

Java EE 5, 6 et les EJBs 3.1

Antonio Goncalves

Tours JUG – le 11/06/08

Agenda

- Présentation
- Java EE 5
- Java EE 5 vs J2EE 1.4
- Java EE 6
- EJB 3.1
 - Encore plus facile
 - Nouvelles fonctionnalités
- Questions



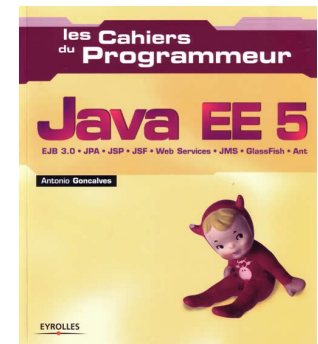
Agenda

- Présentation
- Java EE 5
- Java EE 5 vs J2EE 1.4
- Java EE 6
- EJB 3.1
 - Encore plus facile
 - Nouvelles fonctionnalités
- Questions



Antonio Goncalves

- Consultant / **Architecte** senior
- Spécialiste Java / Java EE
- Ancien de **BEA Systems**
- Enseignant à l'université du **CNAM**
- Expert Group
 - JSR 316 (Java EE 6)
 - JSR 317 (JPA 2.0)
 - JSR 318 (EJB 3.1)
- Auteur Java EE 5
- Co-createur du Paris JUG



Et vous ?

