

EasyBeans and Java EE 6 web profile

Florent.Benoit@ow2.org



Agenda

- ➔ **Java EE 6 overview**
- ➔ **EJB 3.1 overview**
- ➔ **EasyBeans / EJB 3.1**
- ➔ **Java EE 6 web profile / EasyBeans**
- ➔ **Q&A**

Java EE 6 overview

Java EE 6 overview

➔ Java EE 6 / JSR 316

- <http://jcp.org/en/jsr/detail?id=316>

➔ Expected for september 2009

➔ Agenda

- Updated technologies
- New technologies
- Introduction of profiles
 - Web & Full
- Pruning APIs

Java EE 6: Updated technologies

- ➔ **Connector 1.6 (1.5 in Java EE 5)**
- ➔ **EJB 3.1 (3.0)**
- ➔ **JACC 1.2 (1.1)**
- ➔ **JAX-WS 2.2 (2.0/2.1)**
- ➔ **JAXB 2.2 (2.0/2.1)**
- ➔ **JSF 2.0 (1.2)**
- ➔ **JSP 2.2 (2.1)**
- ➔ **Servlet 3.0 (2.5)**

Java EE 6: New technologies

➔ Bean Validation (JSR 303)

- POJO validation / constraints
- Used by JPA 2.0 and JSF 2.0

➔ JAX-RS 1.1 (RESTful Web services)

- WebServices with your browser !

➔ Java Contexts and Dependency Injection (JSR 299) (was previously called WebBeans)

- Not yet voted as part of Java EE 6

Java EE 6 profiles [1/2]

➔ Introduction of profiles in Java EE 6

➔ Only a subset of the full specification may be used in many cases

- ==> No need for the Full platform

➔ Lightweight platform

➔ Two profiles available:

- Full profile (including all technologies)
- Web profile (including a subset of technologies)

Java EE 6 profiles [2/2]

➔ Full vs web profile

Specification	Web profile	Full Profile
Servlet 3.0	☺	☺
JSP 2.2	☺	☺
EL 2.2	☺	☺
JSTL 1.2	☺	☺
JSF 2.0	☺	☺
JTA 1.1	☺	☺
JPA 2.0	☺	☺
EJB 2.1 lite	☺	☺
EJB 2.1 full		☺
Connector 1.1		☺
JACC 1.2		☺
JAX-RS 1.1		☺
JAX-WS 2.2		☺
JAXB 2.2		☺
JMS 1.1		☺
Mail 1.4		☺
Bean validation 1.0		☺
JAXR 1.0		☺
JSR330 1.1		☺

Pruning APIs

- ➔ **Some APIs in Java EE have been replaced by newer APIs**
 - JAX-RPC ==> JAX-WS
 - EJB 2.x CMP ==> JPA
- ➔ **Some of them are not used/supported often**
 - JAXR
 - JSR77 (management)
 - JSR88 (deployment)

- ➔ **In Java EE 7, these APIs may be removed**

EJB 3.1 overview

EJB 3.1

➔ EJB 3.1/ JSR 318

- <http://jcp.org/en/jsr/detail?id=318>

➔ JPA (Java Persistence API) is now developed in its own specification

- JPA 2.0 / JSR 317
 - <http://jcp.org/en/jsr/detail?id=317>

➔ Improved Ease Of Use

➔ New features

EJB 3.1 New features

- ➔ Singleton**
- ➔ No-Interface local view**
- ➔ Embeddable**
- ➔ Asynchronous calls**
- ➔ War Packaging**
- ➔ New annotations for stateful session beans**
- ➔ @AroundTimeout**
- ➔ Timer enhancements**
- ➔ Naming (module, app, global)**
- ➔ ...**

Singleton: Definition [1/4]

- ➔ Component instantiated once per application (one for each JVM)
- ➔ Singleton instance may be shared by clients and should allow concurrent access.

