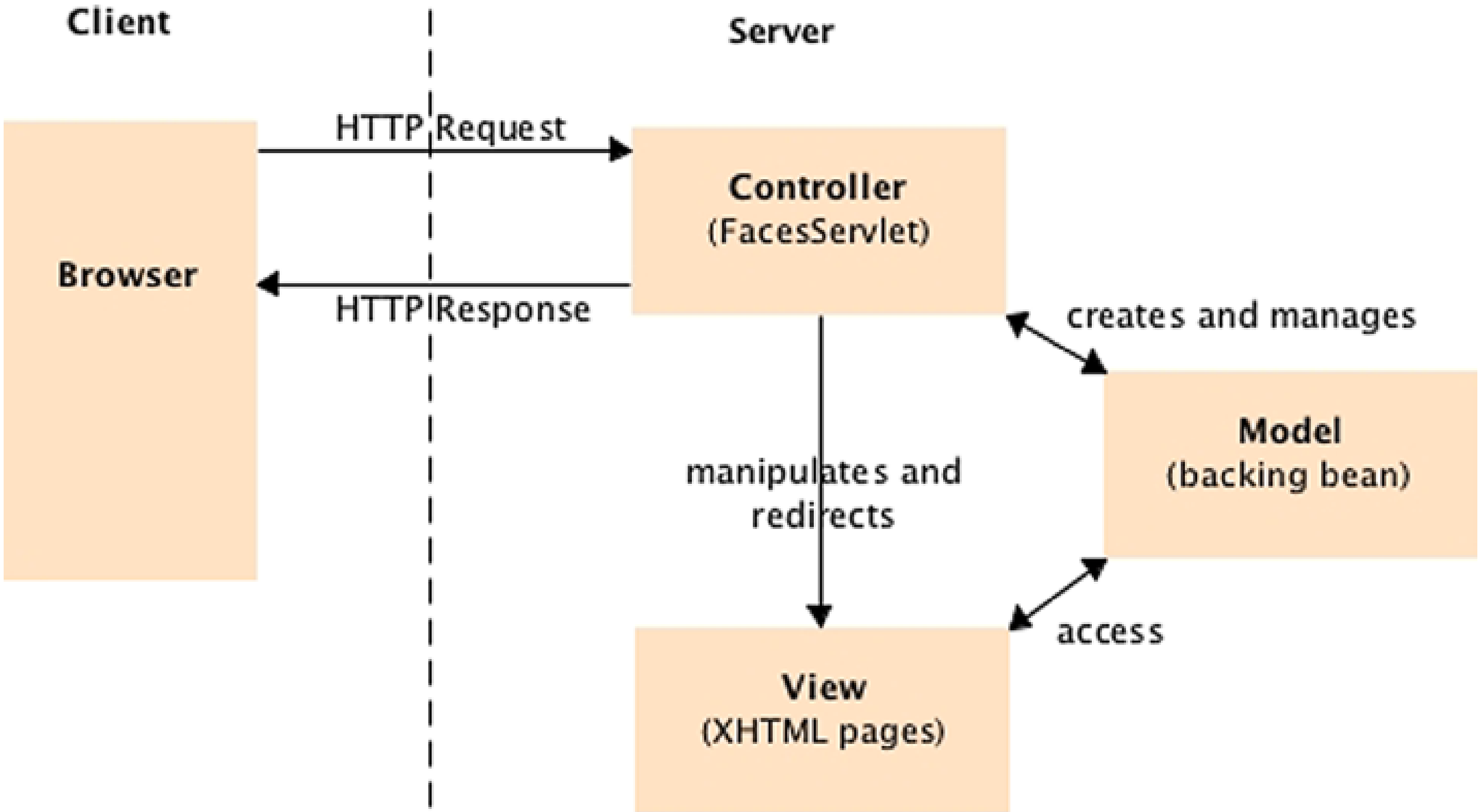
Requête Page Web (HTTP)



