JEE 6 - Is JEE development finally less painful ?

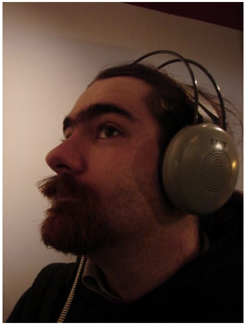
Romain PELISSE - romain@redhat.com

Red Hat Gmbh - Aarhus GOTO 2012

About me...

Romain PELISSE

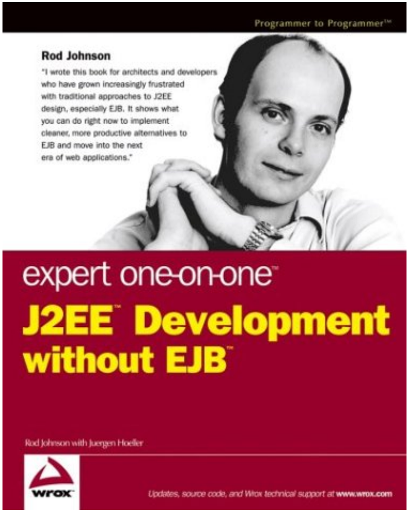
- **Middleware Consultant** at Red Hat (2011)
  - Architect Middleware JBoss
  - Red Hat Enterprise Linux Expert
- Committer PMD and XRadar
- Translation for HgBook
- Teacher at
  - Build and OPP @ESME Sudria, Paris
  - Basic programming @Humboldt University, Berlin
  - Introduction to Middleware @ISEP, Paris
- Technical author for GNU/Linux Magazine France
- Clearly, I don't sleep enough...



What is JEE ?

What is JEE ?

What is JEE ?



Programmer to Programmer™

Rod Johnson

"I wrote this book for architects and developers who have grown increasingly frustrated with traditional approaches to J2EE design, especially EJB. It shows what you can do right now to implement cleaner, more productive alternatives to EJB and move into the next era of web applications."

expert one-on-one™

J2EE™ Development without EJB™

Rod Johnson with Juergen Hoeller

wrox

Updates, source code, and Wrox technical support at **www.wrox.com**

What is JEE ?

- a **standard**
  - a (large) set of JSR plus an umbrella specification
  - designed by committee (JCP)
  - performances varies upon vendor
  - more than 10 years old now (J2EE 1.2, December 12, 1999)
- big **adoption** success during first release
  - EJB programming model
  - high hopes in the *container managed* approach
- but end up being **painful**
  - too complex
  - too many XML descriptor
  - impossible to test outside container
  - ...

Why do I need JEE ?

Maybe you don't...

- JEE is about :
  - distributing programming
  - transaction
  - security
  - asynchronous messaging
- those are, by essence, **complex** to deal with :
  - exception handling
  - rollback for failed transaction
  - manage messages queues
- so, if you don't need this, I don't need JEE ?
  - (Can I go now ?)
- you may still like to have
  - standard
  - programming model

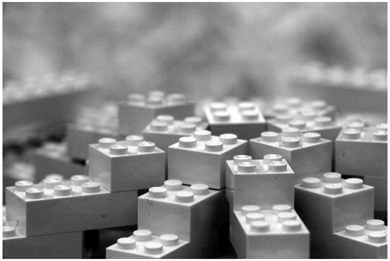
Simplification brought by JEE 5

- reduce needs for XML by introducing annotation
- replace CMP by JPA (an Hibernate inspired API)
- introduced CDI for **dependency injection** - inspired by Spring
- ...

If JEE 5 already simplified greatly JEE applications development, what did bring JEE 6 ?

JEE 6 - Going even farther...

What JEE 6 brings, in term of **simplification** ?



JEE 6 - Going even farther...

Overall strategy

- **convention** over configuration
  - sane default
  - extra configuration only if required
- annotated POJO
- JEE 5 changes applied to the *Web Layer*

JEE 6 - Going even farther...

Key features 1/2

- introduce **profile**
  - **Web Profile 1.0**
  - maybe other profile in upcoming release (messaging ?)
- increase **portability**
- EJB **Lite**
  - allow to use some of the EJB features...
  - ... but within a simple web app (WAR)
- New EJB features
  - Asynchronous calls
  - Planned tasks
- But also :
  - JSF : (faces-config.xml, facelets,...)
  - ReST : Add the JAX-RS 1.0 specification
  - JPA : Update to 2.0 (Collection, JPQL)
- ...

JEE 6 - Going even farther...

Key features 2/2

- ...
- And let's not forget the JDK 6
  - a **lot** of new libraries
  - script easy
  - NIO
  - ...

A few words on Portability

- first of all : it's there - JEE apps **are** already *portable* across container
- JNDI names standardization
- outside container testing for EJB

EJB Lite - the rock star of JEE 6

Why such such a subset ?