- J2EE 1.4
- Java EE 5
- Spring / Hibernate / Struts
- Java EE 6

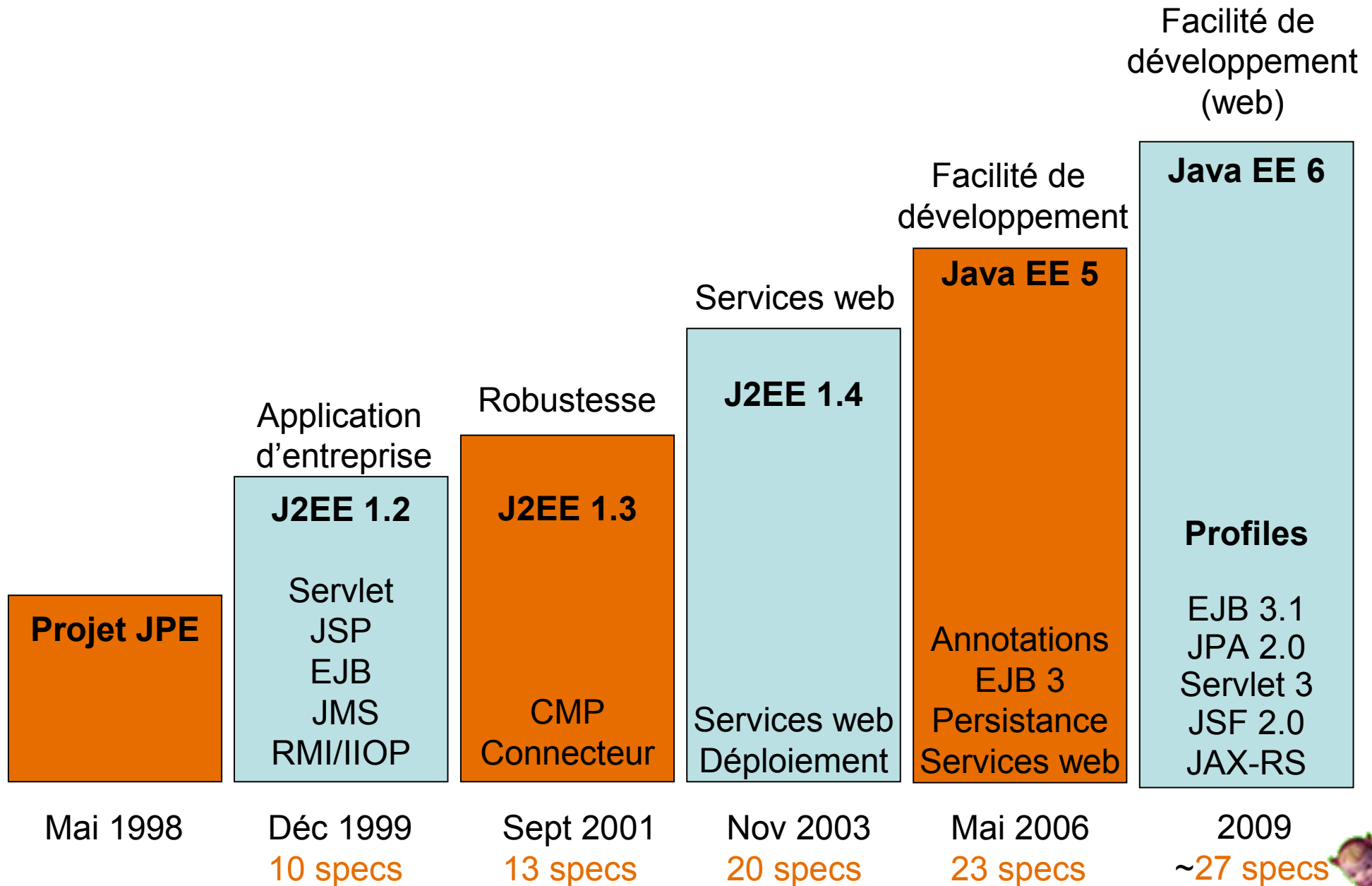


Agenda

- Présentation
- Java EE 5
- Java EE 5 vs J2EE 1.4
- Java EE 6
- EJB 3.1
 - Encore plus facile
 - Nouvelles fonctionnalités
- Questions



Historique de Java EE



A propos de Java EE 5

- Développé sous la JSR 244
- Sortie en Mai 2006 (+ GlassFish V1)
- **Facilité de développement**
 - EoD (Ease of Development)
- Compatible avec J2EE 1.4
- Le meilleur des deux mondes
 - Spécifications
 - Inspiré de l'Open source



Les grandes nouveautés

- Rupture par rapport à J2EE 1.4
- Simplification des EJB et WebServices
- Nouvelle API de persistance (JPA)
- Model de développement POJO
- Injection de dépendance
- Utilisation des annotations
- Descripteurs XML optionnel
- Développement par exception
- Facilite les applications web avec JSF



Contenu Java EE 5

- 23 specifications
 - **Web** : **JSF 1.2**, JSP 2.1, JSTL 1.2, Servlet 2.5
 - **Enterprise** : Common Annotations 1.0, **EJB 3.0**, JAF 1.1, JavaMail 1.4, JCA 1.5, JMS 1.1, **JPA 1.0**, JTA 1.1
 - **Web Services** : JAX-RPC 1.1, JAX-WS 2.0, JAXB 2.0, SAAJ 1.0, StAX 1.0, **Web Services 1.2**, Web Services Metadata 2.0
 - **Management & Security** : JACC 1.1, Java EE Application Deployment 1.2, Java EE Management 1.1



Agenda

- Présentation
- Java EE 5
- Java EE 5 vs J2EE 1.4
- Java EE 6
- EJB 3.1
 - Encore plus facile
 - Nouvelles fonctionnalités
- Questions



Stateless EJB 2.1

Home interface

```
public interface HelloHome extends EJBHome {  
    Hello create() throws RemoteException,  
                CreateException;  
}
```

Interface locale

```
public interface Hello extends EJBObject {  
    String sayHello() throws RemoteException;  
    Date today() throws RemoteException;  
}
```

EJB

```
public class HelloBean implements SessionBean {  
    public HelloBean() { }  
  
    public String sayHello() {return "Hello Petstore !";}  
    public Date today() {return new Date(); }  
  
    public void ejbCreate() throws CreateException { }  
    public void setSessionContext(  
        SessionContext sessionContext)  
        throws EJBException { }  
    public void ejbRemove() throws EJBException { }  
    public void ejbActivate() throws EJBException { }  
    public void ejbPassivate() throws EJBException { }  
}
```

Descripteur

```
<ejb-jar>  
  <enterprise-beans>  
    <session>  
      <display-name>HelloSB</display-name>  
      <ejb-name>HelloBean</ejb-name>  
      <home>HelloHome</home>  
      <remote>Hello</remote>  
      <ejb-class>HelloBean</ejb-class>  
      <session-type>Stateless</session-type>  
      <transaction-type>Container</transaction-type>  
    </session>  
    <assembly-descriptor>  
      <container-transaction>  
        <method>  
          <ejb-name>HelloBean</ejb-name>  
          <method-name>*</method-name>  
        </method>  
        <trans-attribute>Required</trans-attribute>  
      </container-transaction>  
    </assembly-descriptor>  
  </enterprise-beans>  
</ejb-jar>
```