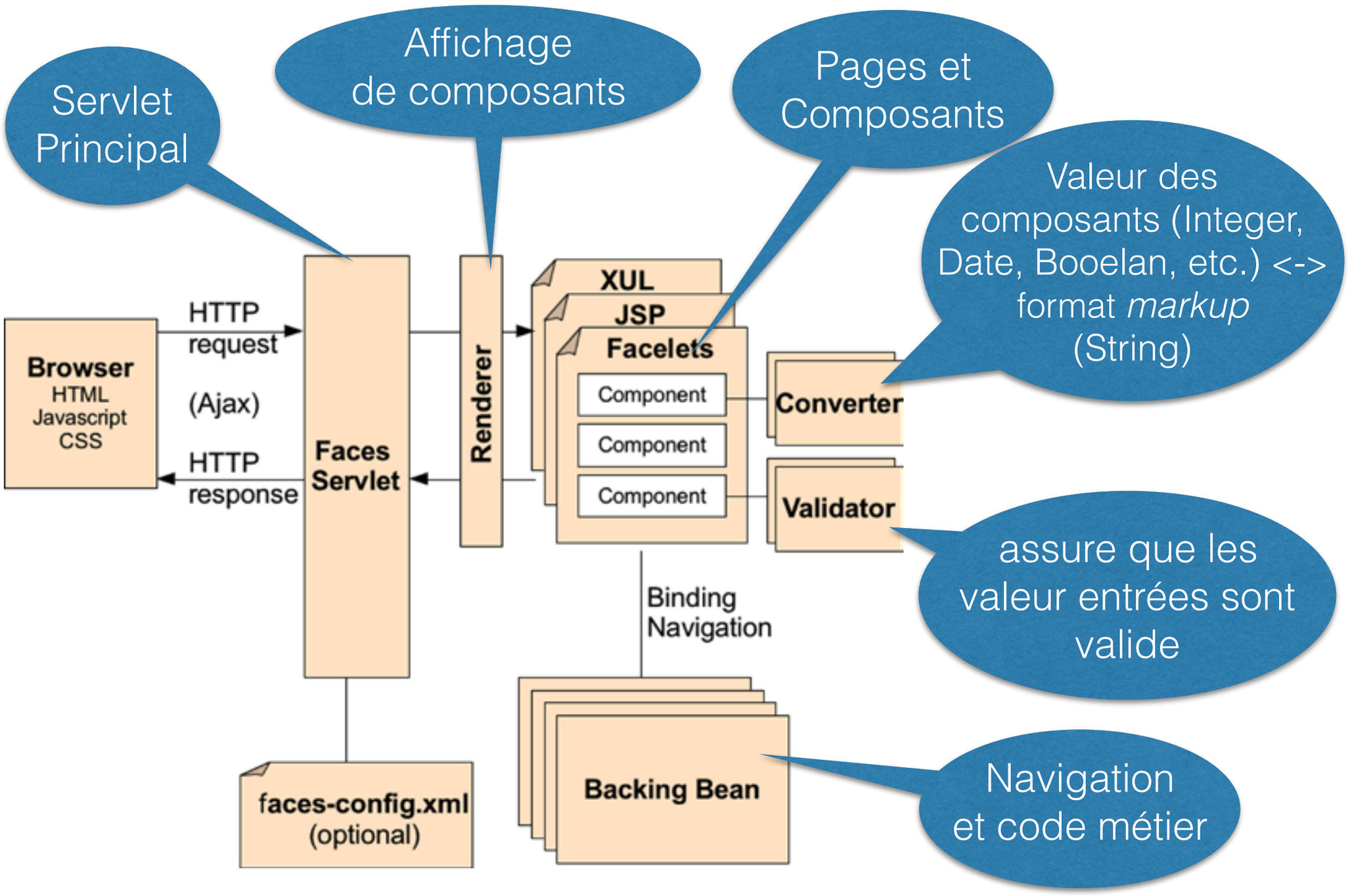
Java Server Pages - JSF

- Faciliter la conception des interfaces graphiques
- Inspirer par Swing et d'autres IHM frameworks
  - On pense en termes de components, events, *backing beans* et l'interaction entre eux
  - Au lieu de requêtes, réponses et balises

