

# Java EE 6 Tutorial Reloaded!

---

Antonio Goncalves & Alexis Moussine-Pouchkine

# Overall Presentation Goal

Focus on the new features of Java EE 6  
Write a web application

*Better if you know Java EE 5*

# What worked well

- Step-by-step
- Demo-driven
- Multiple IDEs
- Comprehensive



Who was here last year?

# What's new?

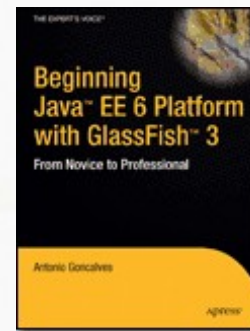
- Covering CDI 1.0
- Portability at work
  - JBoss 6 (Milestone 5) as additional runtime
- Multiple UI clients
  - Be back after the break!
- More polish, more feedback
- Hands-on labs
  - This afternoon @ 13:30 (3 hours)
  - Self-paced, take home if you'd like

# Agenda

- Overview of EE 6 and GlassFish 3
- Dive into some APIs with demos
  - JPA 2.0
  - Servlet 3.0
  - EJB 3.1
  - JSF 2.0
  - Bean Validation 1.0
  - JAX-RS 1.1
  - CDI 1.0
- Summary

# Antonio Goncalves

- Freelance software architect
- Former BEA consultant
- Author (Java EE 5 and Java EE 6)
- JCP expert member
- Co-leader of the Paris JUG
- Les Cast Codeurs podcast
- Java Champion



# Alexis Moussine-Pouchkine

- GlassFish Ambassador at **ORACLE**
- 11-year Sun and AppServer veteran
- Speaker at multiple conferences & JUGs
- Your advocate for anything GlassFish

## The Aquarium

A blog that has nothing to do with fish ...  
... but everything to do with Open Source Java EE, XML, SOA and more ...



## Bistro!

Alexis Moussine-Pouchkine's Weblog  
public enum Topic { **Java**, **GlassFish**,



The Java Spotlight Podcast



# Demo

## The Application

[Create a book](#) [Create a CD](#)

### Create a new book

Title :   
Price :   
Description :   
Number of pages :   
Illustrations :   
Tags :   
Language code \* :   
\* [FR, EN, RU, ES, DE, IT, FI]

[Create a book](#) [Create a CD](#)

### Create a new CD

Title :   
Price :   
Description :   
Music company :   
Number Of CDs :   
Total Duration :   
Style :

### List of CDs

ID	Title	Price	Description	Music Company	Number Of CDs	Total Duration	Style
1451							
351	1212	11.0			1	2.0	

### List of books

ID	ISBN	Title	Price	Language	Description	Number Of Pages	Illustrations	Tags	Purchase
2	1-84023-742-2	The Hitchhiker's Guide to the Galaxy	12.5	FR	Science fiction comedy book	354	false	scifi, fun	<input type="button" value="Buy me!"/>

Tutorial - Beginning with The Java EE 6 Platform

# GlassFish 3.x

<http://glassfish.org>



- The RI for Java EE 6
  - Home of Metro, Grizzly, Jersey, Mojarra, ...
- Yet, production-quality and open source
  - Growing number of production deployments
- Developer friendly, management-compatible
- Modular (OSGi) and Extensible (HK2)
- Fully supported by Oracle
- Full clustering and more in version 3.1

# JBoss 6.x



- Java EE 6 Web Profile compliant
  - Hibernate, RESTEasy, Weld, Mojarra...
- Still in Milestone 5 (RC1 coming soon)
- JBoss 7 in alpha

# A brief history



**Web Profile**

**Managed Bean**

# Java EE 6 APIs

## Web

<b>JSF</b>	<b>2.0</b>
<b>Servlet</b>	<b>3.0</b>
JSP	2.2
EL	2.2
JSTL	1.2
Debugging Support	1.0

## Enterprise

<b>EJB</b>	<b>3.1</b>
JAF	1.1
JavaMail	1.4
<b>JCA</b>	<b>1.6</b>
JMS	1.1
<b>JPA</b>	<b>2.0</b>
JTA	1.1

## Web Services

JAX-RPC	1.1
JAXM	1.0
<b>JAX-RS</b>	<b>1.1</b>
JAXR	1.0
Web Services	1.3
WS Metadata	2.0

## Management, Security, Common

<b>CDI (JSR 299)</b>	<b>1.0</b>
<b>@Inject (JSR 330)</b>	<b>1.0</b>
<b>Bean Validation</b>	<b>1.0</b>
<b>Interceptors</b>	<b>1.1</b>
<b>Managed Beans</b>	<b>1.0</b>
JACC	1.4
Java EE Application Deployment	1.2
Java EE Management	1.1
JASPIC	1.0

## + Java SE 6

### JAX-WS 2.2

JAXB	2.2
JDBC	4.0
JNDI	1.5
SAAJ	1.3

**Common** 1.1

### Annotations

RMI	
Java IDL	
JMX	
JAAS...	

# Pruning (Soon less specs)

- Marks specifications optional in next version
- Pruned in Java EE 6
  - Entity CMP 2.x
  - JAX-RPC
  - JAX-R
  - JSR 88 (Java EE Application Deployment)
- Might disappear from Java EE 7
  - Vendors may decide to keeps them...
  - ... or offer the delta as a set of modules

# Profiles

**Full Java EE 6**

The diagram illustrates the relationship between Java EE 6 profiles. A large white oval labeled 'Full Java EE 6' contains three smaller yellow ovals: 'Web Profile', 'Profile X', and 'Profile Y'. 'Web Profile' is the largest of the three and is positioned on the left. 'Profile X' is a smaller oval located in the upper right area of the 'Full Java EE 6' oval. 'Profile Y' is a large oval located in the lower right area, partially overlapping the bottom edge of the 'Full Java EE 6' oval.

**Web Profile**

**Profile X**

**Profile Y**

# Web Profile 1.0

- Subset of full platform
    - Fully functional right-sized profile
    - Not the kitchen sink
  - For web development
    - Packages in a war
  - Separate specification
  - Evolves at its own pace
  - Others considered
    - Portal, Telco, Integration, ...
- |                 |  |     |
|-----------------|--|-----|
| JSF             |  | 2.0 |
| Servlet         |  | 3.0 |
| JSP             |  | 2.2 |
| EL              |  | 2.2 |
| JSTL            |  | 1.2 |
| <b>EJB Lite</b> |  | 3.1 |
| Managed Beans   |  | 1.0 |
| Interceptors    |  | 1.1 |
| JTA             |  | 1.1 |
| JPA             |  | 2.0 |
| Bean Validation |  | 1.0 |
| CDI             |  | 1.0 |
| @Inject         |  | 1.0 |

# EJB Lite

- Subset of the EJB 3.1 API
- Used in Web profile
- Packaged in a war

Local Session Bean  
Injection  
CMT / BMT  
Interceptors  
Security

Message Driven Beans  
EJB Web Service Endpoint  
RMI/IIOP Interoperability  
Remote interface  
EJB 2.x  
Timer service  
CMP / BMP

# Portable JNDI names

- Client inside a container (use DI)

```
@EJB Hello h;
```

- Client outside a container

```
Context ctx = new InitialContext();  
Hello h = (Hello) ctx.lookup("xyz");
```

- Portable JNDI name is specified

```
java:global/foo/bar/HelloEJB
```

# Portable JNDI names

- **OrderBean** implements **Order** packaged in **orderejb.jar** within **orderpp.ear**

- `java:global/orderapp/orderejb/OrderBean`  
`java:global/orderapp/orderejb/OrderBean!`  
`org.foo.Order`

Usable from any application  
in the container

- `java:app/orderejb/OrderBean`  
`java:app/orderejb/OrderBean!`  
`com.acme.Order`

Fully-qualified interface name

- `java:module/OrderBean`  
`java:module/OrderBean!org.foo.Order`

# Managed Beans 1.0

- Separate new specification in Java EE 6
- Container-managed POJOs
- Small set of services
  - Injection (`@Resource`, `@Inject...`)
  - Life-cycle (`@PostConstruct`, `@PreDestroy`)
  - Interceptor (`@Interceptor`, `@AroundInvoke`)
- Lightweight component model
- Mostly a foundation for other components

# Managed Beans

## *Streamining the component model(s)*

```
@javax.annotation.ManagedBean
```

```
public class MyPojo {
```

```
    @Resource
```

```
    private DataSource ds;
```

```
    @PostConstruct
```

```
    private void init() {
```

```
        ....
```

```
    }
```

```
    @Interceptors (LoggingInterceptor.class)
```

```
    public void myMethod() {...}
```

```
}
```

*JSR 250*

*Commons annotations*

*Injection*

*Life-cycle*

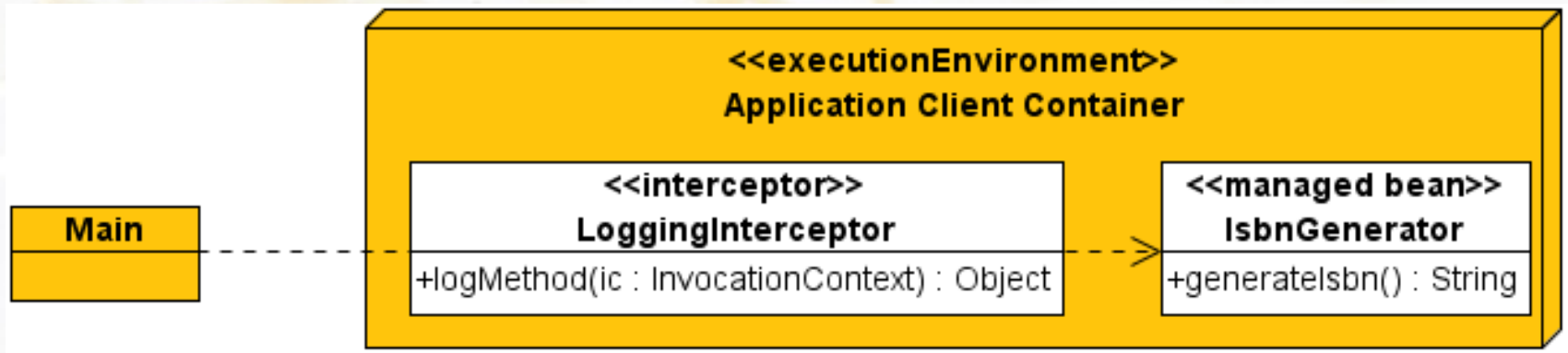
*Interceptors*

# Interceptors 1.1

- Address cross-cutting concerns in Java EE
- Were part of the EJB 3.0 spec
- Now a separate spec shipped with EJB 3.1
- Can be used in EJBs...
- ... as well as ManagedBeans
- `@AroundInvoke`
- `@AroundTimeout` for EJB timers

# Demo 1

## Managed Bean & Interceptor



# Agenda

- Overview of EE 6 and GlassFish v3
- Dive into some specs & demos
  - JPA 2.0
  - Servlet 3.0
  - EJB 3.1
  - JSF 2.0
  - Bean Validation 1.0
  - JAX-RS 1.1
  - CDI 1.0
- Summary

# JPA 2.0

- Evolves separately from EJB now
  - JSR 317
- Richer mappings
- Richer JPQL
- Standard config options
- Criteria API
- ...