Stateless EJB 3.0

Interface

@Remote

```
public interface Hello {  
    String sayHello() throws RemoteException;  
    Date today() throws RemoteException;  
}
```

EJB

@Stateless

```
public class HelloBean implements Hello {  
  
    public String sayHello() {return "Hello Petstore !";}   
    public Date today() {return new Date(); }  
}
```



Annotations des EJBs 3.0

- `@Stateless`, `@Stateful`, `@MessageDriven`
- `@Local`, `@Remote`.
- `@PostConstruct`, `@PreDestroy`,
`@PrePassivate`...
- `@Ejb`, `@Resource`, `@WebServiceRef`
- `@TransactionAttribute(REQUIRED)`
- `@RolesAllowed`, `@PermitAll`



Entity Bean CMP 2.1

Home interface

```
public interface HelloHome extends EJBHome {
    Hello create(String cle) throws
        RemoteException, CreateException;
    Hello findByPrimaryKey(String cle) throws
        RemoteException, FinderException;
}
```

Interface locale

```
public interface Hello extends EJBObject {
    String getCle() throws RemoteException;
    String getValeur() throws RemoteException;
    void setValeur(String valeur) throws RemoteException;
}
```

EJB

```
public abstract class HelloBean implements EntityBean {
    public abstract String getCle() throws RemoteException;
    public abstract void setCle(String cle) throws RemoteException;
    public abstract String getValeur() throws RemoteException;
    public abstract void setValeur(String valeur) throws
        RemoteException;

    public String ejbCreate(String cle) throws RemoteException,
        CreateException {

        setCle(cle);    return null;
    }

    public void ejbPostCreate(String cle) throws CreateException{ }
    public void setEntityContext(EntityContext entityContext)
        throws EJBException { }
    public void unsetEntityContext() throws EJBException { }
    public void ejbRemove() throws RemoveException, EJBException { }
    public void ejbActivate() throws EJBException { }
    public void ejbPassivate() throws EJBException { }
    public void ejbLoad() throws EJBException { }
    public void ejbStore() throws EJBException { }
}
```

Descripteur ejb-jar.xml

```
<ejb-jar>
  <enterprise-beans>
    <entity>
      <display-name>HelloEB</display-name>
      <ejb-name>HelloBean</ejb-name>
      <home>HelloHome</home>
      <remote>Hello</remote>
      <ejb-class>HelloBean</ejb-class>
      <persistence-type>Container</persistence-type>
      <prim-key-class>java.lang.String</prim-key-class>
      <reentrant>False</reentrant>
      <cmp-version>2.x</cmp-version>
      <abstract-schema-name>Hello</abstract-schema-name>;
      <cmp-field>
        <field-name>cle</field-name>
      </cmp-field>
      <cmp-field>
        <field-name>valeur</field-name>
      </cmp-field>
      <primkey-field>cle</primkey-field>
    </entity>
  </enterprise-beans>
</ejb-jar>
```

Descripteur propriétaire

```
<jbosscmp-jdbc>
  <defaults>
    <datasource>java:/petstoreDS</datasource>
    <datasource-mapping>mySQL</datasource-mapping>
  </defaults>
  <enterprise-beans>
    <entity>
      <ejb-name>HelloBean</ejb-name>
      <table-name>HELLO_PETSTORE</table-name>
      <cmp-field>
        <field-name>cle</field-name>
        <column-name>key</column-name>
        <jdbc-type>VARCHAR</jdbc-type>
        <sql-type>varchar(10)</sql-type>
      </cmp-field>
      <cmp-field>
        <field-name>valeur</field-name>
        <column-name>value</column-name>
        <jdbc-type>VARCHAR</jdbc-type>
        <sql-type>varchar(50)</sql-type>
      </cmp-field>
    </entity>
  </enterprise-beans>
</jbosscmp-jdbc>
```



Entity JPA

POJO

@Entity

```
public class Hello {
```

@Id

```
    private Long cle;
```

```
    private String valeur;
```

```
    public String getCle() {return cle;}
```

```
    public String getValeur() {return valeur;}
```

```
    public void setValeur() {this.valeur = valeur;}
```

```
}
```