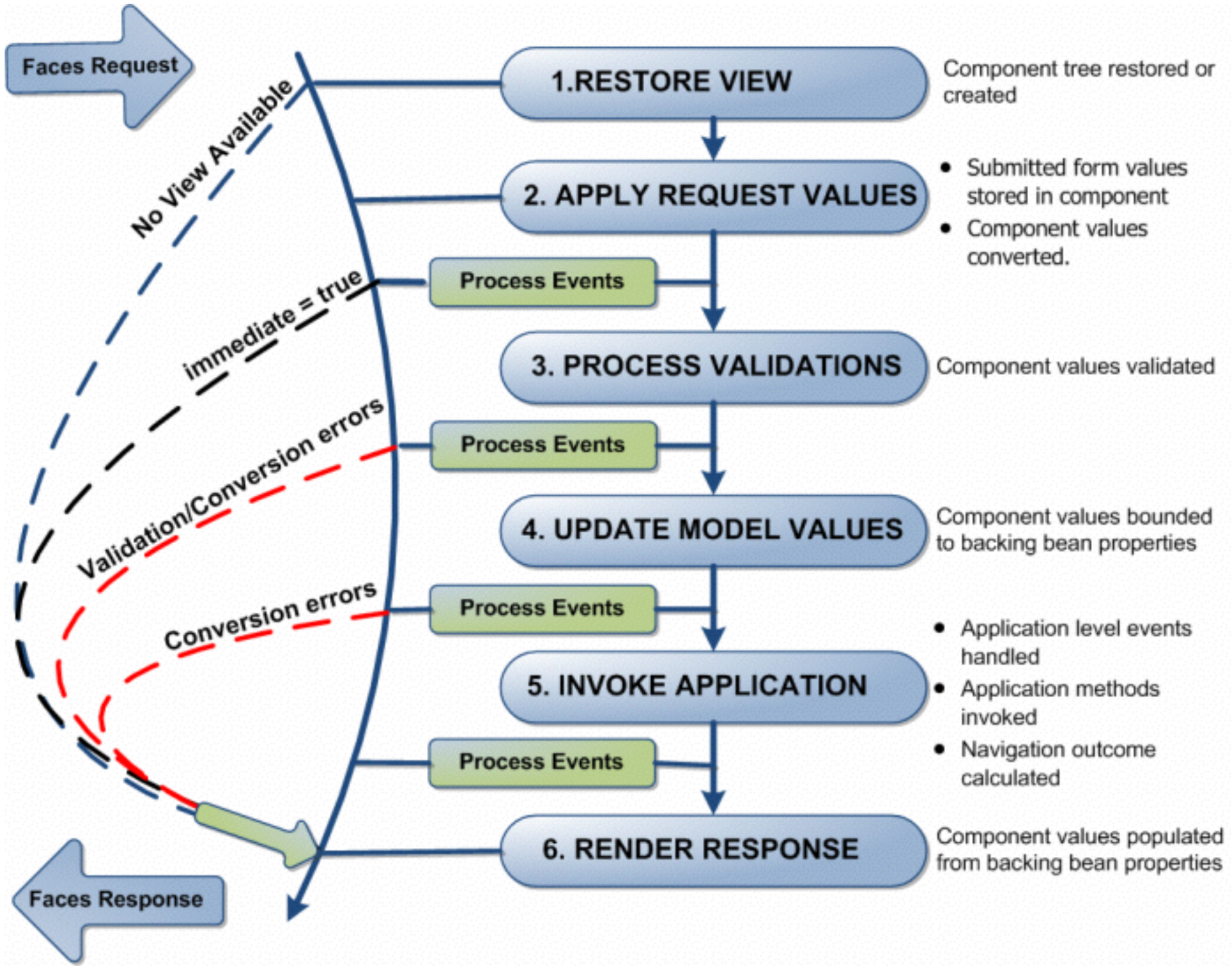
Model View Controller



JSF Architecture



Cycle de vie


Faces Request
No View Available
immediate = true
Validation/Conversion errors
Conversion errors
1.RESTORE VIEW
Component tree restored or created
2. APPLY REQUEST VALUES
Submitted form values stored in component
Component values converted.
Process Events
3. PROCESS VALIDATIONS
Component values validated
Process Events
4. UPDATE MODEL VALUES
Component values bounded to backing bean properties
Process Events
5. INVOKE APPLICATION
Application level events handled
Application methods invoked
Navigation outcome calculated
Process Events
6. RENDER RESPONSE
Component values populated from backing bean properties
Faces Response

