October, 18 2011

The Java EE Platform

An enterprise dedicated platform

Jérôme Blanchard
Specialized Engineer
SED, INRIA Nancy Grand Est
Jerome.Blanchard@inria.fr

Presentation organisation

The presentation is composed of 6 parts !! BUT with a small break in the middle ;-)
Each part is composed of some slide and some live coding.

Part 1 . 10' . Overview
Part 2 . 20' . Persistence
Part 3 . 30' . Business logic

Part 4 . 20' . Presentation
Part 5 . 20' . Interoperability
Part 6 . 10' . Questions & Answers

PART 1 : Overview

Java EE, an enterprise platform

Java Enterprise Edition

Java EE is a standard platform for creating application :

- Transactional
- Secure
- Interoperable
- Distributed
- Robust
- Powerful
- Highly available

Java EE is a set of specification that allows

- Develop once, deploy anywhere...
- Use your own container implementation

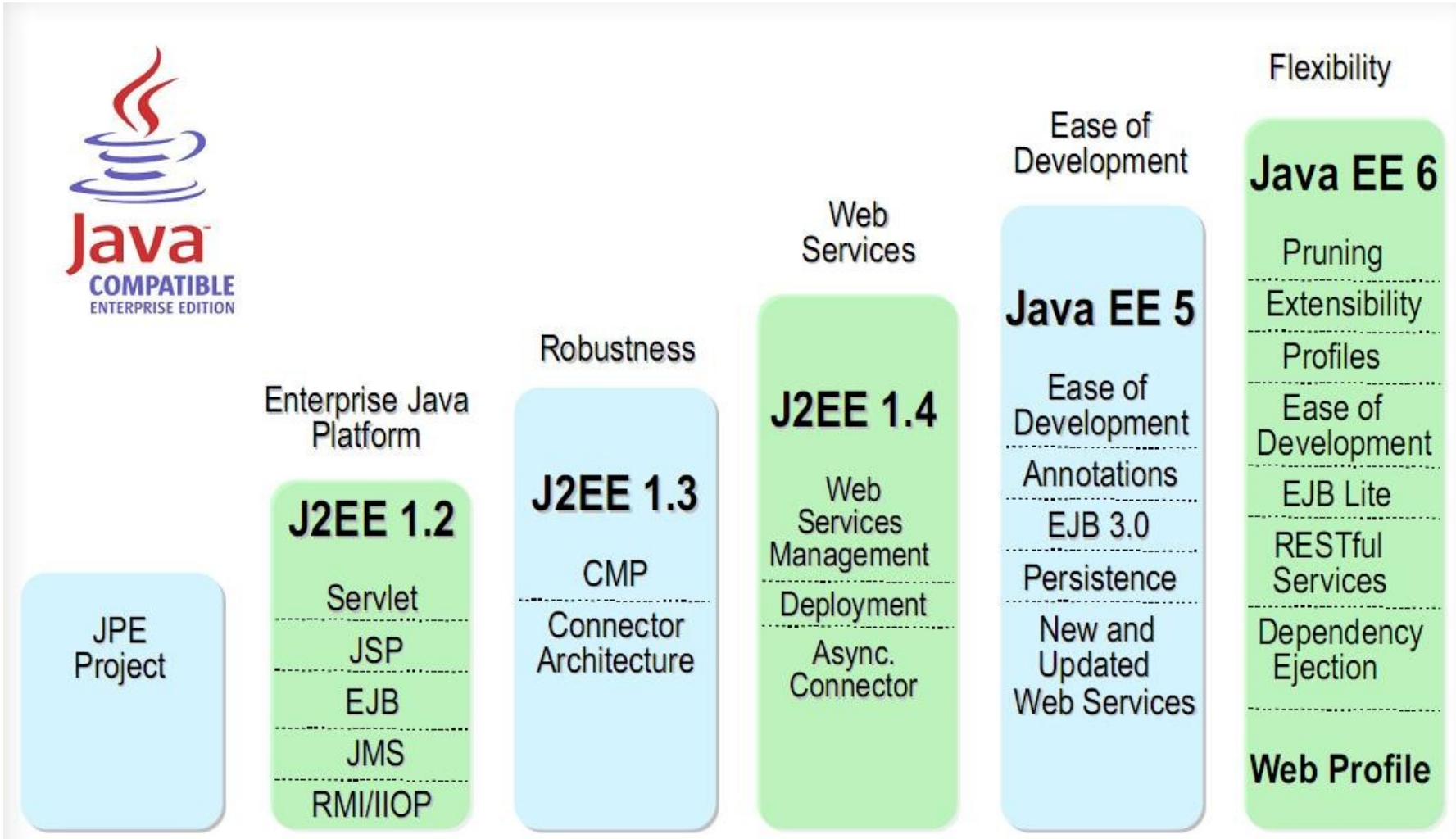
Some Java EE Metrics

How popular is JEE ?

- 10 years of history
- 10 millions downloads of Java EE annually
- Amazon search for J2EE yields more than 4000 books
- Monster lists over 2300 Java EE / J2EE jobs
- Jboss purchased by Red Hat for $300+ million
- BEA purchased by Oracle for $7 Billion
- Application Server market at $2.5B+
- Servlet containers are standard for Web apps
- JMS is an industry standard for messaging

An 10 years old story

Java EE from past to present



What is Java EE ?