@Singleton

```
public class MySharedBean implements IShared {  
    @PostConstruct  
    public void init() {  
        // Initialize shared data  
    }  
}
```

Singleton example

Singleton: Eager initialization [2/4]

- ➔ **Default initialization of a singleton is managed by the container**
 - ==> may be Lazy loading
- ➔ **Ensure code is executed at application startup**
 - By using `@Startup` annotation for eager initialization

```
@Startup
@Singleton
public class MySharedBean implements IShared {
    @PostConstruct
    public void atStartup() {
        // code that is called when application is loaded
    }
}
```

Singleton: Dependencies [3/4]

- ➔ When singletons are initialized, an explicit ordering may be specified
- Use of `@DependsOn` annotation

```
@Singleton  
public class SingletonA implements IShared {  
    // ....  
}
```

```
@DependsOn("SingletonA")  
@Singleton  
public class SingletonBeanB implements ISharedB {  
    // ....  
}
```

Singleton: Concurrency [4/4]

- ➔ **Concurrency defined by @ConcurrencyManagement**
 - BEAN or CONTAINER
 - Default concurrency is @ConcurrencyManagement(CONTAINER)

- ➔ **Container: @Lock(READ) and @Lock(WRITE) can be used to manage concurrent access.**
 - Can be defined for each method
 - Default is @Lock(WRITE)

- ➔ **Bean: developer can use synchronized and volatile primitives**

No interface local view

➔ Use case: A web component needs to call a session bean.

- No need to implement an interface
- All public methods are business methods

```
@Stateful
public class MyStatefulBean {
    public String hello() {
        return "hello";
    }
}
```

```
...
@EJB
private MyStatefulBean bean;

public void caller() { System.out.println(bean.hello()); }
...
```

Embeddable EJB container

➔ EJB container can be started in Java SE

- Properties can be given when using `createEJBContainer(Map)` method

```
...  
EJBContainer ejbContainer = EJBContainer.createEJBContainer();  
Context context = ejbContainer.getContext();  
context.lookup("java:global/.....");  
  
ejbContainer.close();  
....
```

Asynchronous calls

➔ Simplified model to avoid Message Driven Beans

@Asynchronous

```
public Future<Integer> performCalculation(...) {  
    // ... do calculation  
    Integer result = ...;  
    return new AsyncResult<Integer>(result);  
}
```

Packaging (war)

➔ EJBs can be defined in a .war file

- `ejb-jar.xml` can be located in `META-INF/` or `WEB-INF/` folder
- EJB classes are located in `WEB-INF/classes` or `WEB-INF/lib`

Stateful: new annotations

➔ New annotations for synchronization:

- @AfterBegin
- @BeforeCompletion
- @AfterCompletion

Timeout interceptors

- ➔ **EJB Timer methods can be intercepted**
 - `@AroundTimeout` method

- ➔ **`InvocationContext.getTimer()` allows to get the timer object**

Timer enhancements

➔ Every Monday at Midnight

- `@Schedule(dayOfWeek="Mon")`

➔ Every five minutes within the hour

- `@Schedule(minute="*/5", hour="*")`

➔ Automatic creation of timers

```
@Schedules({
    @Schedule(hour="12",dayOfWeek="Mon-Thu"),
    @Schedule(hour="11", dayOfWeek="Fri")
})
public void sendLunchNotification() { ... }
```

Naming

➔ **EJBs components used only `java:comp/env`**

➔ **New namespaces**

- **`java:module/`** (ejb-jar scope)
 - `java:module/<bean-name>[!<fully-qualified-interface-name>]`

- **`java:app/`** (application scope)
 - `java:app[/<module-name>]/<bean-name>[!<fully-qualified-interface-name>]`

- **`java:global/`** for the whole applications
 - `java:global[/<app-name>]/<module-name>/<bean-name>`