Service Web J2EE 1.4

Service Web

```
public class HelloServiceImpl implements HelloServiceSEI {
    public String sayHello(String param)
        throws java.rmi.RemoteException {
        return "Hello " + param;
    }
}

package endpoint;
import java.rmi.*;
public interface HelloServiceSEI extends java.rmi.Remote {

    public String sayHello(String param)
        throws java.rmi.RemoteException;
}
```

```
<?xml version='1.0' encoding='UTF-8' ?>
<webservices
xmlns='http://java.sun.com/xml/ns/j2ee'
version='1.1'>
<webservice-description>
<webservice-description-name>
HelloService</webservice-description-name>
<wsdl-file>
WEB-INF/wsdl/HelloService.wsdl</wsdl-file>
<jaxrpc-mapping-file>
WEB-INF/HelloService-mapping.xml
</jaxrpc-mapping-file>
<port-component xmlns:wsdl-
port_ns='urn:HelloService/wsdl'>
<port-component-name>HelloService</port-
component-name>
<wsdl-port>wsdl-
port_ns:HelloServiceSEIPort</wsdl-port>
<service-endpoint-interface>
endpoint.HelloServiceSEI</service-endpoint-
interface>
<service-impl-bean>
<servlet-link>WSServlet_HelloService</servlet-
link>
</service-impl-bean>
</port-component>
</webservice-description>
</webservices>
<?xml version='1.0' encoding='UTF-8' ?>
<configuration
xmlns='http://java.sun.com/xml/ns/jax-
rpc/ri/config'>
<service name='HelloService'
targetNamespace='urn:HelloService/wsdl'
typeNameSpace='urn:HelloService/types'
packageName='endpoint'>
<interface name='endpoint.HelloServiceSEI'
servantName='endpoint.HelloServiceImpl'>
</interface>
</service>
</configuration>
```



Service Web Java EE 5

```
package endpoint;  
import javax.jws.WebService;
```

@WebService

```
public class Hello {  
    public String sayHello(String param) {  
        return "Hello " + param;  
    }  
}
```



JAXB 1.0

- Classe générée

[illegible]

308 lignes pour

 $\langle x=1, y=2 \rangle$

- 38 fichiers

- 219 Ko de code



JAXB 2.0

Classe générée

```
@XmlAccessorType(FIELD)
@XmlType(name = "", propOrder = {"x", "y"})
@XmlRootElement(name = "point")
public class Point {
    protected float x;
    protected float y;

    public float getX() {
        return x;
    }
    public void setX(float value) {
        this.x = value;
    }
    public float getY() {
        return y;
    }
    public void setY(float value) {
        this.y = value;
    }
}
```

62 lignes pour

<point><x>1</x><y>2</y></point>

- 2 fichiers
- 3 Ko de code



Lookup J2EE 1.4

```
public class MyEJB implements SessionBean {
    private DataSource myDS;
    public void ejbCreate() {
        try {
            InitialContext ctx = new InitialContext();
            myDS = (DataSource)ctx.lookup("myDS");
        } catch (NamingException ex) {
            // Que faire
        }
    }
}
```

- + le descripteur de déploiement XML



Injection en Java EE 5

```
@Stateless  
public class MyEJB {  
    @Resource(name="myDs") DataSource myDS;  
    @EJB private HelloEJB helloEjb;  
}
```



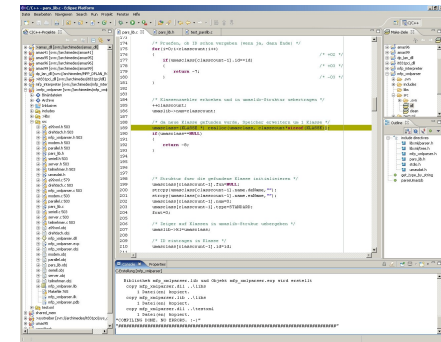
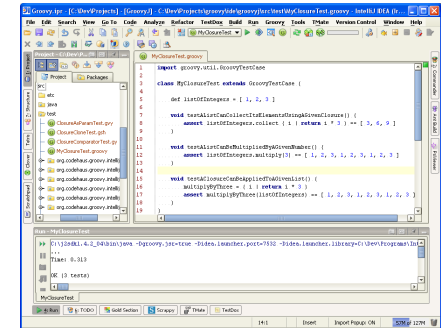
Moins de lignes de code