Technology Content

Java EE defines containers that provide some services to hosted components

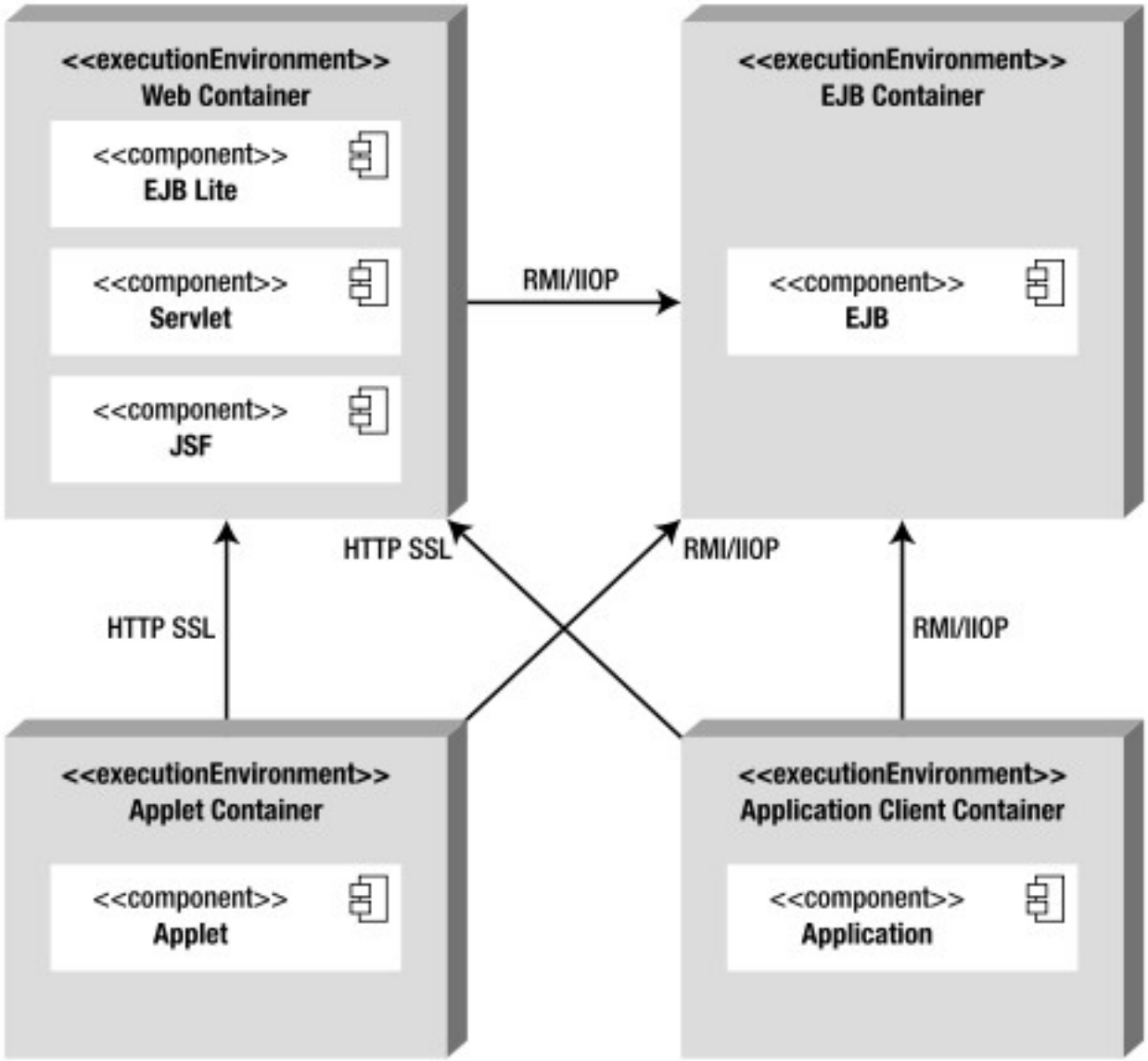
Components use well defined contracts to communicate with containers

Components need to be packaged in a standard way to be deployed

Java EE defines 4 types of components : applets, applications, web applications, enterprise applications

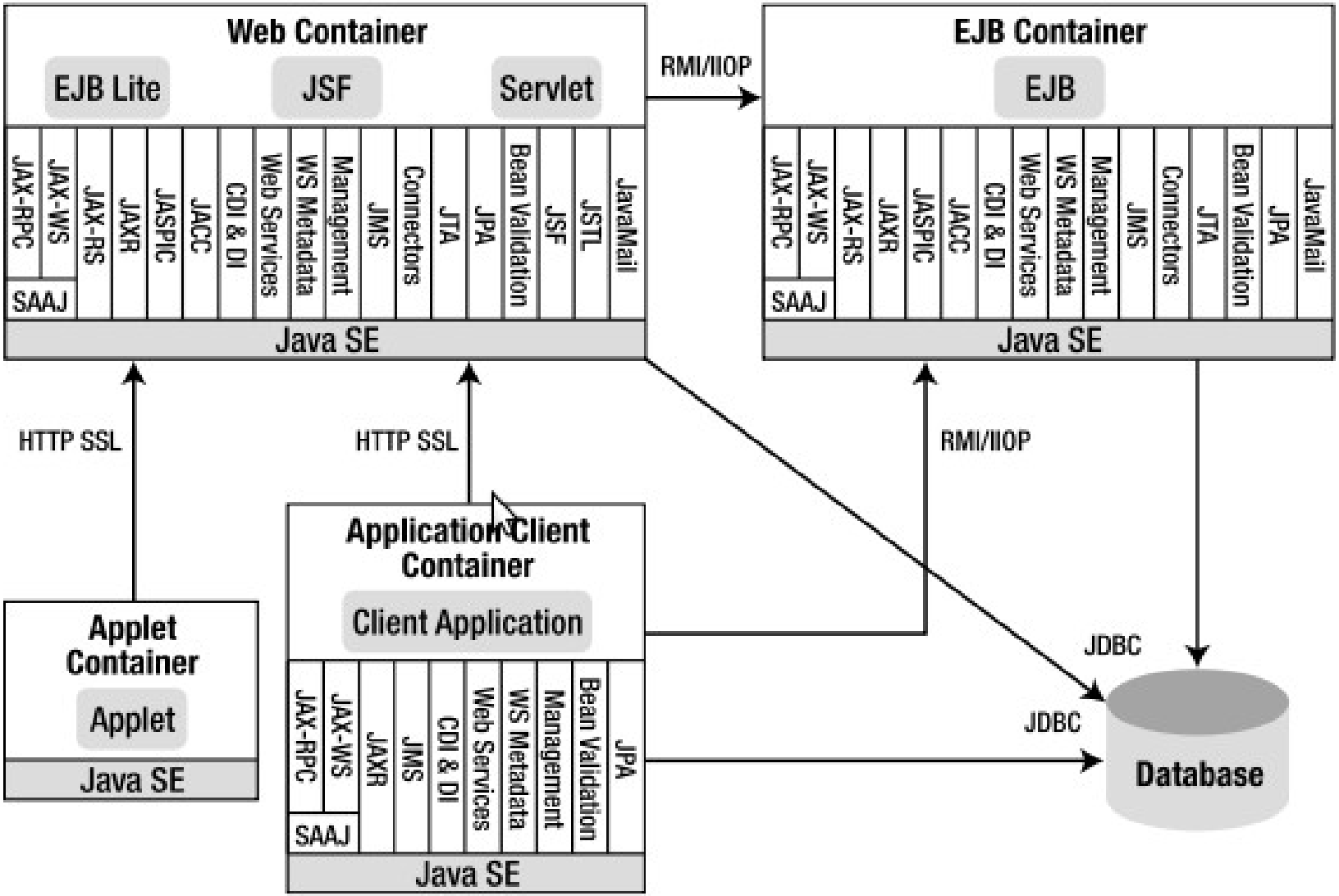
What is Java EE ?