EJB 3.1 lite vs EJB 3.1 full

➔ EJB 3.1 lite => Java EE 6 web profile

➔ EJB 3.1 full => Java EE 6 full profile

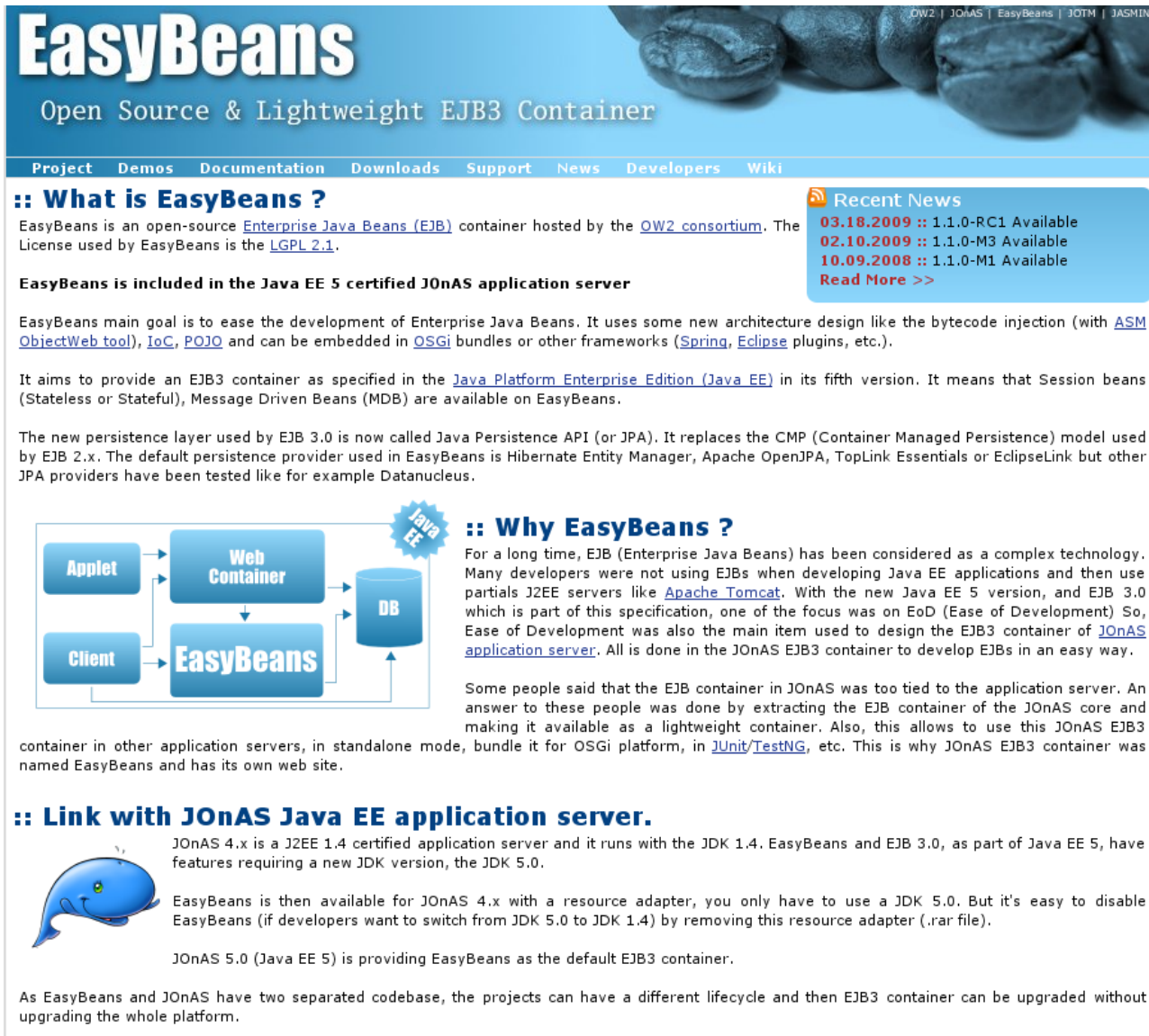
Components	EJB 3.1 Lite	EJB 3.1 Full
Session beans: Stateless, Stateful, Singleton	☺	☺
Java Persistence 2.0	☺	☺
Local/No-Interface view	☺	☺
Interceptors	☺	☺
Container managed and Bean Managed transactions	☺	☺
Declarative and Programmatic security	☺	☺
Embeddable API	☺	☺
Message Driven Beans		☺
CMP 1.x/2.x / BMP Entity Beans		☺
3.0 business remote interfae		☺
2.x Remote Home/Component		☺
JAX-WS Web Service Endpoint		☺
JAX-RPC Web Service Endpoint		☺
EJB Timer service		☺
Asynchronous session bean invocations		☺
RMI-IIOP Interoperability		☺