- EJB programming model enthusiasm
- bean life cycle (singleton, session bean, ...)
- interceptors (AOP capabilities)
- brings to web app :
  - **support for proper transaction support**
  - **security**
- ... but removes :
  - **message driven bean** (MDB) and JMS
  - remote interface and distributed feature such as
    - Web Services (JAX-WS)
    - ReST (JAX-RS)
    - Remote Procedure Call (JAX-RPC)

EJB Lite - the rock star of JEE 6

Embedded

- EJB are managed by the container, so they need a container to run
- make **unit testing** difficult at best
- JEE 6 and its **EJB lite** feature, makes them **embeddable**
- EJB lite container can be executed **outside** the container, as part of unit test
- simplifies greatly development
  - allow use of local debugger
  - *"EJB logic"* is tested during tests

EJB Lite - the rock star of JEE 6

Deploy in WAR

- prior to JEE 6, deploying EJB based app required a EAR artifact
- most people prefer WAR or are simply not use to EAR
- EAR can only be deployed by JEE Application Server, not Servlet container (Tomcat)

EJB lite leverage the real success of EJB adoption : the programming model, by enabling one to use EJB for none distributed application

JPA : Going in production...

- JPA 1.0 was a standardization of ORM
  - based on the success of *Hibernate*
  - focus on consensus
  - and portability
  - already quite simple to use
- JPA 2.0 features show a growth in **maturity**
  - add locking strategy - such as `PESSIMISTIC`
  - cache
  - query API
- JEE 6 makes JPA - **operation ready**

SQL portability is a now a reality

JSF : Easier, more integrated

- **Java Server Faces** - the only JEE standard web framework
  - strong programming model
  - handles life cycle
  - validation
- designed to build **web applications**...
  - powerful validation
  - interface builds like native UI (assembling components)
  - managed bean life cycle
- ... but not **website**
  - poor URL scheme
  - session based - not stateless
  - server side validation

JSF : Easier, more integrated

- infamous `faces-config.xml` becomes **optional** !
    - annotation, sane default
- no more JSP - integration of **facelets**
    - simple XHTML pages - with extra name spaces for UI component
- AJAX support

Easier to use and implement, lightweight, but probably still too *"application centered"*

Dive in - Asynchronous calls

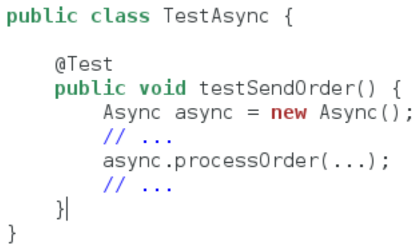
Dive in - Asynchronous call

```
@Stateless
public class Async {

    @Asynchronous
    private void sendOrder(Order order) {
        // ...
    }

    public void processOrder(...) {
        Order order = new Order();
        // ... fill up instance and process order
        sendOrder(order);
        // ... keep going
    }
}
```

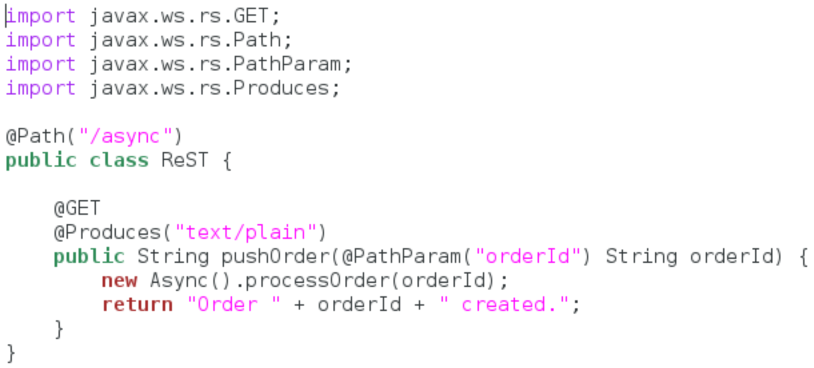
Dive in - Unit test for Asynchronous call



```java
public class TestAsync {

    @Test
    public void testSendOrder() {
        Async async = new Async();
        // ...
        async.processOrder(...);
        // ...
    }
}
```

Dive in - ReST service



```java
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;

@Path("/async")
public class ReST {

    @GET
    @Produces("text/plain")
    public String pushOrder(@PathParam("orderId") String orderId) {
        new Async().processOrder(orderId);
        return "Order " + orderId + " created.";
    }
}
```

JEE 6 and JBoss AS

How JBoss Application Server aligns with all the changes and simplification of JEE 6 ?

JEE 6 and JBoss AS 7.x

- JBoss AS 7.x is JEE 6 **fully** compliant
  - web profile
  - full JEE 6
- Red Hat supported version - **JBoss EAP 6.0.0**
- products redesigned and adhere to JEE 6 philosophy :
  - lightweight
    - lazy loaded services - **on demand**
    - kernel no longer based on JMX
    - services runs in **parallel**
  - simplified **configuration**
    - better internal/user configuration separation
    - one configuration file *"... to rule'em all!"*

JBoss AS redesigned matches the simplification strategy adopted for JEE 6

And so ?

To conclude

- JEE is …
  - … a **standard**, **proven** and **lightweight** technology
  - … that has never been as **simple** to use
  - … and handle quite a lot of **complexity** on your behalf
- but there is still some **pain** because
  - distributed and scalable apps are just not easy to build
  - a lot of pruning still to do, and backwards compatibility is a b*tch
  - fear and resentment are long to disappear

Questions and (hopefully) Answers