- Adventure Builder
 - J2EE 1.4 – 67 classes, 3284 lignes de code
 - Java EE 5 – 43 classes, 2777 lignes de code
 - 36% de classes en moins
- RosterApp
 - J2EE 1.4 – 17 classes, 987 lignes de code
 - Java EE 5 – 7 classes, 716 lignes de code
 - J2EE 1.4 descripteurs XML– 9 fichiers, 792 lignes
 - Java EE 5 descripteurs XML – 1 fichier, 5 lignes
 - 58% de classes en moins, 89% de fichiers XML en moins



IDEs pour Java EE 5

- NetBeans
- IntelliJ IDEA
- Eclipse
- JBuilder
- JDeveloper
- JBoss IDE
- BEA Workshop Studio
- Sun Java Studio Creator (JSF)



Les serveurs d'applications

- GlassFish & Sun Java System AppServer
- BEA's WebLogic Server 10
- TmaxSoft JEUS 6
- SAP Netweaver 7.1
- Oracle OC4J 11
- Apache Geronimo 2.0
- JOnAS 5.0
- Websphere 7
- JBoss 5.0 beta



Tests

- Pas de spécification
 - JUnit
 - HTTPUnit
 - Canoo WebTest
 - Mock
 - ...
- Plus facile à tester
 - POJO
 - JPA, moins besoin du conteneur (EntityManager)
 - Ejb3Unit (out of container EJB 3.0 testing)
 - JBoss et GlassFish V3
 - EJB 3 embeddable container



Le déploiement est facilité

- Descripteurs de déploiement optionnels (ejb-jar.xml, application.xml)
 - sauf web.xml, faces-config.xml
- Utilisation des annotations
 - définir les composants (EJB, WebService...)
 - ORM avec JPA (@Entity, @Id, @Column...)
 - définir sécurité (@DenyAll, @PermitAll...)
 - mode transactionnel (@TransactionAttribute)
- Pour les clients riches
 - Java Web Start
 - Application Client Container



Packaging

- Conventions
 - **.war** pour les applications web
 - **.rar** pour les ressources
 - Le répertoire **lib** contient les fichiers jar commun à l'application (classpath automatique)
 - **.jar** avec un attribut Main-Class pour les applications clientes (**ACC**)
 - **.jar** avec une classe annotée **@Stateless** pour les EJBs
 - **.ear** pour les applications d'entreprise



Packaging

foo.ear

foo_web.war

WEB-INF/web.xml
WEB-INF/classes/
com/acme/FooServlet.class
WEB-INF/classes
com/acme/Foo.class

foo_ejb.jar

com/acme/FooBean.class
com/acme/Foo.class

OR

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/FooBean.class



Compatibilité avec J2EE 1.4

- **Compatibilité ascendante** assurée dans la spécification
- EJB 3.0 peuvent être clients des EJB 2.1 (et **inversement**)
- JPQL peut être utilisé dans les méthodes `finder()` des anciens EJB 2.1
- Compatibilité des EL (\$, #) entre JSP et JSF
- JSP 2.0 s'exécutent dans un conteneur JSP 2.1



Les patterns ?

- DTO
 - Les Entity JPA peuvent servir de DTO
- DAO
 - EntityManager pour requêtes simples (CRUD)
 - DAO générique
- Service Locator
 - Injection, Application Client Container
- Delegate
 - Injection
- Action Controller
 - JSF Managed Bean



Java EE 5 et Couche Web

- Pas autant d'améliorations que la couche EJB
 - JSP 2.1, JSF 1.2, JSTL
- Problèmes maintenus
 - Back-refresh du navigateur
 - Double click, double thread
 - Dialogues/flow => Application, Session, Request
=> **JBoss Seam** (future WebBeans avec Java EE 6)
- Utilisation d'autres frameworks
 - Clay, Yahoo UI, Facelets, Seam, Shale, Ajax4JSF, ICEFaces, Avatar
- **Java EE 5 : rupture entreprise, pas web !**