Execution environments



What is Java EE ?

Execution environments



Presentation Tutorial

GOAL

The goal is to develop a clone of transfert.inria.fr.

This application is a simple file sharing plateform which avoid file size limitation of some mailbox providers. The principe is to upload a file that is going to be available online for a time limited period. The online service will notify users of this file availablility by sending an email with the download link.

We'll try to include maximum aspects of JEE6 plateform technology in this tutorial.

Presentation Tutorial

Environment

The following tools and softwares are going to be used to create this application :

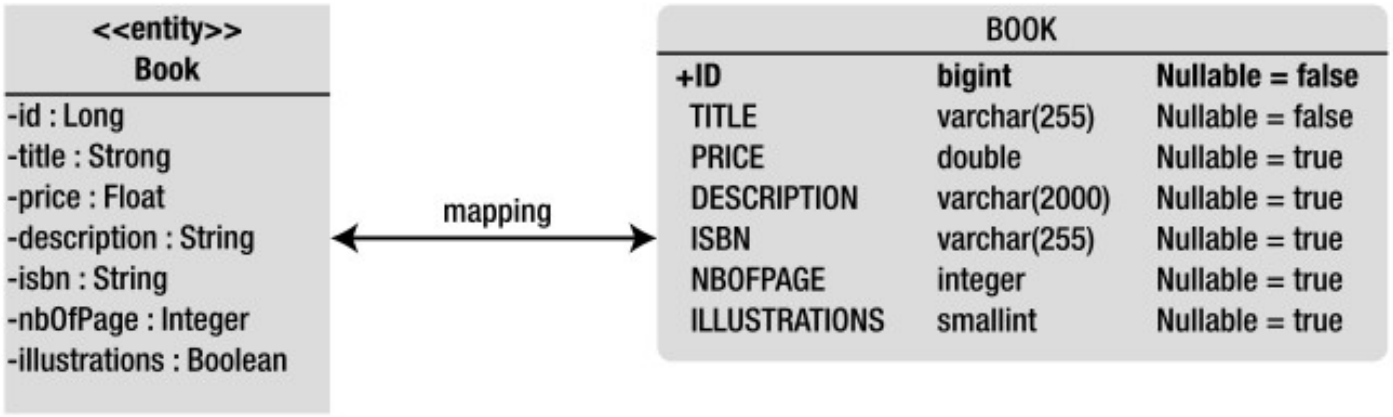
JDK >= 1.6
Maven
Glassfish >= 3.1
Derby
Eclipse
JUnit

PART 2 : Persistence

Persistence in JEE

Basics of Object-Relational Mapping

- The goal of ORM is to delegate the task of creating the correspondance between an object and a table.
- The object paradigm is mapped to relationnal database.
- The programmer use program objects instead of tables.
- A dedicated query language allows retreiving object or collections of objects from storage.



Persistence in JEE

The Java Persistence API

Java Persistence API (JPA) is an abstraction above JDBC
All classes and annotations are in the `javax.persistence` package
The `EntityManager` handle database operations (CRUD)
A language is designed for querying : JPQL
Transactions and Locking are provided by JTA
Callbacks and Listeners allows hooking the life cycle of a persistent object

Persistence in JEE

Object-Relational Mapping using JPA

JPA map java objects to database using metadata (annotations)

JPA use the concept of configuration by exception :

- Some default rules are applyed by default
- You set specific metadata only to change default mapping

```
@Entity
public class Profile {
    @Id
    private String id;
    @Column(nullable = false)
    private String email;
    private String fullname;
    // (...)
}
```

Persistence in JEE

How to query entities ?

JPA provide the Entity Manager interface

JPQL queries are similar to SQL but using object field names :

```
SELECT p FROM Profile p WHERE p.id = "jayblanc"
```

JPQL statement can be executed with dynamic queries, static queries or with a native SQL statement.