EJB 3.1 / EasyBeans

EasyBeans

- ➔ Available at <http://www.easybeans.net>
- ➔ Used as EJB 3.0 container in the Java EE 5 certified **JOnAS Application Server**
- ➔ Lightweight and modular
- ➔ Embeddable in JOnAS 4.x, Jetty & Tomcat application servers
- ➔ Available as OSGi bundles
 - Tested on Apache Felix/Eclipse Equinox/Knopflerfish
- ➔ Standalone mode (`java -jar easybeans.jar`)
- ➔ Requirement : JDK 5+





EasyBeans

Open Source & Lightweight EJB3 Container

Project Demos Documentation Downloads Support News Developers Wiki

:: What is EasyBeans ?

EasyBeans is an open-source [Enterprise Java Beans \(EJB\)](#) container hosted by the [OW2 consortium](#). The License used by EasyBeans is the [LGPL 2.1](#).

EasyBeans is included in the Java EE 5 certified JOnAS application server

EasyBeans main goal is to ease the development of Enterprise Java Beans. It uses some new architecture design like the bytecode injection (with [ASM ObjectWeb tool](#)), [IoC](#), [POJO](#) and can be embedded in [OSGi](#) bundles or other frameworks ([Spring](#), [Eclipse](#) plugins, etc.).

It aims to provide an EJB3 container as specified in the [Java Platform Enterprise Edition \(Java EE\)](#) in its fifth version. It means that Session beans (Stateless or Stateful), Message Driven Beans (MDB) are available on EasyBeans.

The new persistence layer used by EJB 3.0 is now called Java Persistence API (or JPA). It replaces the CMP (Container Managed Persistence) model used by EJB 2.x. The default persistence provider used in EasyBeans is Hibernate Entity Manager, Apache OpenJPA, TopLink Essentials or EclipseLink but other JPA providers have been tested like for example Datanucleus.

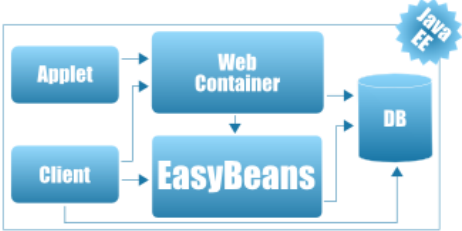
Recent News

03.18.2009 :: 1.1.0-RC1 Available
02.10.2009 :: 1.1.0-M3 Available
10.09.2008 :: 1.1.0-M1 Available
[Read More >>](#)


:: Why EasyBeans ?

For a long time, EJB (Enterprise Java Beans) has been considered as a complex technology. Many developers were not using EJBs when developing Java EE applications and then use partials J2EE servers like [Apache Tomcat](#). With the new Java EE 5 version, and EJB 3.0 which is part of this specification, one of the focus was on EoD (Ease of Development) So, Ease of Development was also the main item used to design the EJB3 container of [JOnAS application server](#). All is done in the JOnAS EJB3 container to develop EJBs in an easy way.

Some people said that the EJB container in JOnAS was too tied to the application server. An answer to these people was done by extracting the EJB container of the JOnAS core and making it available as a lightweight container. Also, this allows to use this JOnAS EJB3 container in other application servers, in standalone mode, bundle it for OSGi platform, in [JUnit/TestNG](#), etc. This is why JOnAS EJB3 container was named EasyBeans and has its own web site.