Agenda

- Présentation
- Java EE 5
- Java EE 5 vs J2EE 1.4
- **Java EE 6**
- EJB 3.1
 - Encore plus facile
 - Nouvelles fonctionnalités
- Questions



A propos de Java EE 6

- Développé sous la JSR 316
- Sortie Q1 2009 (+ GlassFish V3)
- Utilisation de profiles
- **Pruning** : suppression progressive
 - EJB CMP => JPA
 - JAX-RPC => JAX-WS
- **Servlet 3.0** et **JSF 2.0** (annotations)
- **WebBeans 1.0** (inspiré de JBoss Seam)
- **JAX-RS 1.0**: RESTful Web Services



Contenu Java EE 6

- ~27 specifications
 - **Web** : **JSF 2.0**, JSP 2.2, JSTL 1.2, **Servlet 3.0**, **JAX-RS 1.0**, **Web Beans 1.0**
 - **Enterprise** : Common Annotations 1.0, **EJB 3.1**, JAF 1.1, JavaMail 1.4, JCA 1.5, JMS 1.1, **JPA 2.0**, JTA 1.1
 - **Web Services** : JAX-RPC 1.1, JAX-WS 2.0, JAXB 2.0, SAAJ 1.0, StAX 1.0, Web Services 2.0, Web Services Metadata 2.0
 - **Management & Security** : JACC 1.1, Java EE Application Deployment 1.2, Java EE Management 1.1



Servlet 2.5

Servlet

```
public class SimpleSample extends HttpServlet {  
  
    public void doGet (HttpServletRequest req,  
                       HttpServletResponse res){  
  
    }  
}
```

web.xml

```
<web-app>  
    <servlet>  
        <servlet-name>MyServlet</servlet-name>  
        <servlet-class>samples.SimpleSample</servlet-class>  
    </servlet>  
    <servlet-mapping>  
        <servlet-name>MyServlet</servlet-name>  
        <url-pattern>/MyApp</url-pattern>  
    </servlet-mapping>  
</web-app>
```



Servlet 3.0

```
@Servlet(urlMapping="MyApp", name="MyServlet")
public class SimpleSample {

    @Get
    public void handleGet(HttpServletRequest req,
                          HttpServletResponse res) {

    }
}
```



Servlet 3.0

- Développé sous la JSR 315
- web.xml (optionel) modulaire
 - Découpé en web-fragment
- Servlet asynchrone (style Comet)
- Programmation par exception
- @Servlet, @ServletFilter, @ServletListener



JSF 2.0

- Développé sous la JSR 314
- Inspiré par Facelet
- Facilite la création de composants graphiques
- Support d'Ajax
- Moins de configuration (faces-config.xml)
- Comet ?



REST

- **REST** (REpresentational State Transfer)
 - Architectures orientées ressource
 - Tout est ressource
 - Toute ressource est adressable par une URL
- **Sémantique CRUD**
 - GET /articles/123
 - DELETE /articles/123

	SQL	HTTP
Créer	INSERT	PUT
Lire	SELECT	GET
Mise à jour	UPDATE	POST
Supprimer	DELETE	DELETE



JAX-RS 1.0

@Remote

@Path("orders/{order_id}")

```
public class OrderResource {
```

@Get

```
Order getOrder(@PathParam("order_id") String id) {
```

```
    ...
```

```
}
```

```
}
```



Web Beans 1.0

- Développé sous la JSR 299
- Inspiré de JBoss Seam
- Spec Lead : Gavin King
- Unifie la couche web et enterprise
 - Glue entre composants (JSF et les EJBs)
 - Contexte conversationnel
- Un composant Web Beans peut être :
 - EJB
 - Managed Bean JSF
 - Entity JPA
 - POJO



Web Beans 1.0

Composant Web Beans

```
public
@Component
class Hello {
    public String say(String name) {
        return "hello " + name;
    }
}
```

JSF

```
<h:commandButton value="Say Hello"
    action="#{hello.say}"/>
```



Web Beans 1.0

Composant Web Beans

@Stateless

public

@Component

```
class Hello {  
    public String say(String name) {  
        return "hello " + name;  
    }  
}
```

JSF

```
<h:commandButton value="Say Hello"  
    action="#{hello.say}"/>
```



Agenda

- Présentation
- Java EE 5
- Java EE 5 vs J2EE 1.4
- Java EE 6
- **EJB 3.1**
 - Encore plus facile
 - Nouvelles fonctionnalités
- Questions