Static queries (named queries) are defined using annotations :

```
@NamedQuery(name = "findProfileByEmail", query =
"SELECT p FROM Profile p WHERE p.email =
:email")
```

Persistence in JEE

The persistence unit

EntityManager is either injected by container nor created by hand but ni all cases you must define the persistence unit in a dedicated configuration file : persistence.xml
In containers, persistence unit can use a datasource (a kind of preconfigured database access)

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"  version="1.0">
   <persistence-unit name="file-exchange-unit">
      <jta-data-source>java:/DefaultDS</jta-data-source>
   </persistence-unit>
</persistence>
```

Persistence in JEE

ORM Relationship Mapping

Different cardinality :

@OneToOne, @OneToMany, @ManyToOne, @ManyToMany

And direction : unidirectional or bidirectional.

The simplest case, 1 to 1 unidirectional :

```
@Entity
public class File {
    @Id private String id ;
    @OneToOne private Profile owner;
    (…)
}
```

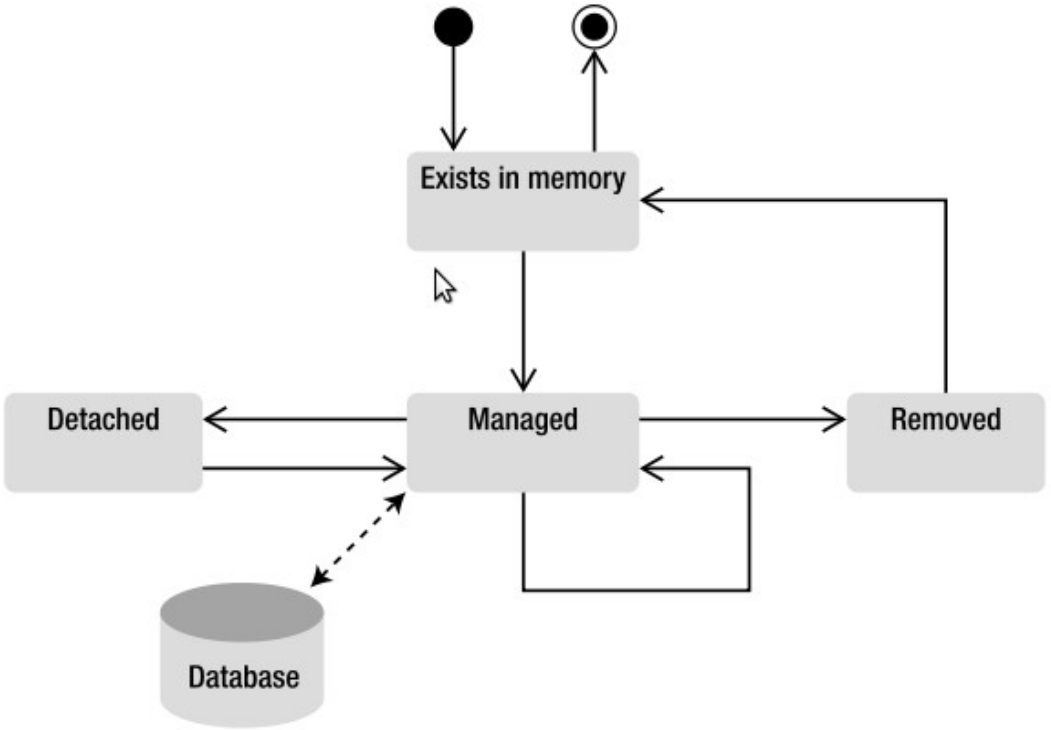
Persistence in JEE

Managing persistent objects

LifeCycle of a persistent object

Entity Manager operations :

- find(Class, pKey)
- persist(Object)
- merge(Object)
- remove(Object)
- detach(Object)
- contains(Object)
- flush()
- refresh(Object)
- clear()



Persistence in JEE

Querying the persistence : JPQL

Mostly like SQL adapted to object paradigm
Binding parameters with :param syntax

```
Query query = em.createQuery("SELECT p FROM Profile p WHERE p.id =
:id", Profile.class) ;
query.setParameter("id", "jayblanc");
List<Profile> profiles = query.getResultList();
```

@NamedQuery are defined in the Entity object itself

```
@NamedQuery(name="findByEmail", query="SELECT p FROM Profile p WHERE
p.email = :email")

Query query = em.createNamedQuery("findByEmail").setParameter("email",
"jayblanc@free.fr").setMaxResult(3);
List<Profile> profiles = query.getResultList();
```

Persistence in JEE

Concurrency & Versionning

JPA use the locking mechanism to ensure transaction isolation
JPA 2.0 allows usage of PESSIMISTIC and OPTIMISTIC locking

The EntityManager.lock() operation allows you to set the locking mechanism you want to put on a specific managed object.

@Version to detect concurrency problems in optimistic locking

Pessimistic locking is very ressource consuming because the lock is acquired on the DB when loading the object (SELECT FOR UPDATE)

Persistence in JEE

Summary

In this part we've seen that we can :

- define Java classes as Entities
- specify sql to object mapping using only annotations
- query entities using a SQL like syntax
- ensure transactional isolation of access

Lots of things are predefined when you use your entities in a container managed environment :

- EntityManager is injected
- Transaction and Isolation are managed by the container

PART 3 : Business Logic

Enterprise Java Beans

How to give the best of your business logic

- Persistence layer does not handle complex logic and components interactions
- EJB are server side components that takes care of Transaction and Security
- EJB have an integrated stack for messaging, scheduling, remote access, web service endpoints, dependency injection, component lifecycle, AOP with interceptors.
- EJB integrates with many Java SE and Java EE technologies :
  - JMS, JAAS, JDBC, JavaMail, JPA, JTA, JNDI, RMI
- EJB are the perfect solution for business logic implementation

Enterprise Java Beans

Why EJB a good choice of implementation

- EJB follow component oriented architecture
- Today EJB is as simple as annotating a POJO
- EJB are deployed in an EJB container
- The container provides services like transaction management, security authorization, pooling, concurrency control.
- Some servers gives also clustering, load balancing and failover
- EJB : write once, deploy everywhere

Enterprise Java Beans

EJB anatomy

An EJB is as simple as annotating a POJO :