# Richer mapping

- Collection of embeddables and basic types
  - Not just collection of JPA entities
  - Multiple levels of embeddables
- More flexible support for Maps
  - Keys, values can be one of : entities, embeddables or basic types
- More relationship mapping options
  - Unidirectional 1-many foreign key mappings

# Collections of Embeddable Types

```
@Embeddable public class BookReference {  
    String title;  
    Float price;  
    String description;  
    String isbn;  
    Integer nbOfPage;  
    ...  
}
```

```
@Entity public class ListOfGreatBooks {  
    @ElementCollection  
    protected Set<BookReference> javaBooks;  
    ...  
}
```

# Multiple levels of Embedding

```
@Embeddable public class BookReference {  
    @Embedded Author author;  
    ...  
}
```

```
@Entity public class Book {  
    @Id Long id;  
    String title;  
    BookReference theBook;  
    ...  
}
```

# Standard properties

- In `persistence.xml` :
  - `javax.persistence.jdbc.driver`
  - `javax.persistence.jdbc.url`
  - `javax.persistence.jdbc.user`
  - `javax.persistence.jdbc.password`
  - `javax.persistence.datasource`
  - `javax.persistence.lock.timeout`
  - ...

# Criteria API

- Strongly typed criteria API
- Object-based query definition objects
  - (Rather than JPQL string-based)
- Operates on the meta-model
  - Browse the structure of a Persistence Unit
  - Dynamically: `EntityManager.getMetamodel()`
  - Statically:  
Each entity **x** has a metamodel class **x\_**
- `CriteriaQuery` as a query graph

# Criteria API

```
EntityManager em = ...;
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Book> query =
    cb.createQuery(Book.class);

Root<Book> book = query.from(Book.class);

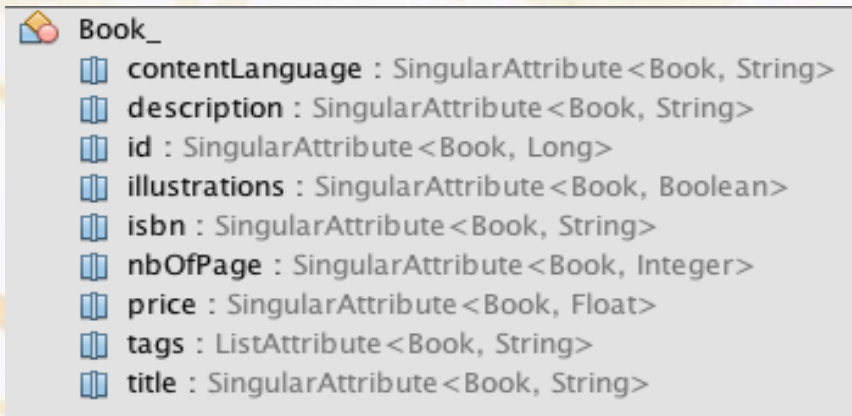
query.select(book)
    .where(cb.isNull(book.get("description")));
```

```
SELECT b
FROM Book b
WHERE b.description IS NULL
```

# Criteria API

## Type-safe

```
EntityManager em = ...;  
CriteriaBuilder cb = em.getCriteriaBuilder();  
CriteriaQuery<Book> query =  
    cb.createQuery(Book.class);  
  
Root<Book> book = query.from(Book.class);  
  
query.select(book)  
    .where(cb.isNull(book.get(Book_.description)));
```



A screenshot of a Java IDE showing the structure of the `Book_` class. The class is listed with a small icon. Below it, several attributes are listed, each with a book icon and its type: `contentLanguage` (SingularAttribute<Book, String>), `description` (SingularAttribute<Book, String>), `id` (SingularAttribute<Book, Long>), `illustrations` (SingularAttribute<Book, Boolean>), `isbn` (SingularAttribute<Book, String>), `nbOfPage` (SingularAttribute<Book, Integer>), `price` (SingularAttribute<Book, Float>), `tags` (ListAttribute<Book, String>), and `title` (SingularAttribute<Book, String>).

Statically generated  
JPA 2.0 MetaModel

# Criteria API

## Builder pattern

```
EntityManager em = ...;
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Book> query =
    cb.createQuery(Book.class);

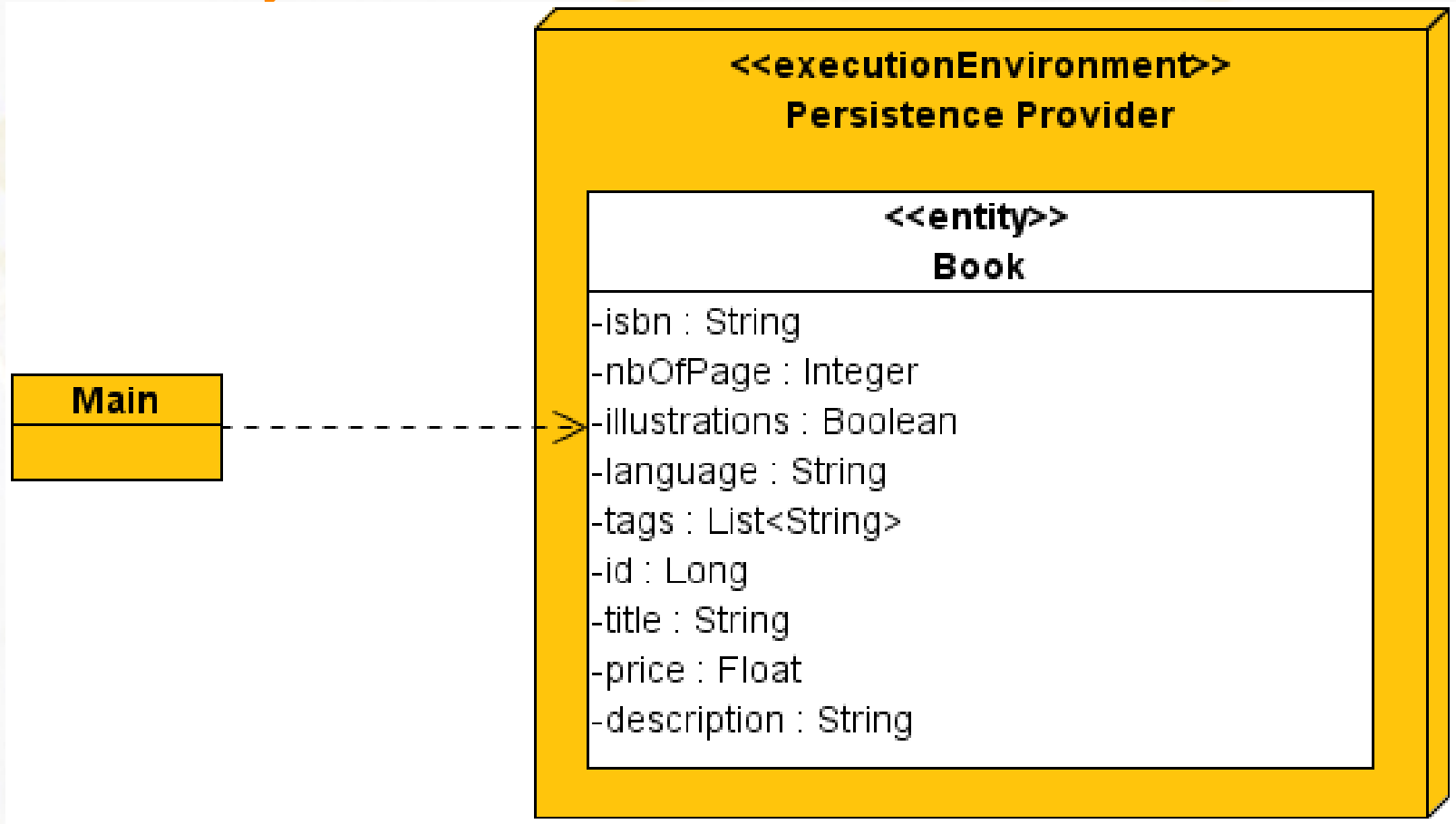
Root<Book> book = query.from(Book.class);

query.select(book)
    .where(cb.isNull(book.get(Book_.description)))
    .join(...)
    .orderBy(...)
    .distinct(true)
    .having(...)
    .groupBy(...);

List<Book> books = TypedQuery<Book>
    em.createQuery(query, Book.class).getResultList();
```

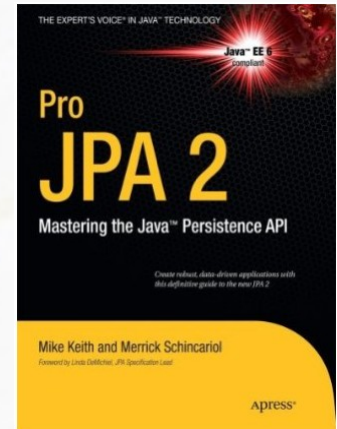
# Demo 2

## Book Entity



# And more...

- `detach()`
- `Join<X, Y>`, `ListJoin`, `MapJoin`
- Orphan removal functionality
  - `@OneToMany(orphanRemoval=true)`
- BeanValidation integration on lifecycle
- Second-level cache API
  - `@Cacheable` annotation on entities
  - `contain(Class, PK)`, `evict(Class, PK)`, ...
- Pessimistic locking options



# Agenda

- Overview of EE 6 and GlassFish v3
- Dive into some specs & demos
  - JPA 2.0
  - Servlet 3.0
  - EJB 3.1
  - JSF 2.0
  - Bean Validation 1.0
  - JAX-RS 1.1
  - CDI 1.0
- Summary

# Servlet 3.0

- Ease of development
- Pluggability
- *Asynchronous support*

# A servlet 3.0 example

```
@WebServlet(urlPatterns={"/MyApp"})
public class MyServlet extends HttpServlet {

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



*web.xml is optional*

- Same for @WebFilter  
and @WebListener

# Pluggability

- Enable use of frameworks without `web.xml`
  - Fragment the `web.xml` to allow frameworks to be self-contained in their own jar
- Dynamic container extension framework using `ServletContainerInitializer`
  - Simple JAR library can manipulate `ServletContext` at startup
- `/META-INF/resources` in any JAR to serve resources (applies to libraries)

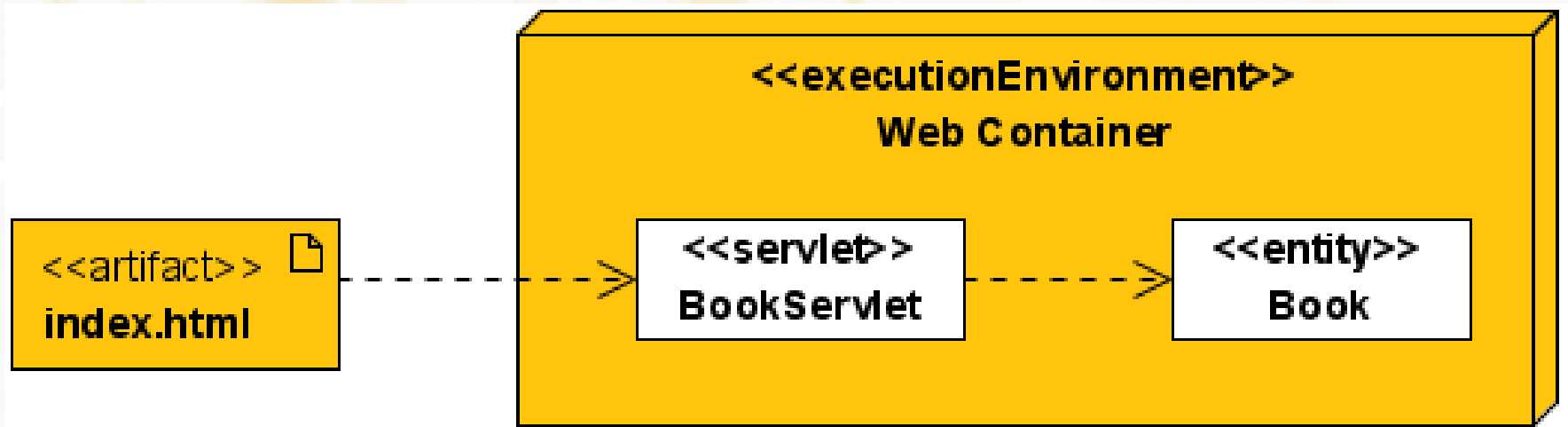
# Pluggability

## Web Fragments

- Fragments are similar to `web.xml`
- `<web-fragment>` instead of `<web-app>`
  - Declare their own servlets, listeners and filters
- Annotations and web fragments are merged following a configurable order
- JARs need to be placed in `WEB-INF/lib`
- and use `/META-INF/web-fragment.xml`
- Overridden by main `web.xml`

# Demo 3

## Add a Servlet



# And more...

- Async support (Comet-style)
- Configuration API
  - Add and configure Servlet, Filters, Listeners
  - Add security constraints
  - Using `ServletContext` API
- File upload (similar to Apache File Upload)
- Configure cookie session name
- Security with `@ServletSecurity`

# Agenda

- Overview of EE 6 and GlassFish v3
- Dive into some specs & demos
  - JPA 2.0
  - Servlet 3.0
  - EJB 3.1
  - JSF 2.0
  - Bean Validation 1.0
  - JAX-RS 1.1
  - CDI 1.0
- Summary

# EJB Optional Local Interface

- @Local, @Remote
- Interfaces are not always needed
  - Only for local interfaces
  - Remote interfaces are now optional !

Create Interface:

Remote

Local

**@Stateless**

