

# Practical experience with JBoss Seam

Peter Hilton  
Nicolas Leroux

# Introduction to JBoss Seam

## Seam geography

- Seam is like Europe
  - EJB - France
  - JAAS - United Kingdom
  - JBoss Rules - Germany
  - Ajax - Netherlands
  - jBPM - Belgium
- Although each one can work with the others, Seam provides a uniform component model, where the EU allows political representation at the European level.

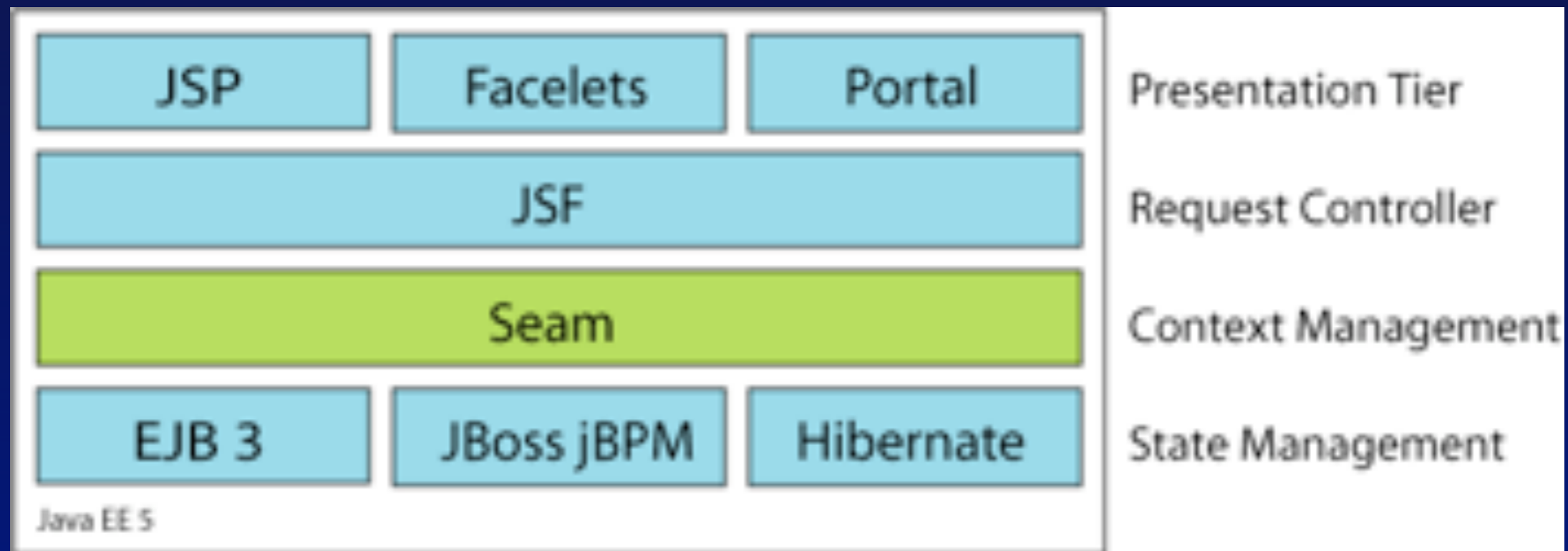
# Introduction to JBoss Seam

## What is Seam exactly?

- Java EE framework for stateful web applications
- Seam is like Spring - unifies the Java Enterprise platform
  - JSF - with JSP, Facelets or Portal
  - EJB3 (thus transactions, Datasource, JMS) - with JPA or Hibernate
  - Web services - WS annotation
  - Seam libraries - logging, e-mail, Seam Text, Ajax integration
  - other tools - JBoss Rules, jBPM, iText
- Dependency injection, configured using annotations
- More focused on Java EE standards than Spring

# Introduction to JBoss Seam

## Seam layer cake



# Introduction to JBoss Seam

## Seam history

- 2000 onwards: Struts is wildly popular
- 2004 onwards: Spring gains popularity
- 2005 Seam started with 1.0 beta 1 (Thomas Heute)
  - removed duplication at the JSF layer
  - conversation scope for stateful web applications
- 2007 Seam 1.2.1 and approaching final 2.0.0 version
- Meanwhile other web frameworks lack traction:
  - Tapestry, WebWork, Wicket, Shale, etc.