```
@Stateless
public class ProfileService {

    @PersistenceContext(unitName="filexchange-pu")
    private EntityManager em;

    public boolean exists(String id) {
        Profile p = em.find(Profile.class, id);
        if ( p == null ) {
            return false;
        }
        return true;
    }
}
```

Enterprise Java Beans

EJB Types

There is 3 types of EJB :

- Stateless, Statefull and Singleton

Once used by a client thread, @Stateless annotated Beans instances are returned to the pool instead of being destroyed

@Statefull Beans are able to keep bean properties values during a defined amout of time

@Singleton Bean have only one instance per application
@Startup annotation ensure instanciation at deployement time

Enterprise Java Beans

EJB interfaces

EJB allows you to define @Local and @Remote interfaces

@Local interface :

- Only visible in application server
- Method parameters are passed by reference

@Remote interface :

- Visible outside application server throught RMI
- Method parameter are passed by value

No Interface :

- Same as local but with no interface

Enterprise Java Beans

EJB Client Side

Client can invoke an EJB from anywhere :

Another EJB, a JSP, a Servlet, a swing application, …

The client never instanciate the Bean itself but get a reference on it via dependency injection or via a JNDI lookup

Client call are often synchronous but now EJB 3.1 allows asynchronous calls which allows the client to get back control. A Futur object will be use for the client callback.

Enterprise Java Beans

EJB Session Context

The EJB container maintains a SessionContext

SessionContext interface is use to interact with container internal services like transaction or authorization

SessionContext.setRollbackOnly()
SessionContext.getCallerPrincipal()
SessionContext.isCallerInRole()

Enterprise Java Beans

EJB Timer Service

The EJB Timer Service acts like a CRON

Timer Service can be use automatically
@Schedule(dayOfMonth = "1", hour = "5", minute = "30")

Or programmatically
TimerService.createTimer() and a @Timeout annotation

Enterprise Java Beans

EJB Callback and Interceptors

As for Entities, you can have some callback control on Session Beans using annotations like @PostConstruct or @PreDestroy

Statefull Beans have more callbacks : @PrePassivate @PostActivate

If you are familiar with AOP, you'll find a AOP-like functionnality in EJB using @Interceptors and @AroundInvoke annotations in order to apply cross-cutting concerns from your business code.

Enterprise Java Beans

EJB Transactions

Transactions are often managed by the container via JTA.
We distinguish 2 types of transactions : local and XA.
XA transactions ensure ACID properties accross distributed resources using two-phase commit via JTS.

In EJB, Transactions are managed by container (CMT) and you can set some attributes to ensure container behaviour : REQUIRED, REQUIRES_NEW, SUPPORTS, MANDATORY, …
Using the session context, you can mark the transaction for rollback (SessionContext.setRollbackOnly())

In some case, you can use Bean Managed Transaction in order to do everything programmatically.

Enterprise Java Beans

EJB Security

In JEE as in other systems, security is based on Principal and Roles. A principal is an authenticated user that can have some specific roles.

Authentication and authorization is done using JAAS.