EJB 3.1

- Goals
 - Easier to use
 - New features
- EJB 3.1 - JSR 318
 - Shipped with Java EE 6 – JSR 316
 - Started in August 2007
 - ~20 expert members : companies/individuals
 - Early draft in February 2008
- JPA has its own spec – JSR 317



Disclaimer

- Still under work
- APIs might change
- Q1 2009



Agenda

- Présentation
- Java EE 5
- Java EE 5 vs J2EE 1.4
- Java EE 6
- EJB 3.1
 - Encore plus facile
 - Nouvelles fonctionnalités
- Questions



Local interface

@EJB

```
private Hello h;  
...  
h.sayHello();
```

@Local

```
public interface Hello {  
    String sayHello();  
}
```

@Stateless

```
public class HelloBean implements Hello {  
  
    public String sayHello() {  
        return "Hello Tours JUG !";  
    }  
}
```



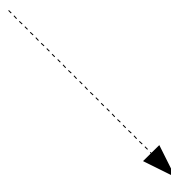
Optional local interface

- Interfaces are not always needed
- “no-interface” view
 - Just the bean
 - All methods are exposed
 - Client injection still needed
 - All container services still available
- Remote interfaces are not optional !



Optional local interface

```
@EJB  
private HelloBean h;  
...  
h.sayHello();
```



```
@Stateless  
public class HelloBean {  
  
    public String sayHello() {  
        return "Hello Tours JUG";  
    }  
}
```



Packaging in ear file

foo.ear

foo_web.war

WEB-INF/web.xml
WEB-INF/classes/
com/acme/FooServlet.class
WEB-INF/classes
com/acme/Foo.class

foo_ejb.jar

com/acme/FooBean.class
com/acme/Foo.class

OR

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/FooBean.class



Packaging

- EJB == `.ear`
 - `lib` directory has simplified deployment
 - But `ear` file is still needed to deploy an EJB
 - Even for a simple web application
- => Deploy an EJB 3.1 in a `war` file
- Keeping all EJBs functionalities



Packaging in war file

foo.war

WEB-INF/classes/
com/acme/FooServlet.class

WEB-INF/classes/
com/acme/FooBean.class



Java EE 6 profiles

- Still in discussion !!!

Web

Servlet 3.0
JSP 2.2
JSR-45
EL 1.2
JSTL 1.2
JSR-250

Web Dev

Servlet 3.0
JSP 2.2
JSR-45
EL 1.2
JSTL 1.2
JSR-250
EJB 3.1 (Lite)
JTA 1.1
JPA 2.0
JSF 2.0
Web Beans 1.0

Full platform

Servlet 3.0	JAX-WS 2.2
JSP 2.2	JAXB 2.2
JSR-45	JSR-109 1.2
EL 1.2	JSR-181 1.1
JSTL 1.2	JMS 1.1
JSR-250	JAF 1.1
EJB 3.1	JavaMail 1.4
JTA 1.1	JSR-115
JPA 2.0	JSR-196
JSF 2.0	JSR-88 1.2
Web Beans 1.0	JSR-77 1.1
JAX-RS 1.0	JAX-RPC1.1
Connectors 1.6	JAXR 1.0



EJB « lite »

- Subset of the EJB 3.1 API
- To be used in Web profile

Local Session Bean
Annotations
Injection
CMT / BMT
Interceptors
Security

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



Global JNDI names

- Client inside a container

```
@EJB Hello h;
```

- Client outside a container

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



Global JNDI names

- Not defined in Java EE
- Vendor specific
 - Not portable
- No standard syntax
- `mappedName` is a partial solution
- Global JNDI names are not just for EJBs (datasource, connection factories, queues...)
- Work in progress !!!