```
public class HelloBean {  
  
    public String sayHello() {  
        return "Hello Devoxx";  
    }  
}
```

# Packaging in a war

**foo.ear**

lib/foo\_common.jar

com/acme/**Foo.class**

foo\_web.war

WEB-INF/**web.xml**

WEB-INF/classes

com/acme/**FooServlet.class**

foo\_ejb.jar

com/acme/**FooEJB.class**

com/acme/**FooEJBLocal.class**

**foo.war**

WEB-INF/classes

com/acme/**Foo.class**

com/acme/**FooServlet.class**

com/acme/**FooEJB.class**

# Asynchronous calls

- How to have asynchronous call in EJBs ?
  - JMS is more about sending messages
  - Threads and EJB's don't integrate well
- `@Asynchronous`
  - Applicable to any EJB type
  - Best effort, no delivery guarantee
- Method returns `void` or `Future<T>`
  - `javax.ejb.AsyncResult` helper class :  
`return new AsyncResult<int>(result)`

# Asynchronous calls

```
@Stateless
public class OrderBean {

    public void createOrder() {
        Order order = persistOrder();
        sendEmail(order); // fire and forget
    }

    public Order persistOrder() {...}

    @Asynchronous
    public void sendEmail(Order order) {...}
}
```

# Timer Service

- Programmatic and Calendar based scheduling
  - « Last day of the month »
  - « Every five minutes on Monday and Friday »
- Cron-like syntax
  - `second [0..59], minute[0..59], hour[0..23]...`
  - `dayOfMonth[1..31]`
  - `dayOfWeek[0..7] or [sun, mon, tue..]`
  - `Month[0..12] or [jan,feb..]`

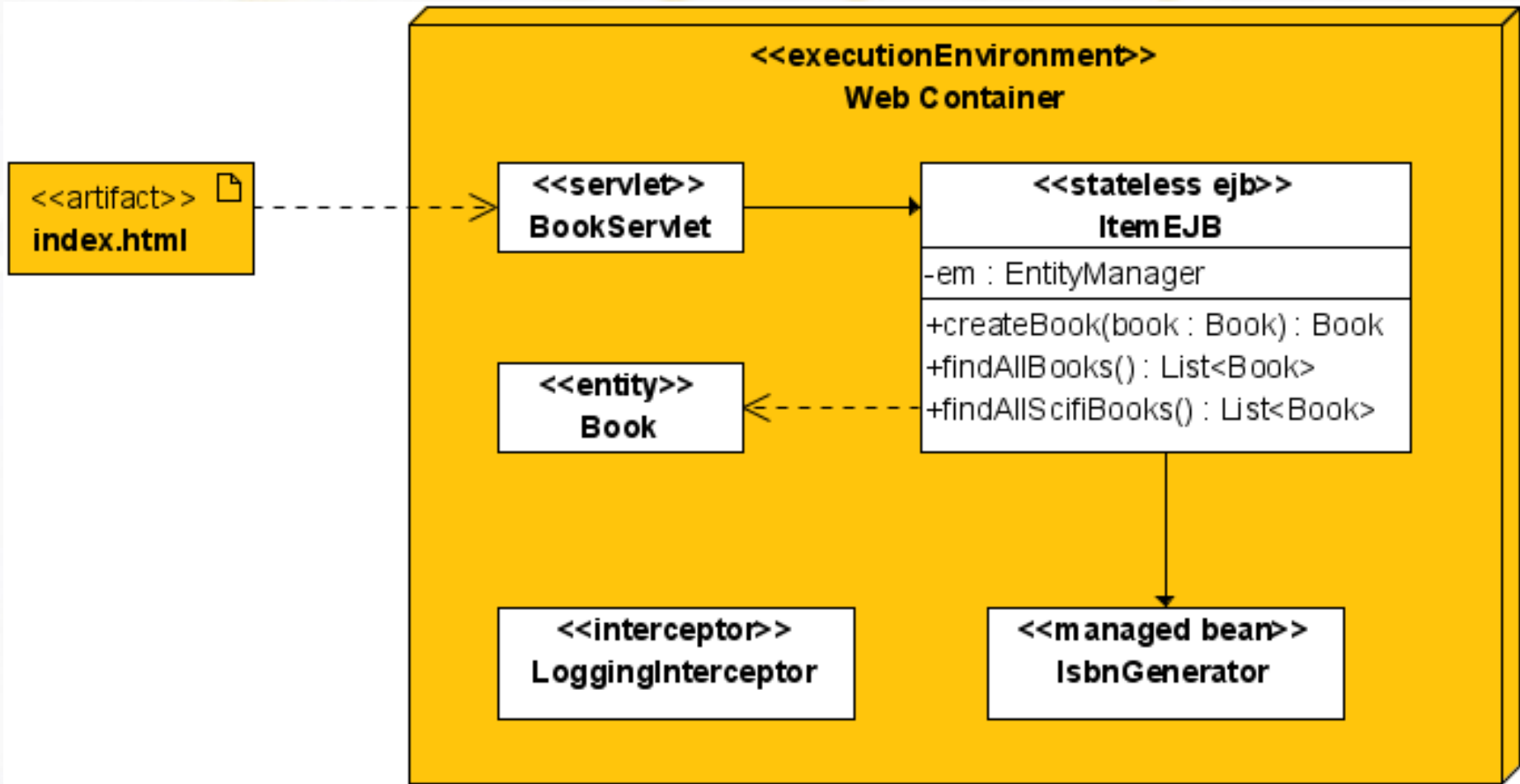
# Timer Service

```
@Stateless  
public class WakeUpBean {  
  
    @Schedule (dayOfWeek="Mon-Fri", hour="9")  
    void wakeUp() {  
        ...  
    }  
}
```

No container config required

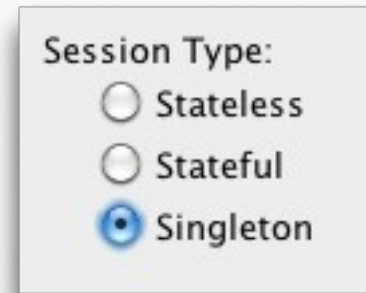
# Demo 4

## Add an EJB stateless



# Singleton

- New component
  - No/local/remote interface
- Follows the Singleton pattern
  - One single EJB per application per JVM
- Used to share state in the entire application
  - State not preserved after container shutdown
- Added concurrency management
  - Default is single-threaded
  - `@ConcurrencyManagement`



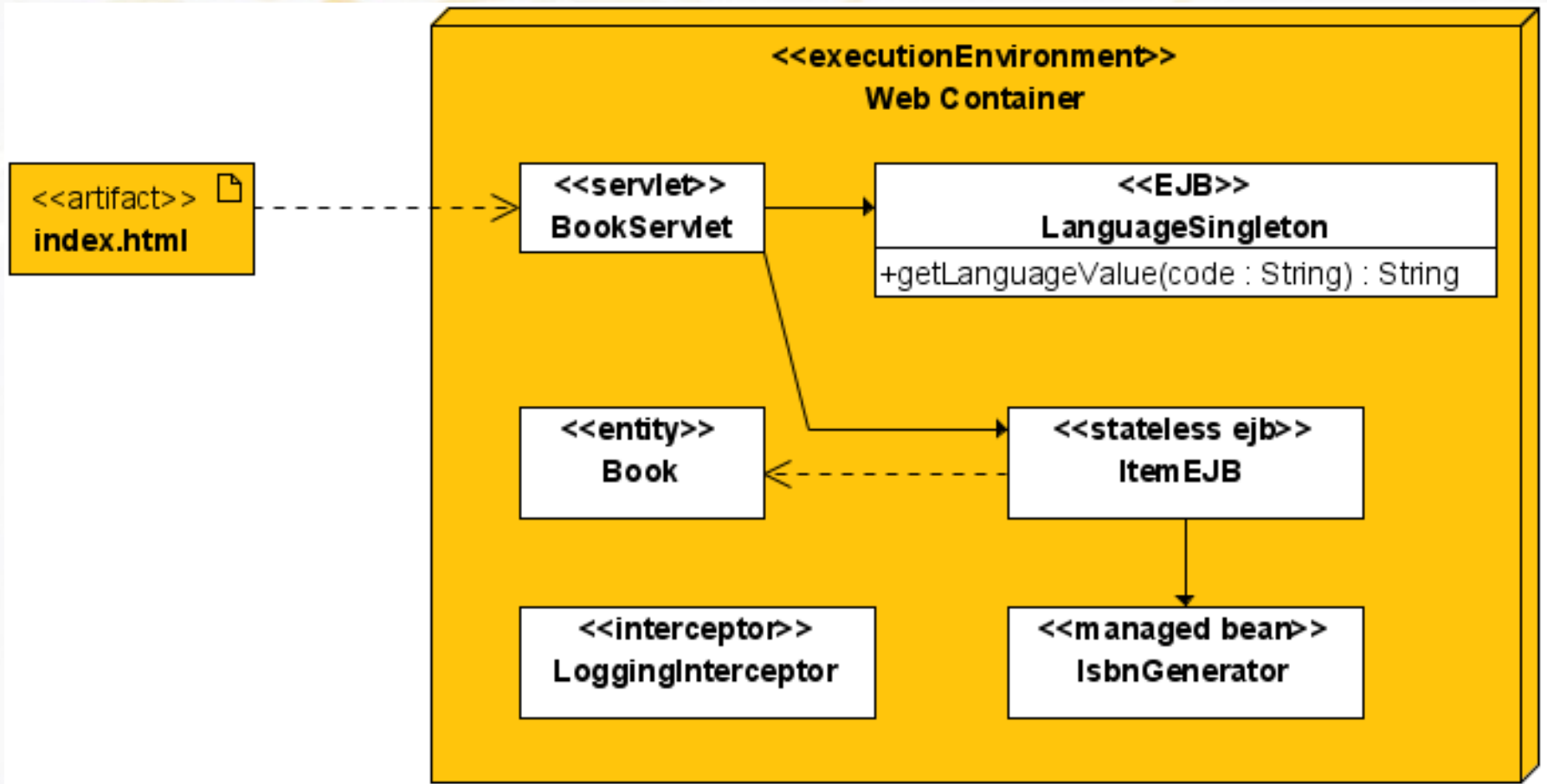
# Singleton

**@Singleton**

```
public class CachingBean {  
    private Map cache;  
  