Pages et Composants

Head

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN”
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Create a new book</title>
  </h:head>
  <h:body>
    ...
</h:body>
</html>
```

librairies de balises

Exemples de librairies de balises

- http://xmlns.jcp.org/jsf/html : composants html et leurs *renderers*
- http://xmlns.jcp.org/jsf/core : actions génériques (ex. : f:selectItem)
- http://xmlns.jcp.org/jsf/facelets : pages modèle

Pages et Composants

Body

```
<html ...>
  ...
  <h:body>
  <h1>Create a new book</h1>
  <hr/>
  <h:form>
    <table border="0">
    <tr>
      <td><h:outputLabel value="ISBN : "/></td>
      <td><h:inputText value="#{bookController.book.isbn}"/></td>
    </tr>
    <tr>
      <td><h:outputLabel value="Title :"/></td>
      <td><h:inputText value="#{bookController.book.title}"/></td>
    </tr>
    ...
    </table>
    <h:commandButton value="Create a book” action="#{bookController.doCreateBook}"/>
  </h:form>
</h:body>
</html>
```

Pages et *Backing Beans*

Page

```
<h:inputText
value="#{bookController.book.title}"/>

<h:commandButton value="Create a book"
action="#{bookController.doCreateBook}"/>
```

Backing Beans

```
@Named
@RequestScoped
public class BookController {
  @Inject
  private BookEJB bookEJB;
  private Book book = new Book();

  public String doCreateBook() {
     book = bookEJB.createBook(book);
     return "listBooks.xhtml";
  }
  // Getters, setters
}
```

Langage d'expression #{empty book.isbn}

navigation

Navigation

Ficher faces-config.xml

```
<navigation-rule>
  <from-view-id>*</from-view-id>
  <navigation-case>
     <from-outcome>
       doCreateBook-success
     </from-outcome>
     <to-view-id>
       /listBooks.xhtml
     </to-view-id>
  </navigation-case>
</navigation-rule>
```

Backing Bean

```
@Named
@RequestScoped
public class BookController {
  @Inject
  private BookEJB bookEJB;
  private Book book = new Book();

  public String doCreateBook() {
     book = bookEJB.createBook(book);
     return “doCreateBook-success";
  }
  // Getters, setters
}
```

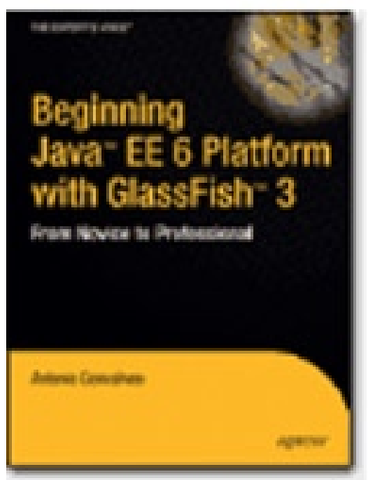
Composants HTML

Commandes

```
<h:commandButton value="A submit button"/>
<h:commandButton type="reset" value="A reset button"/>
<h:commandButton image="book.gif" title="A button with an image"/>
<h:commandLink>A hyperlink</h:commandLink>
```

A submit button

A reset button



A hyperlink

Lié à une action définie dans le Backing Bean

```
<h:commandLink action="#{bookController.doNew}">
  Create a new book
</h:commandLink>
```

Composants HTML

Cibles (navigation entre pages)

```
<h:button outcome="newBook.xhtml" value="A bookmarkable button link"/>
<h:link outcome="newBook.xhtml" value="A bookmarkable link"/>
```

A bookmarkable button link

A bookmarkable link

Paramètres

listBooks.xhtml