Some annotations allows developers to set declarative security in EJB : @PermitAll, @DenyAll, @RolesAllowed, @DeclareRoles, @RunAs

You can also set your own security programmatically using the SessionContext and the getCallerPrincipal() and isCallerInRole() methods.

PART 4 : Presentation

Presentation with JSP, Servlets, JSF

How to open business logic to users

JEE presentation layer is often handle using Web pages and HTML but also Swing applications.

JEE provides two major technologies to produce Web interfaces : Servlet/JSP & JSF

Both are allowing to deal with HttpRequest in order to produce dynamic content depending on the submitted informations.

More than Servlet/JSP, JSF is a technology that brings interface to the web inspired by graphical component models.

Presentation with JSP, Servlets, JSF

Even if Servlet/JSP seems to be pretty old technology, it's widely used and allows to perform most of the job.

A Servlet is a java class used to extend the capabilities of applications hosted by request-response based servers.

Servlets are commonly used to respond to HttpRequest and in this case extends the HttpServlet class.

Servlets are deployed in a web container.

JSP are higher level servlet that allows a html like syntax.

Presentation with JSP, Servlets, JSF

Servlet Lifecycle

The container controls the lifecycle of the servlet (loading class, instanciation, initialisation, invocation of service() method)

Listener class allows you to listen this lifecycle.
Servlet are NOT thread safe and can be accessed concurrently

@WebServlet annotation is used to declare a servlet and binds it to a url pattern

ServletRequest (or HttpServletResquest) interface allows you to deal with data passed from client to the servlet like parameters, or between servlets like object oriented attributes.

Presentation with JSP, Servlets, JSF

ServletResponse contains data passed from servlet to client. The interface gives you control on output stream, content type.

Filter acts like an Interceptor and allows you to apply transversal logic or treatment (template), security control (block)

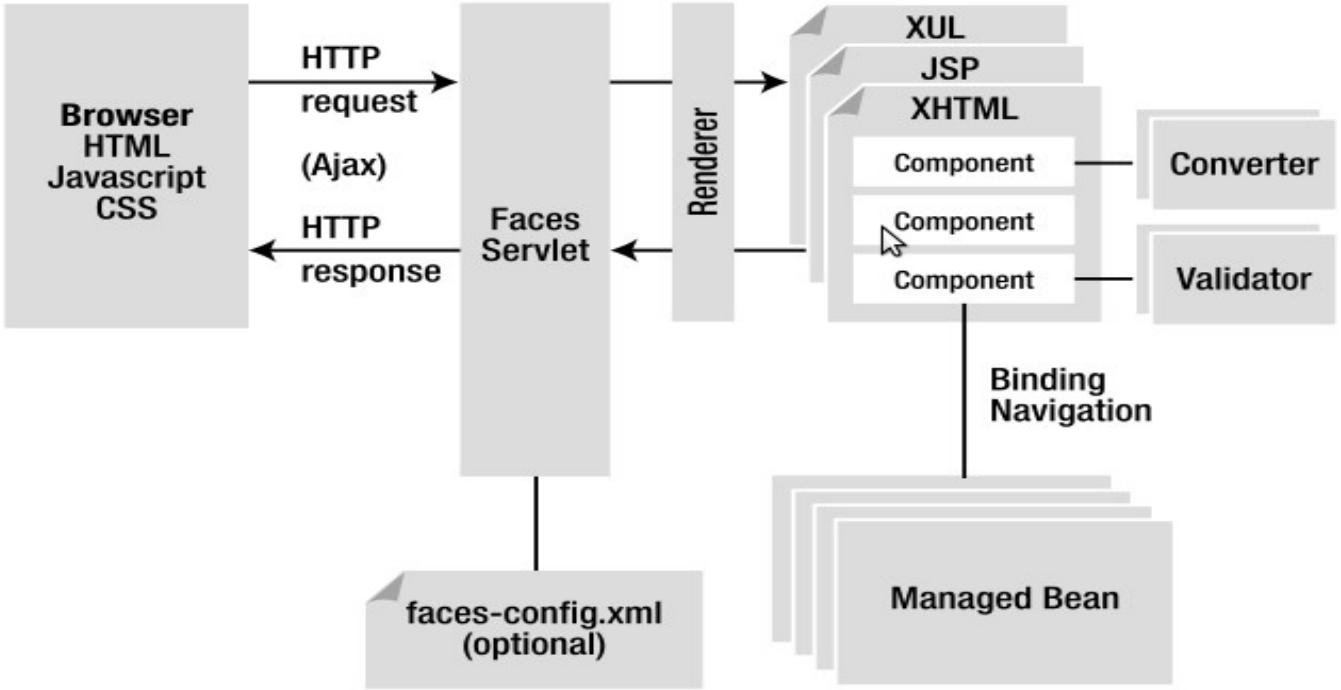
Filters are applyed like a Chain of Responsability using a FilterMapping pattern.

Client State is maintained using a Session object.

Presentation with JSP, Servlets, JSF

Java Server Face

JSF application includes adding components to the page in order to add managed beans, validators, converters, and other server-side object associated with the page.



Presentation with JSP, Servlets, JSF

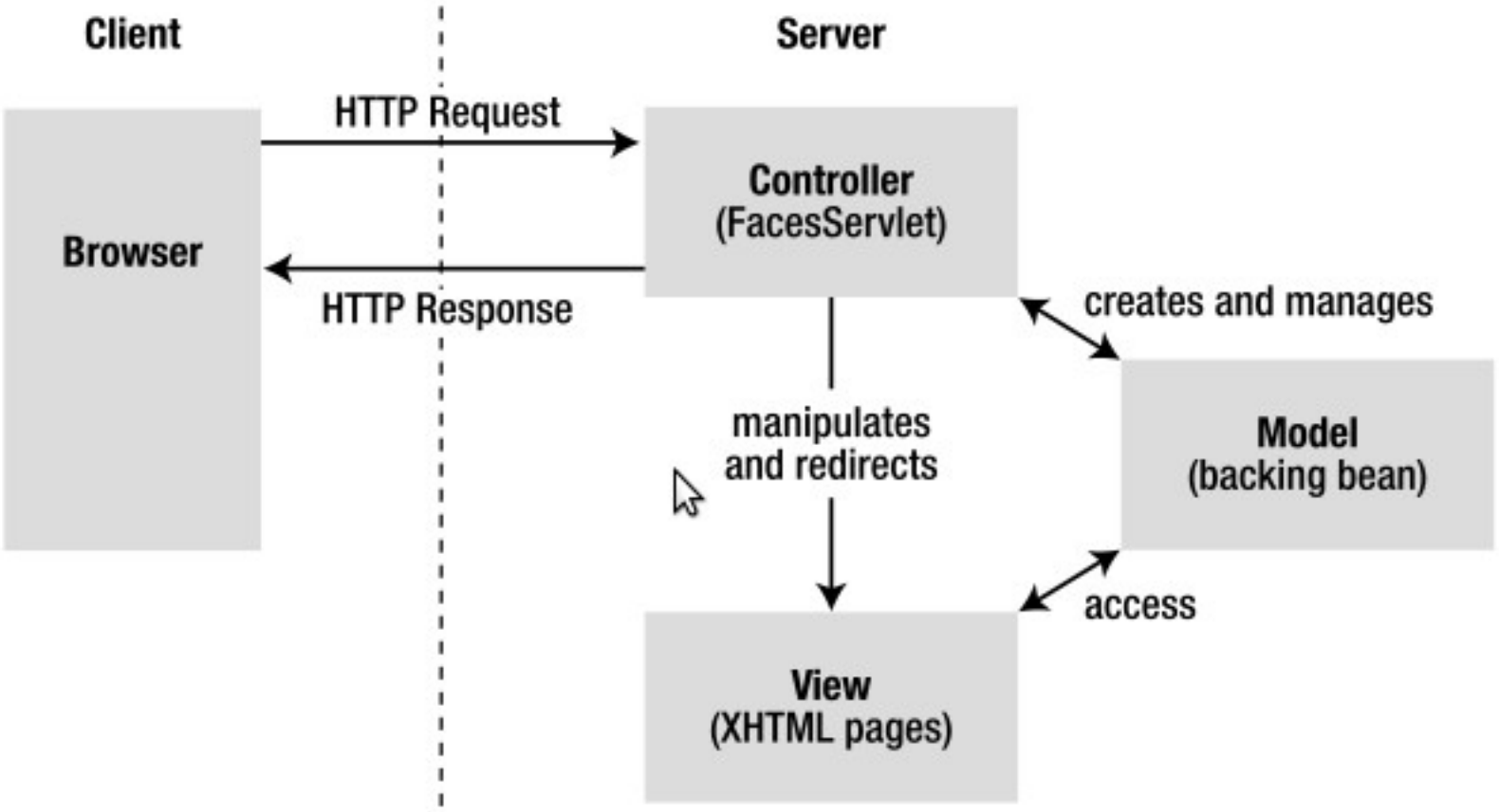
Java Server Face

JSF uses MVC design pattern :

- **FacesServlet is the main servlet for the application and acts as the controller.**
- **Pages and Components uses PDL (page declaration language) for rendering in several technology. PDL used in JSF are either JSP of Facelets.**
- **Converters and Validators components are used to validate user information before any back end treatment**
- **ManagedBean and Navigation components are handling application business logic and navigation accross pages.**

Presentation with JSP, Servlets, JSF

Java Server Face



Presentation with JSP, Servlets, JSF

PDL technologies

- XHTML / CSS / Javascript
- JSP / JSTL / ExpressionLanguage
  - EL simplifies variables output and references