    @PostConstruct void init() {  
        cache = ...;  
    }  
  
    public Map getCache() {  
        return cache;  
    }  
  
    public void addToCache(Object key, Object val) {  
        cache.put(key, val);  
    }  
}
```

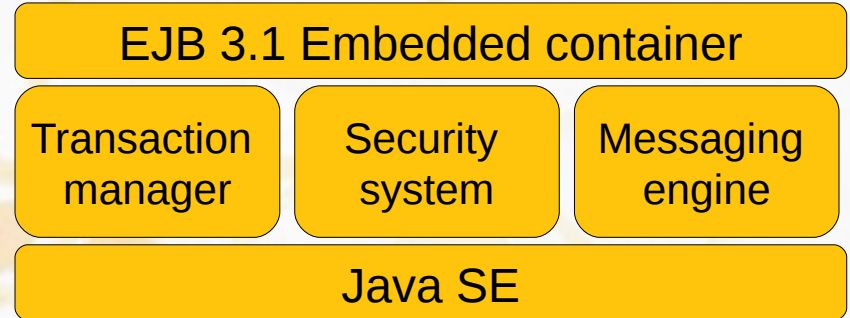
# Demo 5

## Add a Singleton EJB



# Embeddable Container

- API allowing to :
  - Initialize a container
  - Get container context
  - ...



- Can run in any Java SE environment
  - Batch processing
  - Simplifies testing
  - Just a jar file in your classpath

# Embeddable Container

```
public static void main(String[] args) {  
    EJBContainer container =  
        EJBContainer.createEJBContainer() ;  
  
    Context context = container.getContext() ;  
    Hello h = (Hello)  
        context.lookup("Global_JNDI_Name") ;  
    h.sayHello() ;  
  
    container.close() ;  
}
```

# Demo 6

Test using EJBContainer

# And more...

- Interceptors and InterceptorBinding
- Singletons can be chained
- Non persistent timer
- `@StatefulTimeout`
- ...

# Agenda

- Overview of EE 6 and GlassFish v3
- Dive into some specs & demos
  - JPA 2.0
  - Servlet 3.0
  - EJB 3.1
  - JSF 2.0
  - Bean Validation 1.0
  - JAX-RS 1.1
  - CDI 1.0
- Summary

# JavaServer Faces (JSF) 2.0

- The standard component-oriented MVC framework
- Part of Java EE 5, Java EE 6 (Web Profile)
  - Does not require Servlet 3.0
  - Other frameworks can rely on EE 6 extensibility
- Deserves its 2.0 version number
  - New features, issues fixed, performance focus

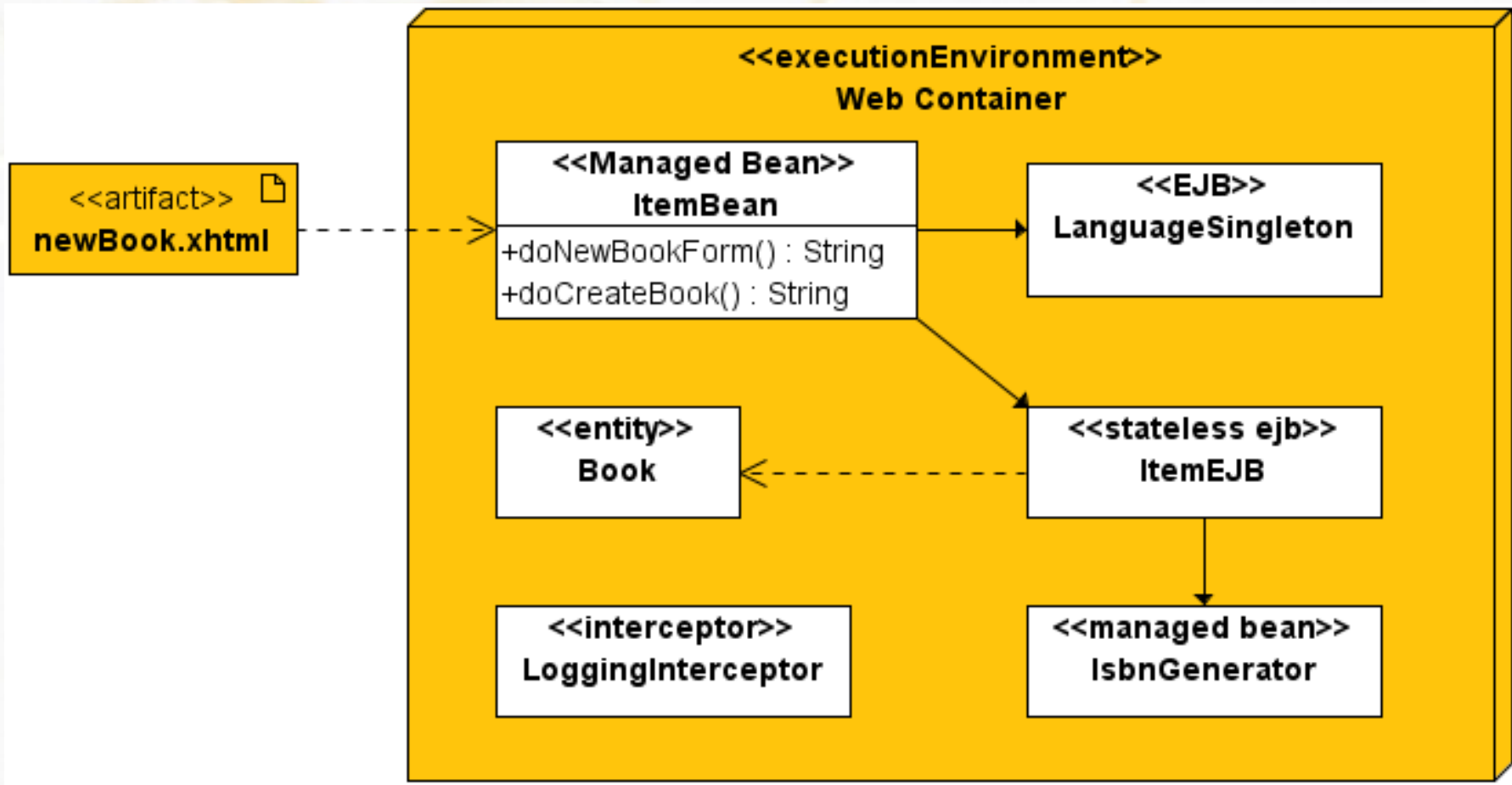
# Less artifacts, better artifacts

- Facelets (XHTML) as alternative to JSP
- IDEs offer traditional value-add:
  - Auto-completion (EL)
  - (Composite) Component management
  - Project management, testing, etc...
- `faces-config.xml` now optional
  - `@javax.faces.bean.ManagedBean*`
  - Navigation can now belong to the page (`<navigation-rules>` become optional)

\*: Not required with JSR 299 (CDI)

# Demo 7

## Add a JSF page



# JSF Components

- Rather healthy component market
- Pretty good IDE support but...

# JSF Components

- Rather healthy component market
- Pretty good IDE support but...
- Building your own components with JSF 1.x was (much) harder than it should be
- *Bummer for an MVC “component” framework...*

# JSF Composite Component

- Using JSF 1.x
  - Implement `UIComponent`, markup in renderer, register in `faces-config.xml`, add `tld`, ...
- With JSF 2.0
  - Single file, no Java code needed
  - Use XHTML and JSF tags to create components

```
<html xmlns:cc="http://java.sun.com/jsf/composite">
```

```
<cc:interface>
```

```
  <cc:attribute ...>
```

```
</cc:implementation>
```

- Everything else is auto-wired

## ./web/resources/ezcomp/mycomponent.xhtml

```
<!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://java.sun.com/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite">
  <!-- INTERFACE -->
  <composite:interface>
    <composite:attribute name="param"/>
  </composite:interface>
  <!-- IMPLEMENTATION -->
  <composite:implementation>
    <h:outputText value="Hello there, #{cc.attrs.param}"/>
  </composite:implementation>
</html>
```

## Using the component

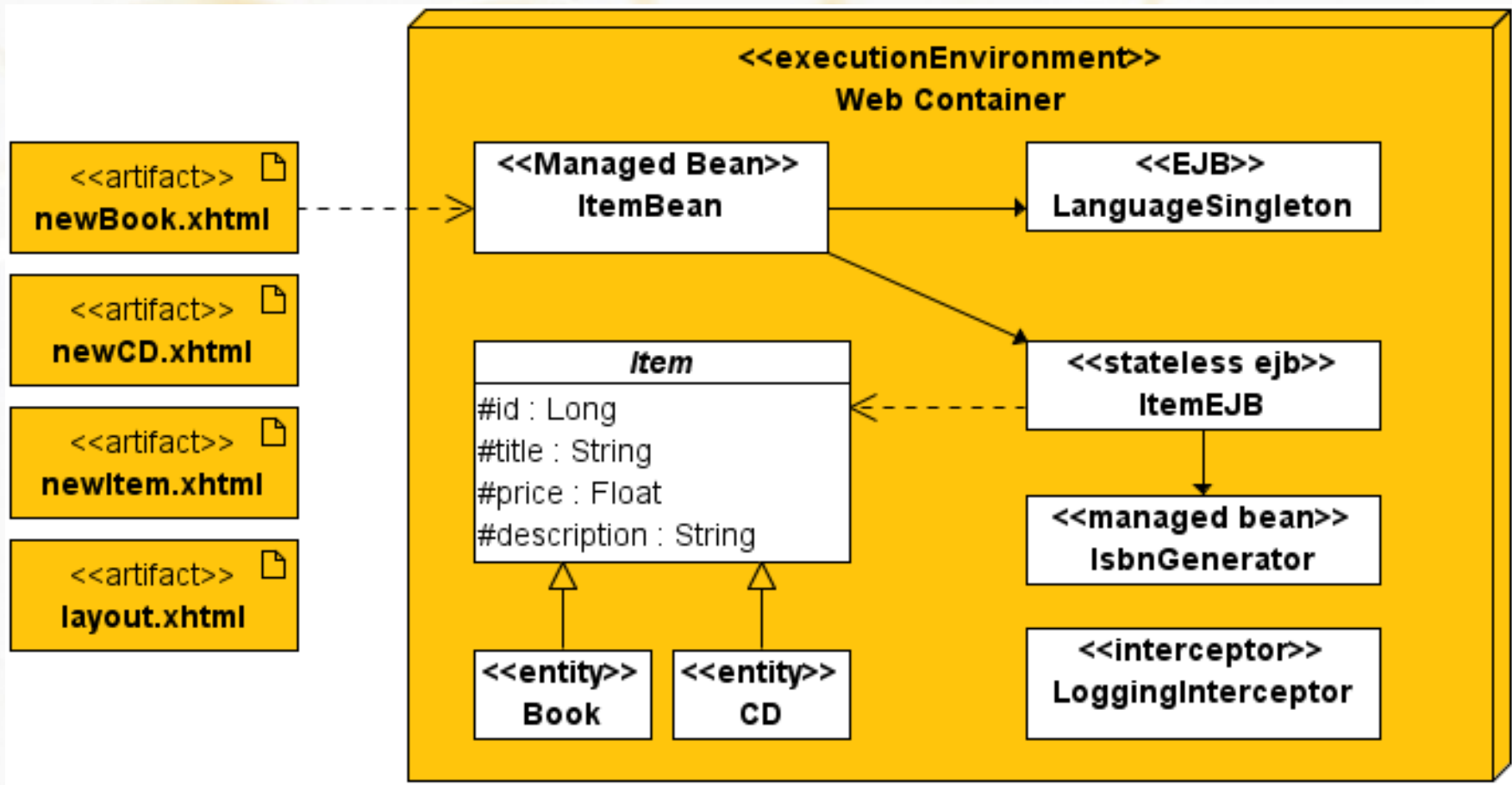
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:custom="http://java.sun.com/jsf/composite/ezcomp">
  <h:body>
    <custom:mycomponent param="Devovx attendees"/>
  </h:body>
</html>
```