```
<h:link outcome="viewBook.xhtml" value="#{book.title}" >
  <f:param name="idBook" value="#{book.id}" />
</h:link>
```

viewBook.xhtml

```
<f:metadata>
   <f:viewParam name="idBook"
                value="#{bookController.book.id}" />
   <f:viewAction action="#{bookController.doFindBookById}" />
</f:metadata>
```

Liste de livres

Titre	Prix
Beginning Java EE 6	49.0
Beginning Java EE 7	53.0
The Lord of the Rings	23.0

Détails du livre

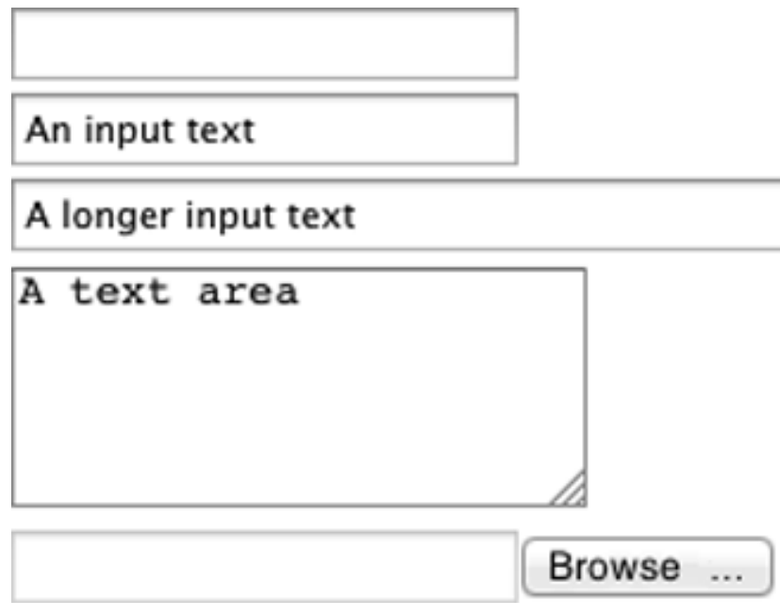
ISBN :	1234-5678
Titre :	Beginning Java EE 6
Prix :	49.0
Description :	Best Java EE book ever
Nombre de pages :	450
Illustrations :	oui

Ajouter au panier

Composants HTML

Entrées

```
<h:inputHidden value="Hidden data"/>
<h:inputSecret maxlength="8"/>
<h:inputText value="An input text"/>
<h:inputText size="40" value="A longer
                              input text"/>
<h:inputTextarea rows="4" cols="20" value="A text
                                           area"/>
<h:inputFile/>
```



Sortie / affichage

```
<h:outputLabel value="#{bookController.book.title}"/> <h:outputText value="A text"/>
<h:outputLink value="http://www.apress.com/">A link</h:outputLink>
<h:outputFormat value="Welcome {0}. You have bought {1} items">
  <f:param value="#{user.firstName}" />
  <f:param value="#{user.itemsBought}" />
</h:outputFormat>
```

Images

```
<h:graphicImage value="book.gif" height="200" width="320"/>
```

Composants HTML

Grilles / tableaux

```
<h:panelGrid columns="3" border="1">
  <h:outputLabel value="One"/>
  <h:outputLabel value="Two"/>
  <h:outputLabel value="Three"/>
  <h:outputLabel value="Four"/>
  <h:outputLabel value="Five"/>
  <h:outputLabel value="Six"/>
</h:panelGrid>
```

One	Two	Three
Four	Five	Six

```
<h:dataTable value="#{bookEJB.findBooks()}" var="book"
border="1">
  <h:column>
    <h:outputText value="#{book.title}" />
  </h:column>
  <h:column>
    <h:outputText value="#{book.price}" />
  </h:column>
</h:dataTable>
```

Beginning Java EE 6	49.0
Beginning Java EE 7	53.0
The Lord of the Rings	23.0

Tutoriel 4

Java Server Faces