```
${profile.id}
```

  - JSTL allow avoiding mixing java code and XHTML

```
<sql:query var="books">select * from book</sql:query>
<c:if test="${fn:length('H2G2') == 4}">yop</c:if>
```

- Facelets

Presentation with JSP, Servlets, JSF

ManagedBean

Following the MVC pattern, ManagedBean are the heart of JSF and are used to process business logic, call EJBs, databases as well as navigating between pages.

Managed Bean have a scope and a lifecycle and exposes methods and properties that are bound to UI using EL.

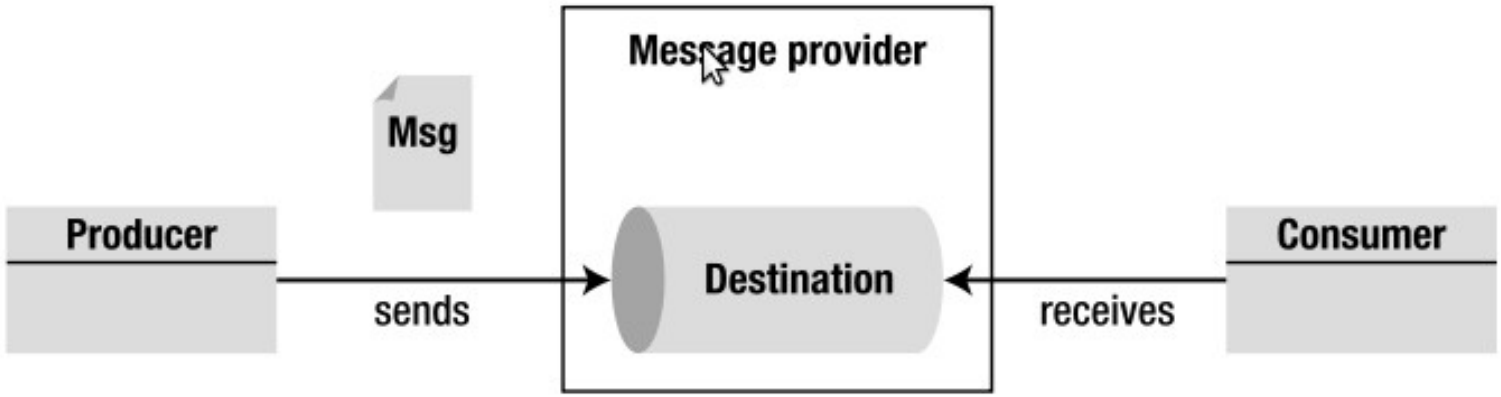
Now JSF2.0 support integrated AJAX technologies to invoke managed bean asynchronously.

PART 5 : Interoperability

Messages & Web Services

How to open business logic to other systems

- In order to allows other applications to deal with your EIS, JEE provides you some interoperability technologies that ease things : Web Services (SOAP & REST) and JMS
- Messaging Oriented Middleware (MOM) enable asynchronous messages between heterogeneous systems.
- MOM are loosely coupled : sender don't know who is going to receive the message. In fact sender and receiver knows nothing each other.



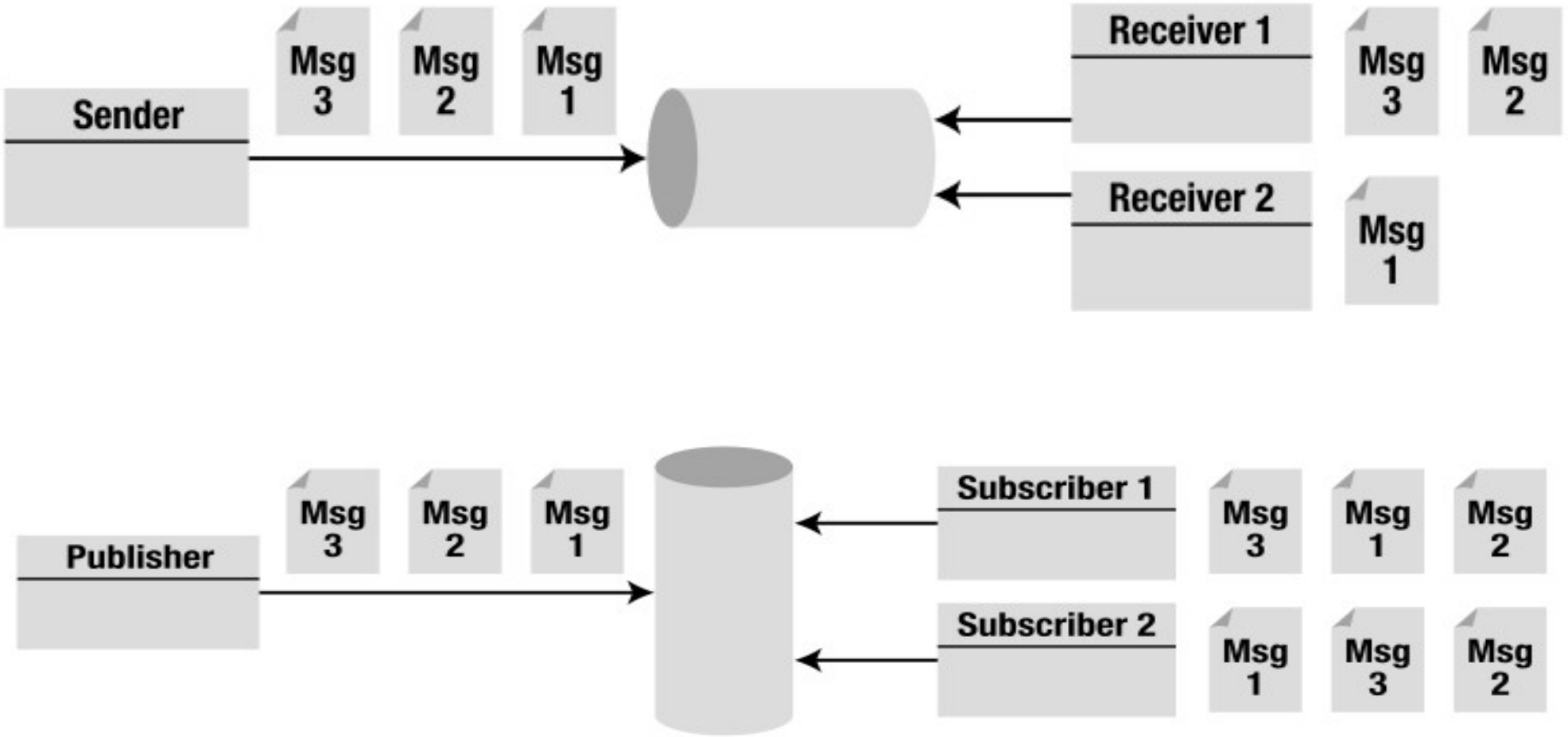
Messages & Web Services

Java Messaging Service

- JMS is a standard API that allows you to deal with messages asynchronously (create, send, receive and read)
- JMS can connect with several providers (OpenMQ, ActiveMQ)