Defining the component

*Implicit EL object*

# Demo 8

2 entities 1 JSF component



# Ajax support

*Look 'ma, no JavaScript!*

- Inspired by RichFaces, IceFaces, DynaFaces, ...
- Common JavaScript library (`jsf.js`)
  - request JavaScript functions captured by `PartialViewContext` for sub-tree processing
  - Client JavaScript updates the DOM

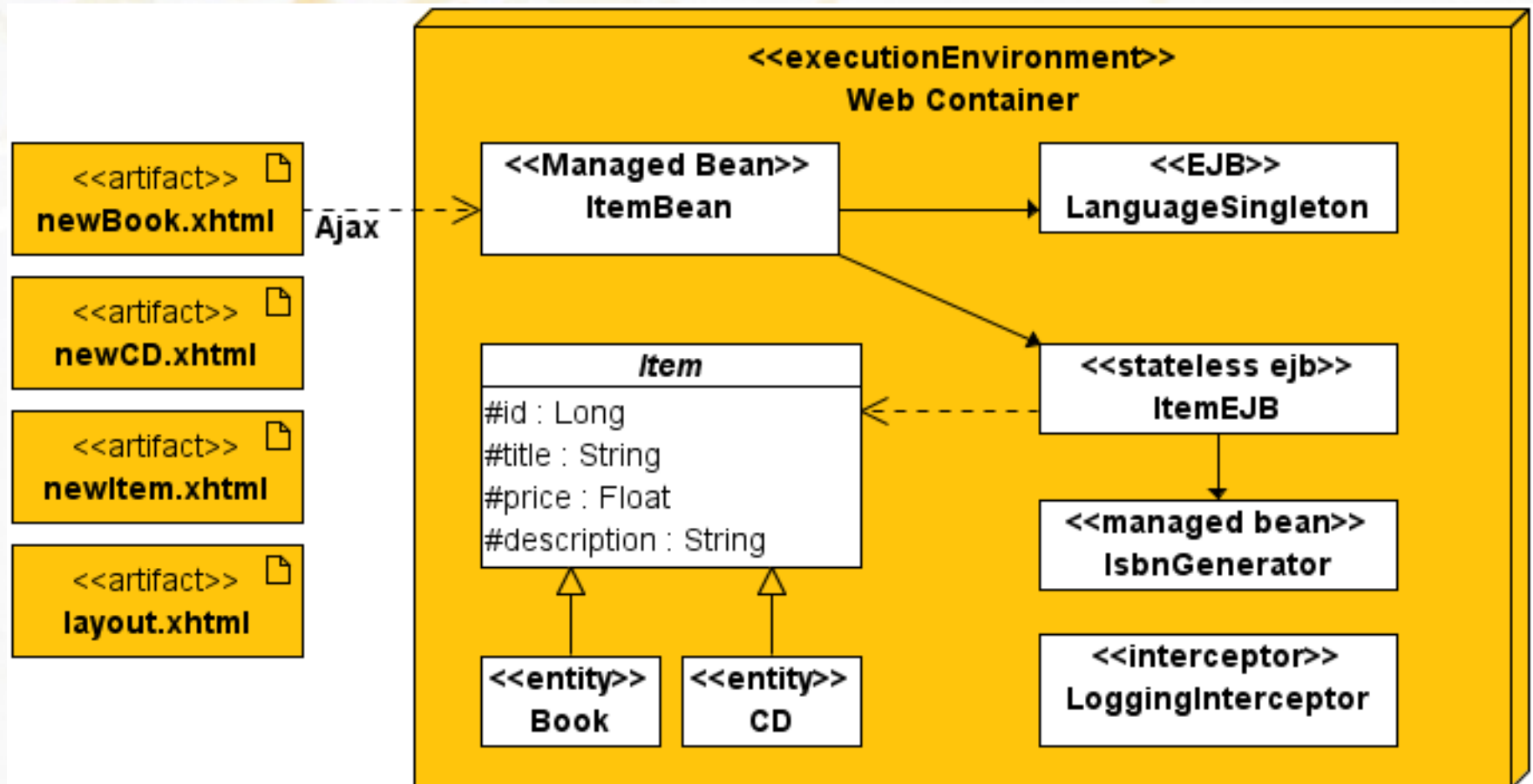
```
<h:commandButton  
    onclick="jsf.ajax.request(this,event,{render:'foo'});  
    return false;"/>
```

- `<f:ajax>` tag to ajaxify existing pages

```
<xmlns:f="http://java.sun.com/jsf/core"/>  
<h:commandButton>  
    <f:ajax event="change" execute="myForm" render="foo" />  
</h:commandButton>
```

# Demo 9

## Ajax call



# And more...

- Validation delegated to BeanValidation
- Easier resources management
- Better error reporting
- New managed bean scope (View)
- Groovy support (Mojarra)
- Bookmarkable URLs
- Templating : define and apply layouts
- Project stages (dev vs. test vs. production)
- ...

# Break

Cool demos when we're back!  
(REST, mutiple clients, CDI, ...)

# Agenda

- Overview of EE 6 and GlassFish v3
- Dive into some specs & demos
  - JPA 2.0
  - Servlet 3.0
  - EJB 3.1
  - JSF 2.0
  - Bean Validation 1.0
  - JAX-RS 1.1
  - CDI 1.0
- Summary

# Bean Validation 1.0

- Enable declarative validation in your applications
- Constrain Once, Validate Anywhere
  - restriction on a bean, field or property
  - not null, size between 1 and 7, valid email...
- Standard way to validate constraints
- Integration with JPA 2.0 & JSF 2.0

# Bean Validation 1.0

```
public class Address {  
    @NotNull @Size(max=30,  
        message="longer than {max}  
characters")  
    private String street1;  
    ...  
    @NotNull @Valid  
    private Country country;  
}
```

```
public class Country {  
    @NotNull @Size(max=20)  
    private String name;  
    ...  
}
```

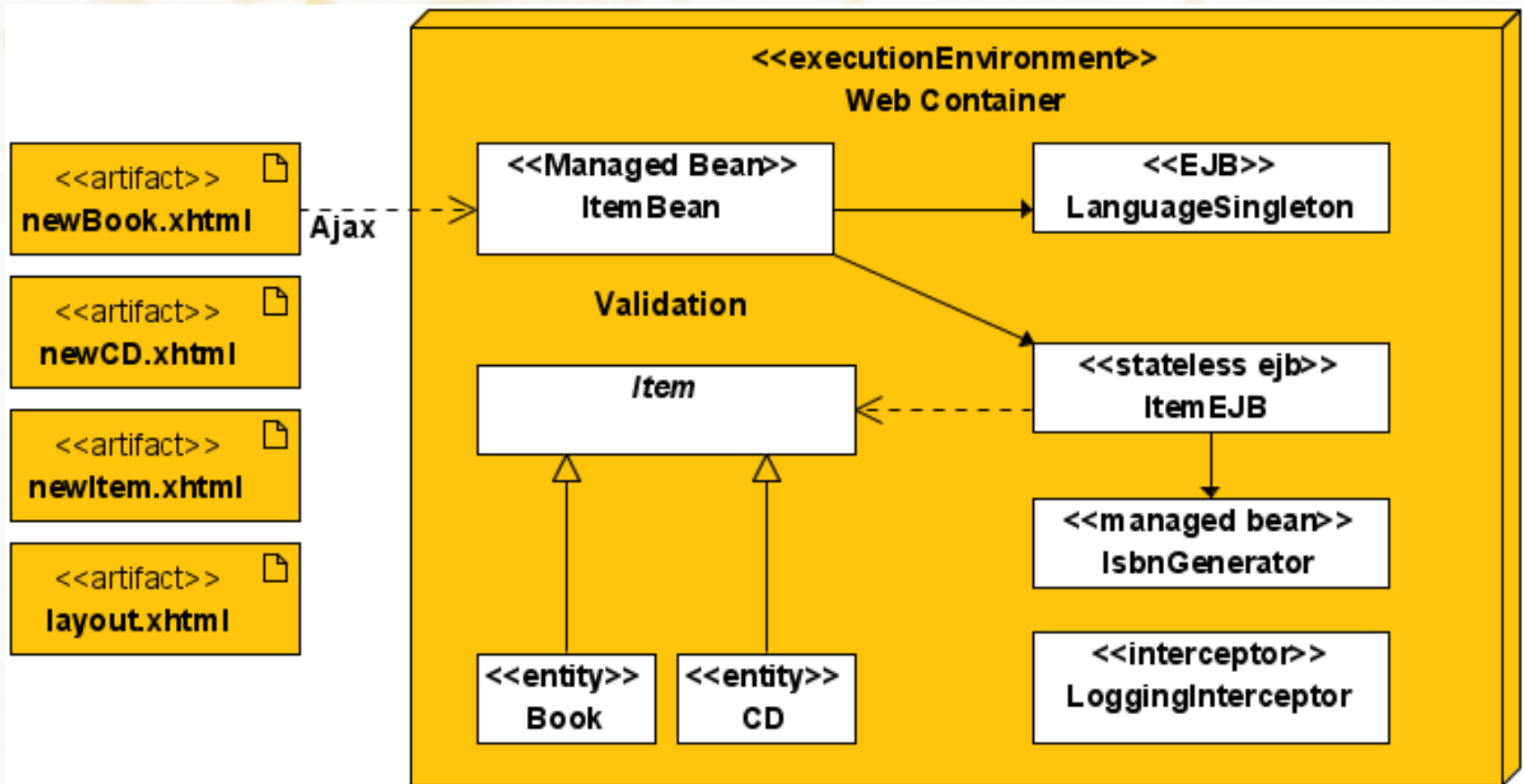
request recursive  
object graph  
validation

# Build your own!

```
@Size(min=5, max=5)
@ConstraintValidator(ZipcodeValidator.class)
@Documented
@Target({ANNOTATION_TYPE, METHOD, FIELD})
@Retention(RUNTIME)
public @interface ZipCode {
    String message() default "Wrong zipcode";
    String[] groups() default {};
}
```

# Demo 10

## Validation Item/ItemBean



# And more...

- Group subsets of constraints
- Partial validation
- Order constraint validations
- Create your own
- Bootstrap API
- Messages can be i18n
- ...

# Agenda

- Overview of EE 6 and GlassFish v3
- Dive into some specs & demos
  - JPA 2.0
  - Servlet 3.0
  - EJB 3.1
  - JSF 2.0
  - Bean Validation 1.0
  - JAX-RS 1.1
  - CDI 1.0
- Summary

# JAX-RS 1.1

- High-level HTTP API for RESTful Services
- POJO and Annotations Based
  - API also available
- Maps HTTP verbs (Get, Post, Put, Delete...)
- JAX-RS 1.0 in 2008
- JAX-RS 1.1 integrates with EJBs  
(and more generally with Java EE 6)
  - CDI integration possible but may be tricky

# Hello World

```
@Path("/helloworld")  
public class HelloWorldResource {  
  
    @GET  
    @Produces("text/plain")  
    public String sayHello() {  
        return "Hello World";  
    }  
}
```

**GET http://example.com/helloworld**

# Hello World

## Request

---

```
GET /helloworld HTTP/1.1  
Host: example.com  
Accept: text/plain
```

## Response

---

```
HTTP/1.1 200 OK  
Date: Wed, 12 Nov 2008 16:41:58 GMT  
Server: GlassFish v3  
Content-Type: text/plain; charset=UTF-8  
Hello World
```

# Different Mime Types

```
@Path("/helloworld")
public class HelloWorldResource {

    @GET @Produces("image/jpeg")
    public byte[] paintHello() {
        ...
    }

    @GET @Produces("text/plain")
    public String displayHello() {
        ...
    }

    @POST @Consumes("text/xml")
    public void updateHello(String xml) {
        ...
    }
}
```