# Introduction to JBoss Seam

## Seam concepts

- Seam component: POJO or EJB
- Dependency injection and bijection
- Conversation scope
- Configuration by exception and annotations

# Introduction to JBoss Seam

Seam components and dependency injection

```
public class MyPojo {  
  
}
```

```
@Name("myPojo")  
public class MyPojo {
```

```
    @Logger Log log;  
    @In EntityManager entityManager;  
    @In FacesMessages facesMessages;  
    @In MyDAO myDAO;
```

```
}
```

# Introduction to JBoss Seam

## Conversation scope

```
@Name("drinkingAction")
public class DrinkingAction {

    @In BeerDrinkingFacade pub;

    @In(create=true) @Out Glass glass;

    @Begin(join=true)
    public void getBeer() {

        pub.fill(glass);

    }
}
```

# Practical experience

- Development environment
- EJB integration
- User-interface integration
- Code
- Testing

# Practical experience

## Context

- Lunatech builds custom database-backed web applications
- JSF/Seam selected to replace Apache Struts
  - JSF too verbose by itself - designed for IDE
  - Tapestry, Wicket, Spring MVC: nice but poor EJB3 integration
- Using JBoss Seam since Seam 1.0.0 beta (admin console)
- Two current projects use Seam 1.2.1
- Migration to Seam 2.0 when GA release is available

# Practical experience

## Getting started

- Excellent documentation
- Easier with a JSF background: documentation scattered between Seam, JSF and Facelets
- More productive very quickly
- Steep learning curve: especially new concepts
- Still improving our use of conversations

# Practical experience

## Development environment

- No new tools required: Seam is just a Java library
- JSF/Seam/EJB3 application deployment is slow - need new PCs
- Incremental updates to Facelets (or JSPs) pages helps
- We do not use JavaBean hot-deployment (partly because we are using EJBs)
- Remote debugging works

# Practical experience

## EJB integration

- It just works
- We use EJB for transaction management and JMS
- Transparent integration with EJBs

# Practical experience

## User-Interface prototyping

- Great for prototyping
- Start with a static HTML prototype
- Evolve the HTML into a Facelets view
- Automatic JSF components with jsfc attribute
- `<input type="text" jsfc="h:inputText" value="#{username}"/>`

# Practical experience

## User-Interface integration

- Facelets is a great improvement on JSP (Sun JSF 1.2 in the 2.0)
- jsfc attributes give readable code than JSF tags
- No built-in support for friendly URLs, but `UrlRewriteFilter` works well

# Practical experience

## Code

- No obvious pattern for Seam components; layering optional.
- Confusion between 'action', 'backing-bean', 'component'.
- We started simple with one layer, then extracted DAO layer.
- Annotations are nice.

# Practical experience

## Testing

- UI testing is possible but hard, manually and automated.
- We expect this to be better in Seam 2.0.

# Practical experience

## Future of JBoss Seam

- Is Seam the next Struts?
- JSR-299 Web Beans attempts to standardise the approach
- What is your favourite feature Peter?
- What is your favourite feature Nicolas?
- What do you see missing Peter?
- What do you see missing Nicolas?

# Practical experience

## Future of JBoss Seam

- Is Seam the next Struts?
- JSR-299 Web Beans attempts to standardise the approach
- What is your favourite feature Peter?
- What is your favourite feature Nicolas?
- What do you see missing Peter?
- What do you see missing Nicolas?

## Code Example: sending e-mail

```
@Name("emailMessage")
public class EmailMessageBean implements EmailMessage {

    public void sendEmail(final Order order, final Person sender,
        final Person recipient, final String templatePath)
        throws Exception {

        final String mailAddress = recipient.getMailAddress();

        Contexts.getEventContext().set("mailAddress", mailAddress);
        Contexts.getEventContext().set("order", order);
        Contexts.getEventContext().set("sender", sender);
        Contexts.getEventContext().set("recipient", recipient);

        final Renderer renderer = Renderer.instance();
        renderer.render(templatePath);
    }
}
```

## Code Example: e-mail Facelet view

```
<m:message xmlns="http://www.w3.org/1999/xhtml"
  xmlns:m="http://jboss.com/products/seam/mail"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:vl="http://visiblelogistics.com/vl">

<m:from name="{sender.organisationName}" address="{config.fromAddress}" />
<m:to address="{mailAddress}" />
<m:subject>Invitation to use VisibleLogistics</m:subject>
<m:body>

<p></p>

<p>#{sender.organisationName}
has invited you to sign up for a FREE VisibleLogistics account.</p>

<p>VisibleLogistics is an on-demand visibility and order management service that
automatically sends order status notifications to you
and your partners via e-mail and SMS.
See <a href="http://www.visiblelogistics.com/">www.visiblelogistics.com</a></p>
```

# Code Example: e-mail Facelet view

```
<f:facet name="alternative">  
<h:outputText>  
#{sender.organisationName} has invited you to sign up for a  
FREE VisibleLogistics account.
```

VisibleLogistics is an on-demand visibility and order management service that automatically sends order status notifications to you and your partners via e-mail and SMS. See <http://www.visiblelogistics.com/>

```
</h:outputText>  
</f:facet>
```

```
</m:body>
```

```
</m:message>
```