Agenda

- Présentation
- Java EE 5
- Java EE 5 vs J2EE 1.4
- Java EE 6
- EJB 3.1
 - Encore plus facile
 - **Nouvelles fonctionnalités**
- Questions



Singleton

- New session bean component
 - Looks like a stateless / stateful
 - “no interface” view/local/remote interface
- One single EJB per application per JVM
- Use to share state in the entire application
- State not preserved after container shutdown
- Container initialization startup



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);  
    }  
}
```



Singleton and concurrency

- Single thread (default) too restrictive
 - One instance / Multiple threads
- CMC (Container Managed Concurrency)
 - Concurrency through annotations
 - Exclusive / shared lock
 - Timeout
- BMC (Bean Managed Concurrency)
 - `synchronized` keyword



Read-Only Singleton (with CMC)

```
@Singleton
public class CachingBean {

    private Map cache;

    @ReadOnly
    public Map getCache() {
        return cache;
    }

    public void addToCache(Object key, Object val) {
        cache.put(key, val);
    }
}
```



Read-Mostly Singleton (with CMC)

```
@Singleton
@ReadOnly
public class CachingBean {

    private Map cache;

    public Map getCache() {
        return cache;
    }

    @ReadWrite
    public void addToCache(Object key, Object val) {
        cache.put(key, val);
    }
}
```



synchronized Singleton (BMC)

```
@Singleton
@BeanManagedConcurrency
public class CachingBean {

    private Map cache;

    public Map getCache() {
        return cache;
    }

    synchronized public void addToCache(Object key,
                                         Object val) {
        cache.put(key, val);
    }
}
```



Singleton Startup

- Richer lifecycle
 - Container initialization
 - Container shutdown
- Same callback annotations
 - @PostConstruct / @PreDestroy
- @Startup
 - Forces initialization at startup



Singleton Startup

@Singleton

@Startup

```
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);  
    }
```

```
}
```



Timer Service

- Timer Service
 - Added in EJB 2.1
 - Didn't change in 3.0
 - Improved in 3.1
- Programmatic and Calendar based scheduling
 - Cron-like syntax (but nicer)



Timer Service

- « Last day of the month »
- « Every Friday at 4pm »
- « Every five minutes on Monday and Thursday »
- Cron syntax
 - second [0..59], minute [0..59], hour [0..23], year
 - DayOfMonth [0..31]
 - DayOfWeek [0..7] or [sun, mon, tue..]
 - Month [1..12] or [jan,feb..]



Programmatic Timer Creation

```
@Stateless
public class wakeUpBean {

    @Resource TimerService timer;

    public void createWakeUpCall() {
        // Every weekday at 9

        ScheduleExpression expr = new ScheduleExpression().
            dayOfWeek("Mon-Fri").hour(9);

        timer.createTimer(expr);
    }

    @Timeout void wakeUp(Timer t) {
        ...
    }
}
```



Automatic Timer Creation

```
@Stateless
public class wakeUpBean {

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



Asynchronous call

- How to have asynchronous call in EJBs ?
- JMS is to send messages not to do asynchronous calls
- Threads are not allowed (don't integrate well)
- New model
 - Annotations
 - `java.util.concurrent` API



Asynchronous call returning void

```
@Stateless
public class OrderBean {

    public void createOrder() {
        Order order = persistOrder();
        sendEmail(order);
        printOrder(order);
    }

    public Order persistOrder() {...}

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

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



Asynchronous call returning value

```
@Stateless
public class OrderBean {

    public void createOrder() {
        Task task = new Task(...);
        Future<int> computeTask = compute(task);
        ...
        int result = computeTask.get();
    }

    @Asynchronous
    public Future<int> sendEmail(Task task) {
        int result = ...;
        return new javax.ejb.AsyncResult<int>(result);
    }

}
```



Agenda

- Présentation
- Java EE 5
- Java EE 5 vs J2EE 1.4
- Java EE 6
- EJB 3.1
 - Encore plus facile
 - Nouvelles fonctionnalités
- Questions



Java EE 5

- Cuvre
 - EJB Stateless, Stateful, MDB (JMS)
 - JPA
 - JSF
 - WebServices
 - Glassfish installation & configuration
 - ...
- Pour les développeurs et architectes
- Inspiré du PetStore de Sun
- 10 chapitres, développement incrémental



Références

- Java EE 5
 - <http://java.sun.com/javaee/>
 - <http://java.sun.com/javaee/5/docs/tutorial/doc/>
- GlassFish
 - <https://glassfish.dev.java.net/>
 - <http://www.glassfishwiki.org/>
- Livre Java EE 5
 - <http://www.eyrolles.com/>
 - <http://www.antoniogoncalves.org>



Questions



Java EE 5, 6 et les EJBs 3.1

Antonio Goncalves