# Parameters & EJBs

```
@Path("/users/{userId}")
@Stateless
public class UserResource {

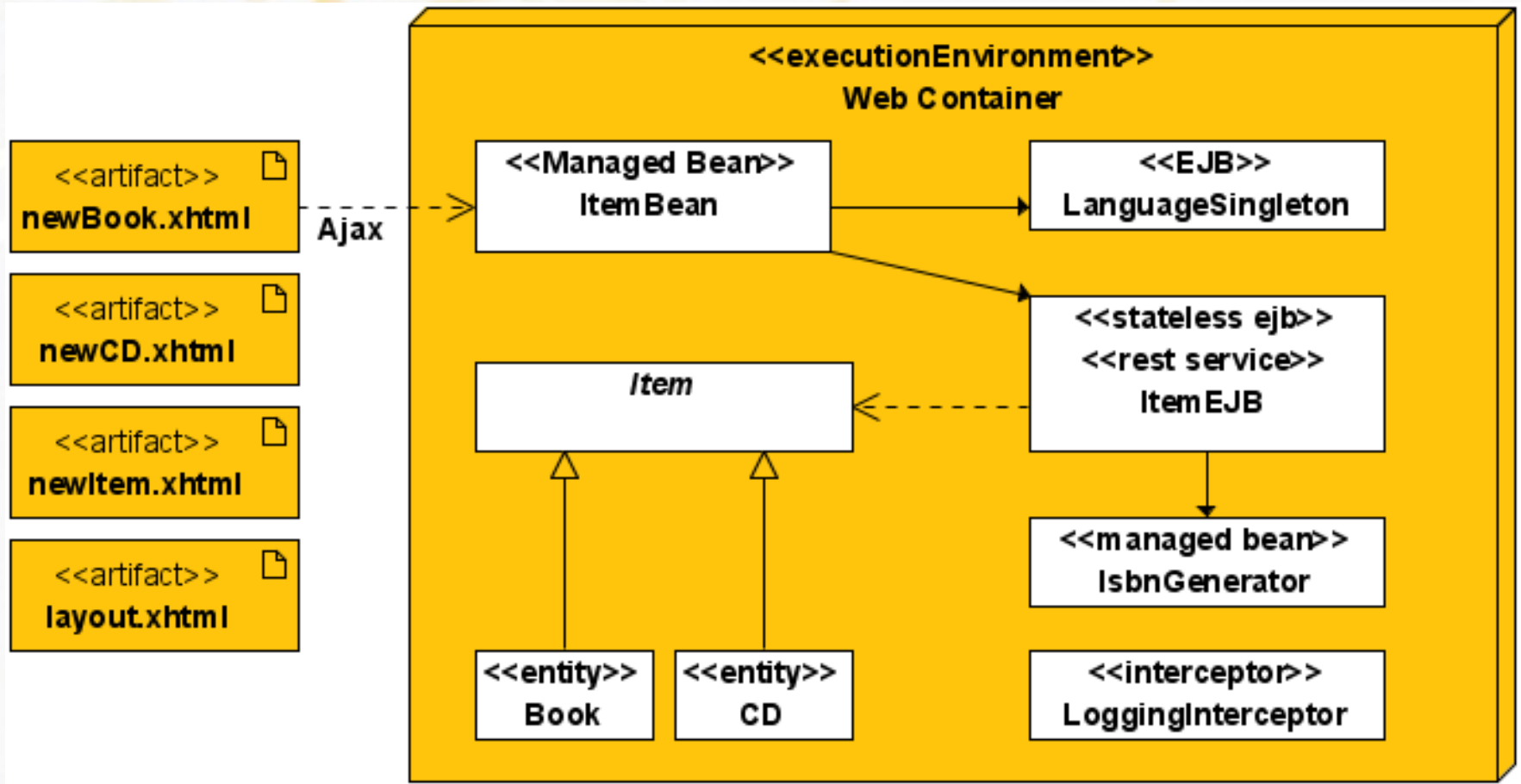
    @PersistenceContext
    EntityManager em;

    @GET @Produces("text/xml")
    public String getUser(@PathParam("userId")
                          String id) {

        User u = em.find(User.class, id)
        ...
    }
}
```

# Demo 11

## Add REST service to EJB



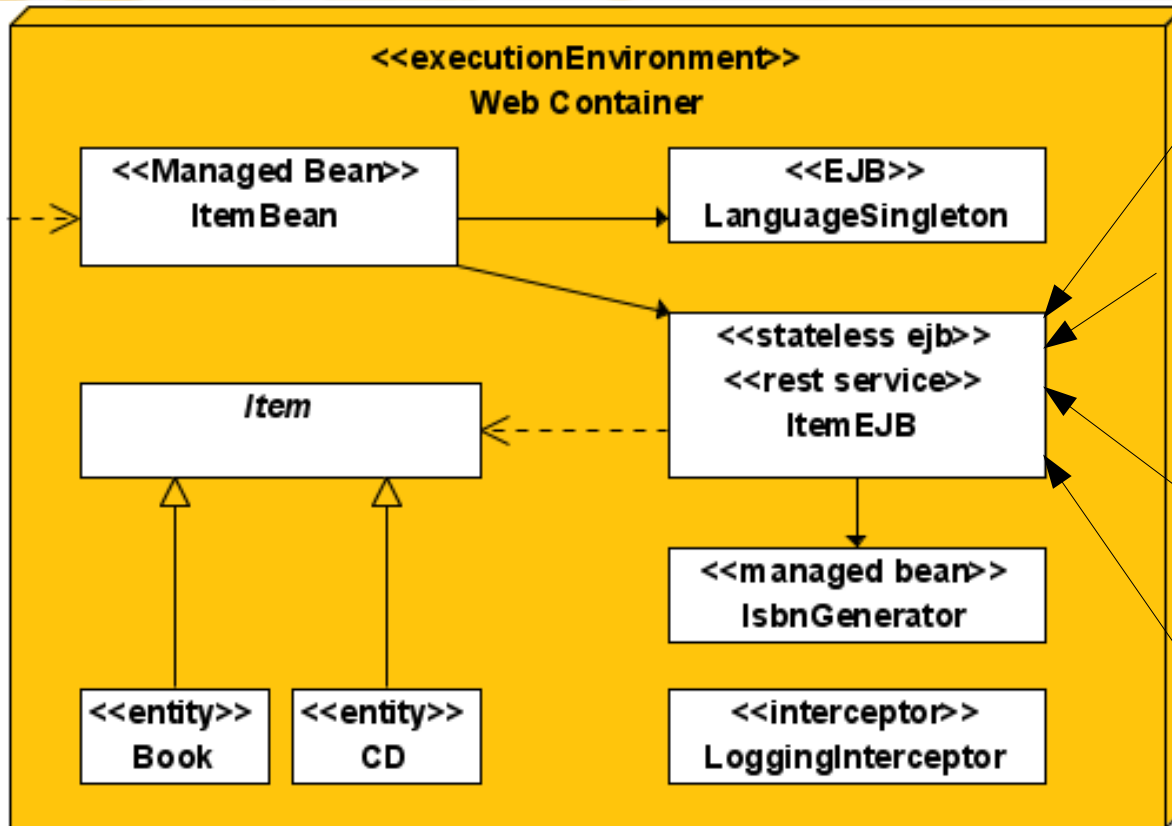
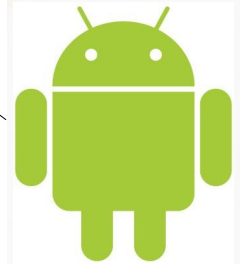
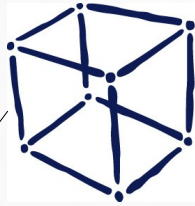
# And more...

- **Different parameters** (`@MatrixParam`, `@QueryParam`, `@CookieParam` ...)
- **Support for** `@Head` **and** `@Option`
- **Inject** `UriInfo` **using** `@Context`
- **JAX-RS servlet mapping with**  
`@ApplicationPath`
- **Integrates well with many other technologies**
- ...

# Demo 12

## Multiple UI using JAX-RS

NetBeans Platform



Ajax

# Many thanks for your help !

- Android
  - **Gabriel Kastenbaum**
  - <http://www.paperolle.com>
- GWT
  - **Tanguy Bayard**
  - <http://fr.linkedin.com/in/tanguybayard>
- NetBeans Platform
  - **Geertjan Wielanga**
- Java FX
  - **Sébastien Stormacq**

# Agenda

- Overview of EE 6 and GlassFish v3
- Dive into some specs & demos
  - JPA 2.0
  - Servlet 3.0
  - EJB 3.1
  - JSF 2.0
  - Bean Validation 1.0
  - JAX-RS 1.1
  - **CDI 1.0**
- Summary

# Injection in Java EE 5

- **Common Annotation**
  - `@Resource`
- **Specialized cases**
  - `@EJB`, `@WebServicesRef`,  
`@PersistenceUnit` ...
- **Requires *managed* objects**
  - EJB, Servlet and JSF Managed Bean in EE 5
  - Also in any Java EE 6's  
`javax.annotation.ManagedBean`

# Injection in Java EE 6

CDI (JSR 299)  
&  
DI (JSR 330)

*Inject just about anything anywhere...*

*...yet with strong typing*

# The tale of 2 dependency JSRs

- Context & Dependency Injection for Java EE
  - Born as WebBeans, unification of JSF and EJB
  - “Loose coupling, strong typing”
  - Weld (RI), Caucho CanDI, Apache Open WebBeans
- Dependency Injection for Java (JSR 330)
  - Lead by Google and SpringSource
  - Minimalistic dependency injection, `@Inject`
  - Applies to Java SE, Guice as the reference impl.
- Both aligned and part of Java EE 6 Web Profile

# @Named and @Inject

- **CDI requires a WEB-INF/beans.xml\* file**
  - Can be empty
  - Beans auto-discovered at startup
- **@Named makes the bean available to EL**
  - Prefer @Named to @ManagedBean (JSF or JSR 250)
- **Use @Inject to, well inject**
  - @Inject ISBNGenerator generator;
- **@Resource still around**
  - Use it for DB connexions, queues, RA's
  - Anything App-managed: use @Inject