- JMS defines some components :
  - Message producers
  - Message consumers
  - Messages themselves
  - Connections and Destinations
- MessageDrivenBean (MDB) are asynchronous message consumers that are executed in an EJB container
- JMS is either P2P (Queue) or publish/subscribe (Topic)

Messages & Web Services

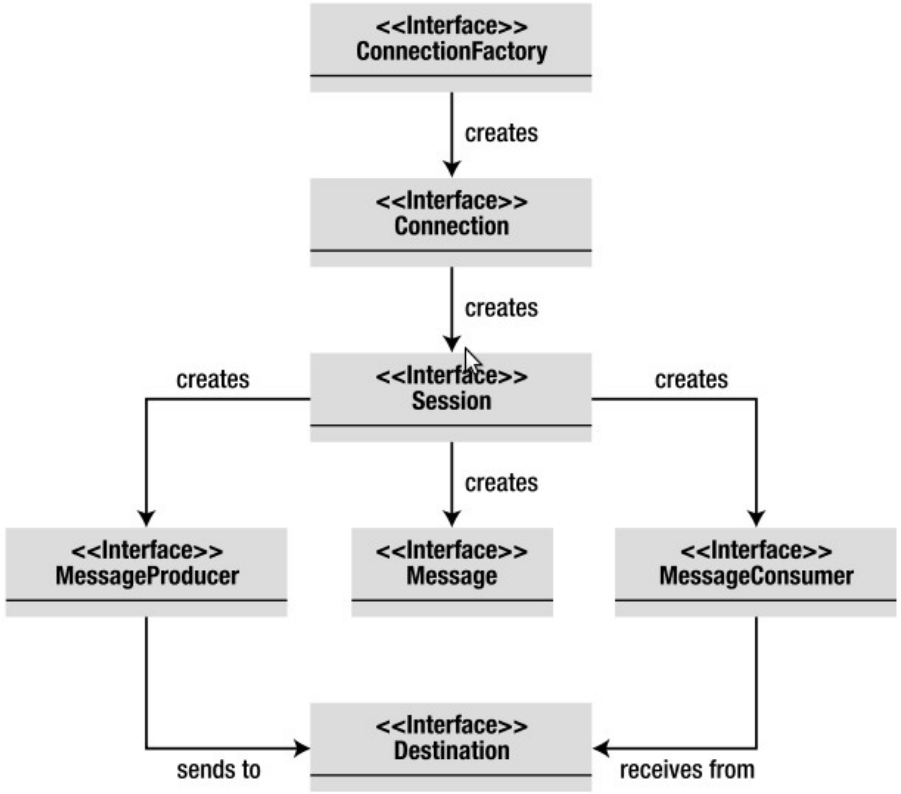
Java Messaging Service



Messages & Web Services

Java Messaging Service

Container manages ConnectionFactory and Destinations, both are injected by the container but have to be declared in the JNDI



Messages & Web Services

SOAP Web Service

SOAP Web Services is a Distributed Application technology that allows remote method invocation throught heterogenous systems.

@WebService annotations declares a POJO as a SOAP WebService endpoint

JAX-WS and JAX-B are used to handle java to xml marshalling and unmarshalling.

JAX-RS allows to build RESTfull Web service using @Path(/resources) annotation syntax.

Credits & References

Some images have been taken from book :
Apress - Beginning Java EE 6 platform with Glassfish 3
Antonio Goncalves

Lots of informations availables in the Java EE6 tutorial
http://download.oracle.com/javaee/6/tutorial/doc/

Source Code for Tutorial available at
https://bitbucket.org/jayblanc/jee6-screencast

Slides & Source Code Development Screencast available at :
http://wiki.jayblanc.fr/wiki/index.php?title=Main_Page

Question & Answers

Thank you