:: Link with JOnAS Java EE application server.



JOnAS 4.x is a J2EE 1.4 certified application server and it runs with the JDK 1.4. EasyBeans and EJB 3.0, as part of Java EE 5, have features requiring a new JDK version, the JDK 5.0.

EasyBeans is then available for JOnAS 4.x with a resource adapter, you only have to use a JDK 5.0. But it's easy to disable EasyBeans (if developers want to switch from JDK 5.0 to JDK 1.4) by removing this resource adapter (.rar file).

JOnAS 5.0 (Java EE 5) is providing EasyBeans as the default EJB3 container.

As EasyBeans and JOnAS have two separated codebase, the projects can have a different lifecycle and then EJB3 container can be upgraded without upgrading the whole platform.

➔ Gap between EasyBeans and EJB 3.1 lite

Components	EasyBeans 1.1	EJB 3.1 lite
Session beans: Stateless, Stateful	☺	☺
Interceptors	☺	☺
Java Persistence 1.0	☺	☺
Local interface	☺	☺
Container managed and Bean Managed transactions	☺	☺
Declarative and Programmatic security	☺	☺
Java Persistence 2.0	TODO	☺
Session beans: Singleton	TODO	☺
No interface local view	TODO	☺
Embeddable API	TODO	☺
Stateful enhancement	TODO	☺
Timer enhancement	TODO	☺
AroundTimeout	TODO	☺

➔ Enhancements

- EasyBeans architecture allows to plug easily new features
- EasyBeans is already Embeddable

Java EE 6 Web Profile & EasyBeans

EasyBeans / Web profile

➔ Embedded EasyBeans:

- Packages for integration in Tomcat, Jetty and JOnAS 4.x

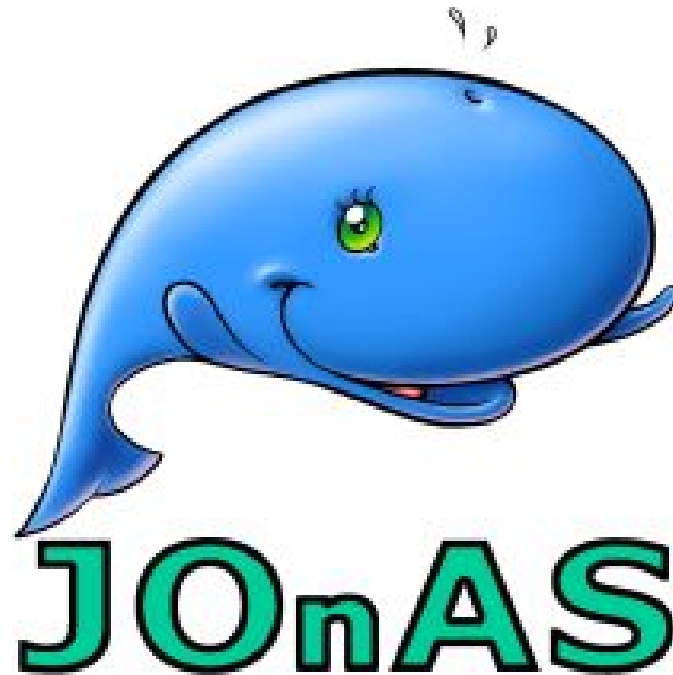
➔ ==> EasyBeans could be plugged in Tomcat/Jetty and JOnAS (by providing Ejb 3.1 lite) and this should provide Java EE 6 Web Profile requirements

➔ EasyBeans can run on top of OSGi gateways

- OSGi based EE 6 web profile

JOnAS 6 Web Profile

- ➔ JOnAS with EasyBeans as EJB 3.1 lite container
- ➔ JOnAS 6 web profile
 - OSGi bundles/services offering Java EE 6 Web Profile



Q&A

easybeans.ow2.org

For more informations
Please contact
Florent Benoit
Florent.Benoit @ ow2 org