\*: META-INF for JAR/EAR

# Stereotype

@Named

@RequestScoped

# Stereotype

@Named

@RequestScoped

@Documented

@Stereotype

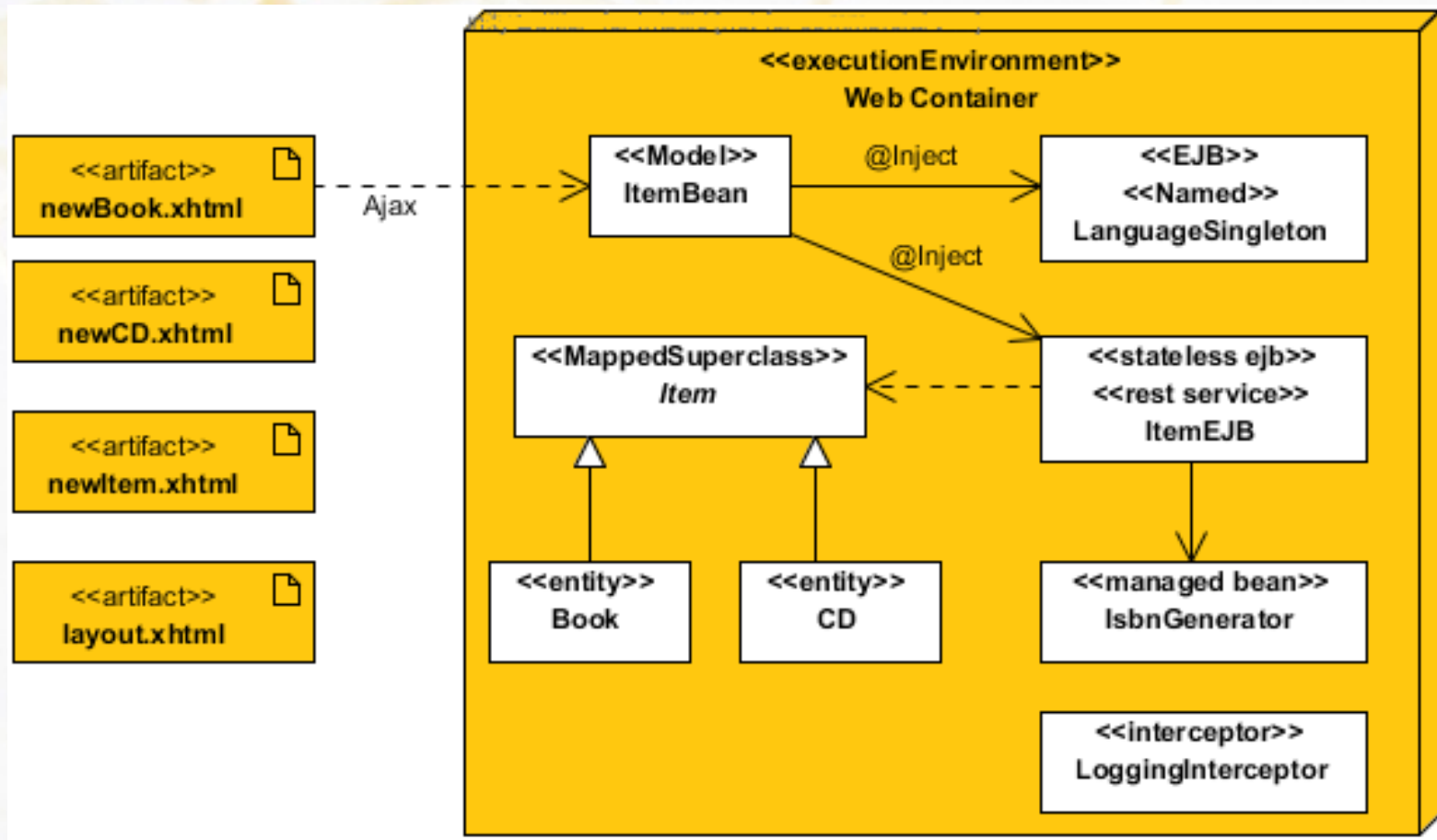
@Target(value={TYPE, METHOD, FIELD})

@Retention(value=RUNTIME)

public @interface **Model**

# Demo 13

Enable CDI and refactor @Resource into @Inject



**qualifier** (user-defined label)  
i.e. « which one? »

```
@Inject @Premium Customer cust;
```

injection point

type

# Qualifier Annotation

```
@Target ( {TYPE, METHOD, PARAMETER, FIELD} )
```

```
@Retention (RUNTIME)
```

```
@Documented
```

```
@Qualifier
```

```
public @interface Premium { ... }
```

```
@Premium // my own qualifier (see above)
```

```
public class SpecialCustomer
```

```
    implements Customer {
```

```
        public void buy() { ... }
```

```
    }
```

# Loose-coupling

**qualifier** (user-defined label)  
i.e. « which one? »

```
@Inject @Premium Customer cust;
```

injection point

type

# Alternatives

- Resolve injection ambiguity with beans.xml
  - When XML is better than Java ;-)

@Alternative

```
public class SpecialCustomer implements Customer {...}
```

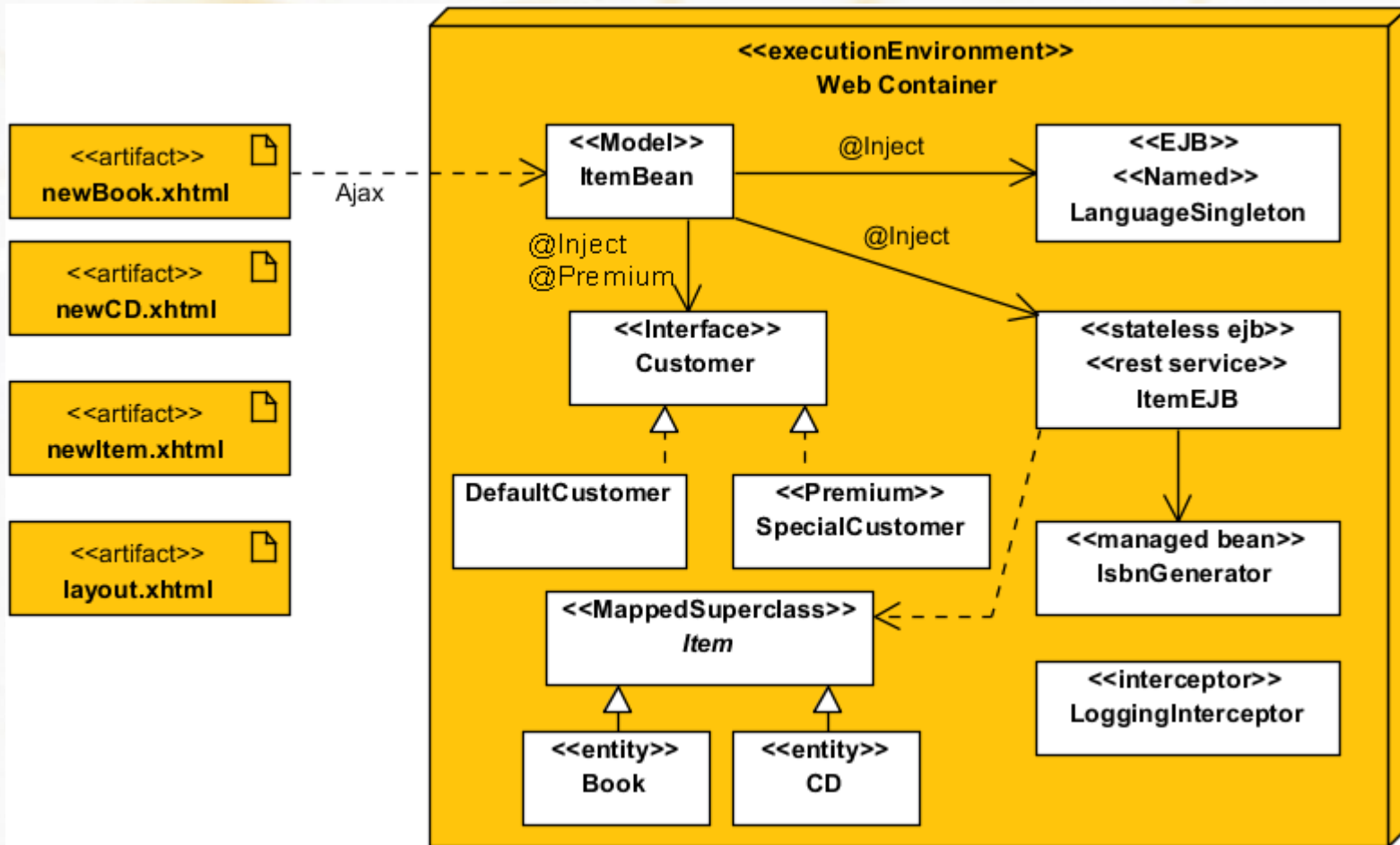
@Alternative

```
public class DefaultCustomer implements Customer {...}
```

```
<beans ...>  
  <alternatives>  
    <class>SpecialCustomer</class>  
  </alternatives>  
</beans>
```

# Demo 14

## Use CDI qualifiers (and alternatives)



# Contexts

## The 'C' in CDI

- Built-in “Web” Scopes :

\*: requires `Serializable` fields to enable passivation

- `@RequestScoped`
- `@SessionScoped*`
- `@ApplicationScoped*`
- **`@ConversationScoped*`**

- Other Scopes

- `@Dependent` is the default pseudo-scope for un-scoped beans (*same as Managed Beans*)
- Build your own `@ScopeType`

- Clients need not be scope-aware

# @ConversationScoped

- *A conversation* is :
  - explicitly demarcated
  - associated with individual browser tabs
  - accessible from any JSF request

@Named

**@ConversationScoped**:

```
public class ItemFacade implements Serializable {  
    @Inject Conversation conversation;  
    ...  
    conversation.begin(); // long-running  
    ...  
    conversation.end(); // schedule for destruction
```

# javax.enterprise.event.Event

- Loosely defined :

```
@Inject  
Event<OrderItem> order;
```

- Can be qualified :

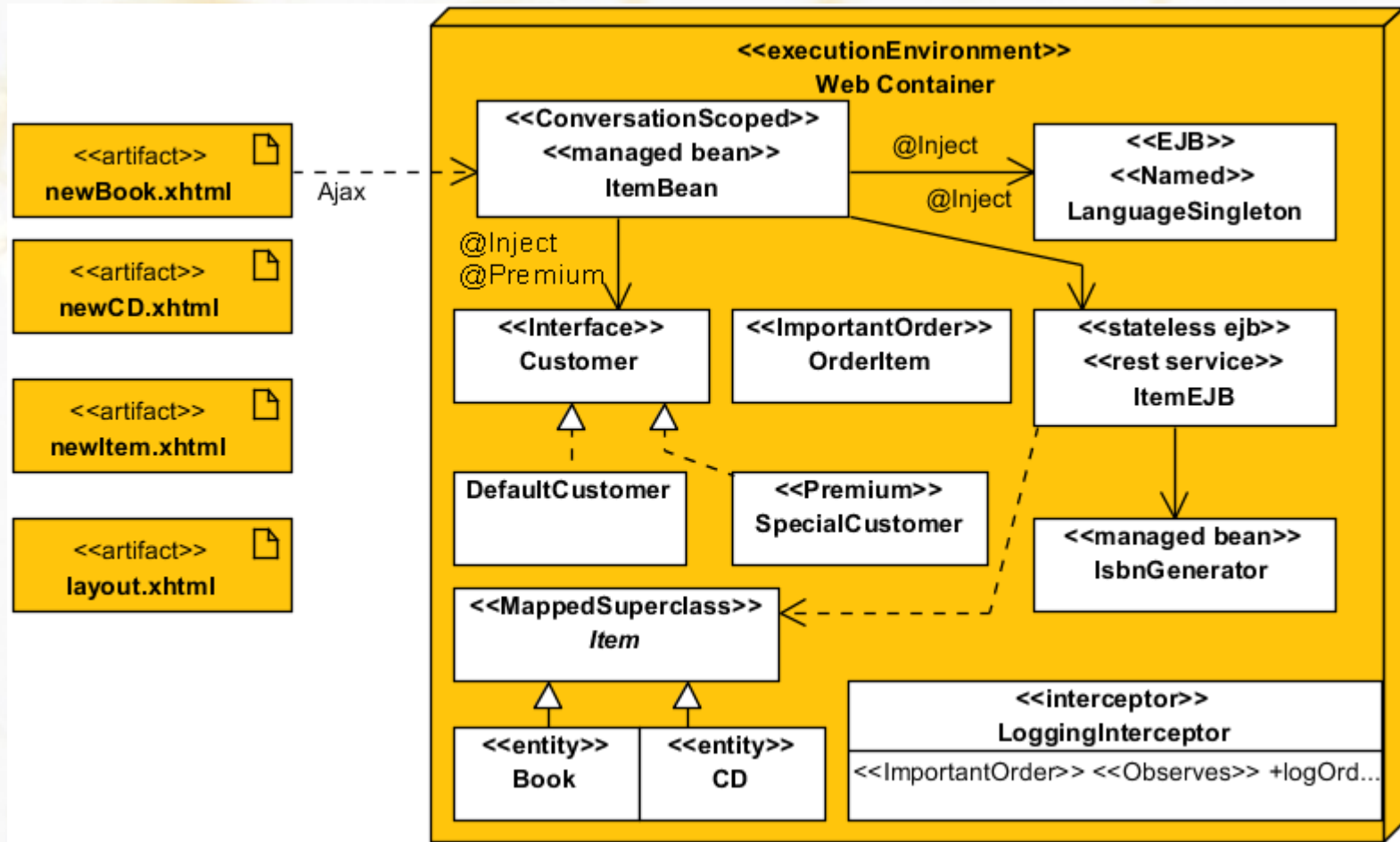
```
@Inject @ImportantOrder  
Event<OrderItem> order;
```

- Easily caught :

```
void logOrders(@Observes  
@ImportantOrder OrderItem order) {
```

# Demo 15

Use CDI conversation scope (and events)



# And more...

- Part of Web Profile
- Easy to start with : @Inject
- Grow as you require
- What we haven't covered:
  - Producers/Consumers, Decorator, portable extensions
  - Best practices

# Reality check

Java EE 6  
Adoption



GlassFish Server

**ORACLE**  
WebLogic

**TmaxSoft**  
TmaxSoft Co., Ltd.



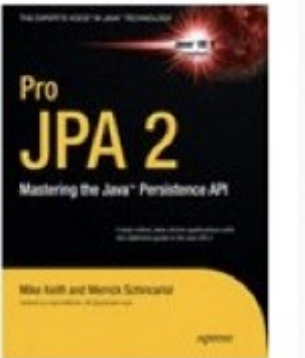
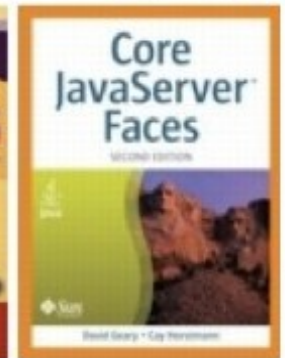
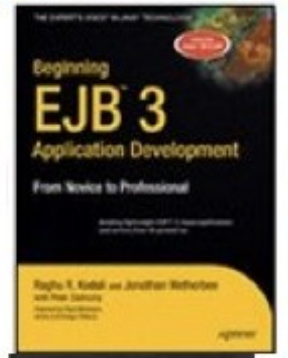
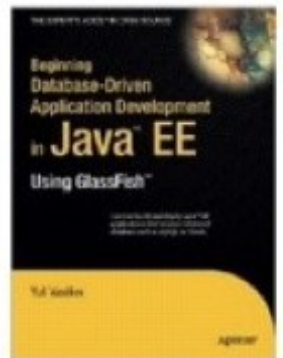
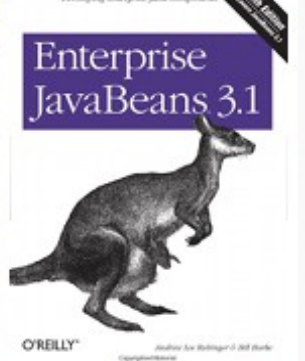
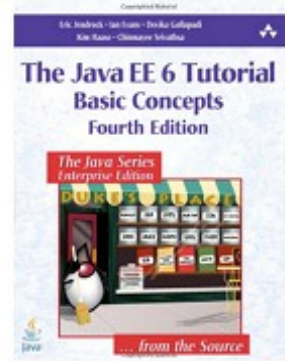
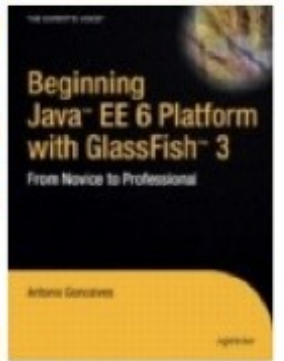
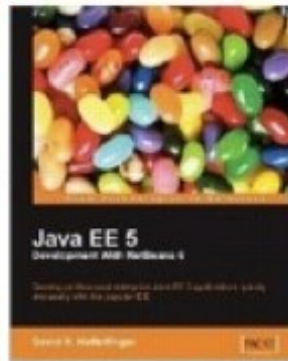
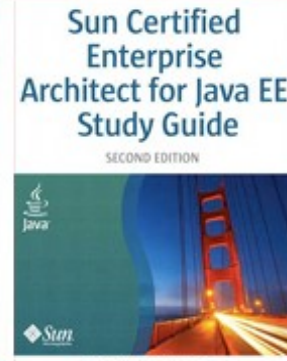
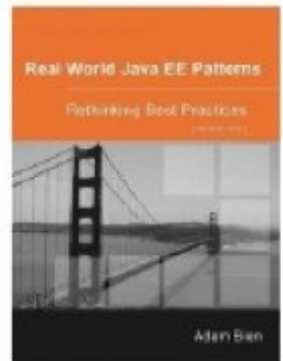
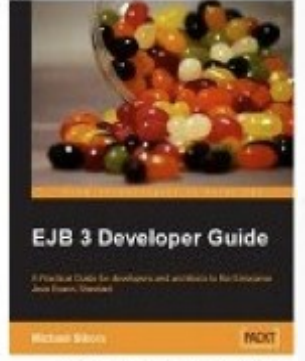
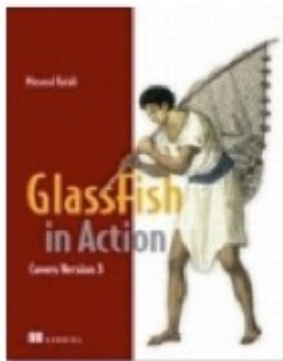
**caucho**



APACHE  
**GERONIMO**

...





# 9 Reasons Why Java EE 6 Will Save Real Money

Submitted by adam bien on Fri, 2009/12/25 - 1:00am



**TODAY**  
on *TheServerSide.COM*

Share info our members need-to-know!

**SUBMIT STORY**

## Moving from Spring to Java EE 6: The Age of Frameworks is Over

Cameron McKenzie | 4 OCT 2010 | 0 Comments

Frameworks solved the J2EE problem. But with Java EE 6, it's that your frustration with J2EE created. Frameworks are a thing of the past, extensions are the future.

*Developers think they can double the productivity of Java EE 6. 200%, not orders of magnitude »*

*« Java EE 6 provides many promising features that will truly make enterprise application development simple and easier. » - DevX.com*

OcpSoft Home PrettyFaces PrettyTime

**39** ZONE IT  
**0** DZone

### Spring to Java EE - A Migration Experience

October 1st, 2010 by Lincoln

So Java EE 6 is out, and you've decided to give it a go. You're trying to migrate to a new stack (or are trying to create a new one for the first time.) but e

**InfoQ** update

501,897 Sep unique visitors



Register

Tracking change and innovation in the enterprise software

News

## Java EE6: EJB3.1 Is a Competitor

Posted by Josh Long on Feb 15, 2010

Community [Java](#) Topics [JCP Standards](#) Tags [Java EE](#)

Share

# Glen Smith

Java, XML and all that Jazz... from Canberra

OCT 05 2010

"When Elephants Dance - Why Java EE6 is not your Grandma's Enterprise Stack"

« Frameworks like Spring are really just a bridge between the mistakes of the J2EE past and the success of the Java EE 6 future.

Frameworks are out, and extensions to the Java EE 6 platform are in. »



Gotchas

# Lightweight Java EE Component

```
@javax.annotations.ManagedBean  
public class ItemBean {  
    ...  
}
```

```
@Resource  
private ItemBean item;
```

# JSF Controller

```
@javax.faces.ManagedBean("item")  
public class ItemBean {  
    ...  
}
```

```
    @Resource  
    private ItemBean item;
```

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

# CDI-style

```
@javax.inject.Named("item")
@javax.enterprise.context.RequestScoped
public class ItemBean {
    ...
}

@Inject @Premium
private ItemBean item;

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

# CDI Stereotype

```
@javax.enterprise.inject.Model  
public class ItemBean {  
    ... // request-scoped logic  
}
```

```
    @Inject @Premium  
    private ItemBean item;
```

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

# EJB injection point?

```
@javax.ejb.EJB myEJB;
```

```
@javax.inject.Inject myEJB;
```

# EJB injection point?

```
@javax.ejb.EJB myEJB;
```

```
@javax.inject.Inject myEJB;
```

carries EJB semantic,  
direct reference

requires CDI,  
uses a proxy

# Scopes?

@javax.faces.bean.RequestScoped vs.  
@javax.enterprise.context.RequestScoped

@javax.faces.bean.SessionScoped vs.  
@javax.enterprise.context.SessionScoped

@javax.faces.bean.ApplicationScoped vs.  
@javax.enterprise.context.ApplicationScoped

# If CDI is present (Java EE 6)

@javax.faces.bean.RequestScoped vs.  
@javax.enterprise.context.RequestScoped

@javax.faces.bean.SessionScoped vs.  
@javax.enterprise.context.SessionScoped

@javax.faces.bean.ApplicationScoped vs.  
@javax.enterprise.context.ApplicationScoped

*JSF 2.0 runs on servlet 2.5  
@ViewScoped not in CDI*

# Singleton?

- `@javax.ejb.Singleton`
  - EJB 3.1, thread-safe and transactional
- `@javax.inject.Singleton`
  - JSR 330, a type that the injector only instantiates once

# Assuming Java EE

- `@javax.ejb.Singleton`
  - EJB 3.1, thread-safe and transactional
- `@javax.inject.Singleton`
  - JSR 330, a type that the injector only instantiates once

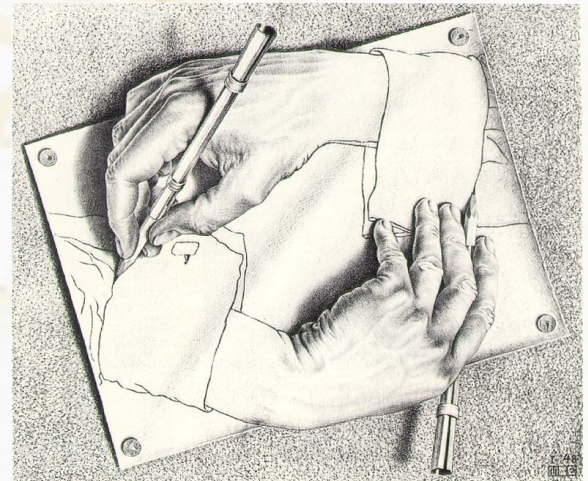
*Assuming managed JavaEE environment  
Neither provides a cluster-wide singleton*

# Java EE 6

- Breadth of technology
  - Enhanced features (EJB, Servlet, JPA, JSF)
  - New features (JAX-RS, CDI, BeanValidation)
  - Strong focus on EoD
- Breadth of ecosystem
  - Implementations
  - Tools
  - Testing

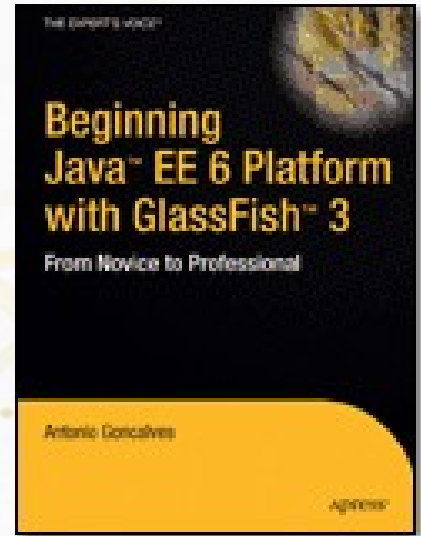
# Hands on Lab @ 13:30 / BOF 2

- Getting your hands on Java EE 6
- BYOL (Bring Your Own Laptop)
  - Software requirements: NetBeans 6.9.1 (with GlassFish 3.0.1), Maven, cURL or wget, that's it.
  - Instructions + Code handed during lab
- Self-paced (not instructor-led)
  - But feel free to ask questions
  - Three exercises:
    - JSF 2.0
    - JAX-RS 1.1
    - CDI 1.0



# Book signing session

- 450 pages about Java EE 6
- Second edition
- Covers most specs
- Come and get it signed
- ... sometime during Devovx



See you next year ?  
Java EE 7 ?

---

@agoncal & @alexismp  
<http://beginningee6.kenai.com>