JBoss Seam
…and beyond

Jeroen Verhulst
Joris De Winne
Karel Maes
Overall Presentation Goal

- basic concepts of Seam with practical demo (Jeroen)
- testing Seam applications (Joris)
- real-life project with Seam (Karel)
RealDolmen Java Competence Center

- Rock solid passion for JAVA
- 200 Java Super Powers
- 10 years of experience in Java projects
RealDolmen Java Competence Center

- Rock solid passion for JAVA
- 200 Java Super Powers
- 10 years of experience in Java projects
- Proven approach: Java Project Platform
RealDolmen Java Expertise
RealDolmen Java Expertise

- Tailor-made software
- Rich Internet Applications
- Mobile Solutions
- Service Oriented Architecture
- Enterprise Architecture
- Web Content Management
- Business Process Management

- Eclipse
- IntelliJ
- RAD
- NetBeans
- FLEX
- JavaFX
- Java EE
- EJB 3
- Spring
- Hibernate
- Agile
- SCRUM
- Test Driven Development
- Websphere
- JBoss
- JIRA
- Bamboo
- Tomcat
- ESB
- Mule / AquaLogic
- Polopoly
- Alfresco
- Swing
- JSF
- Struts
Ask your free poster here!
Ask your free poster here!
JBoss Seam
JBoss Seam

JSF

EJB
JBoss Seam
JBoss Seam

AJAX

jBPM

Spring

Drools

JSF

EJB
Components

- seam container
- lightweight
- control lifecycle
- factory for component instances

![Diagram of application, seam container, and stateful contexts with method calls: getInstance(name), lookup(name), newInstance(), and [not found].]
Components

- component instance
- proxy object
- contextual
Context model

- event 1
- event 2
- event 3
- event 4
- event 5

- page

- conversation 1
- conversation 2
- conversation 3
- conversation 4

- session 1
- session 2
- session 3

- application 1

- business process
Defining components

- @Name
  declares a class as a seam component

- @Scope
  scope in which the context variable will be stored

  - defaults
    - SFSB, JPA entity: conversation
    - SLSB, MDB: stateless
    - JavaBean: event
Accessing components

- **seam API**
  
  ```java
  Component.getInstance("componentName");
  Component.getInstance(ComponentClass.class)
  ```

- **expression language (EL)**
  
  can be used almost everywhere in Seam

- **seam enhancements**
  
  ```java
  List result = new ArrayList();
  for(Object listItem : someList) {
    result.add(listItem.getProperty());
  }
  return result;
  ```
Defining components in XML

- components.xml

- define components

- static injection

```
<components xmlns="http://jboss.com/products/seam/components"
             xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="
                  http://jboss.com/products/seam/components
                  http://jboss.com/products/seam/components-2.0.xsd">
    <component name="matchBean"
               class="com.realdolmen.soccer.business.MatchBeanImpl"
               scope="conversation">
        <property name="match">#{nextMatch}</property>
        <property name="matchDurationInMinutes">15</property>
    </component>
</components>
```
Bijection

- **@In**
- inject value from seam container into property

```java
@In(create=true)
private FacesMessages facesMessages;

@In(required=true, scope=ScopeType.APPLICATION)
private MatchControllerBean matchControllerBean;

@In(value="nextMatch")
public void setMatch(Match match) {
    this.match = match;
}
```
Bijection

- @Out
- outject value from property towards context

```java
@Out(scope=ScopeType.SESSION, value="loggedInUser")
private User user;

@Out(value="nextMatch", required="false")
public Match getMatch() {
    return this.match;
}
```

- null value removes the context variable
Bijection

**dynamic vs. static**

```java
@In("loggedInUser")
@Out("loggedInUser")
private User user = null;

@In
@Out
private Account account = null;

public void updateAccount() {
    account.setNumber("321-7654321-21");
    user = null;
}
```

```
loggedInUser
private String name = "Verhulst"
private String firstName = "Jeroen"

account
private String number = "123-1234567-12"
```
Bijection

injection

seam component instance

@In("loggedInUser")
@Out("loggedInUser")
private User user;

@In
@Out
private Account account;

public void updateAccount() {
    account.setNumber("321-7654321-21");
    user = null;
}

seam container

loggedInUser

private String name = "Verhulst"
private String firstName = "Jeroen"

account

private String number = "123-1234567-12"
Bijection

outjection

```java
private User user;

private Account account;

public void updateAccount() {
    account.setNumber("321-7654321-21");
    user = null;
}
```
Bijection

disinjection

<table>
<thead>
<tr>
<th>seam component instance</th>
<th>seam container</th>
</tr>
</thead>
<tbody>
<tr>
<td>@In(&quot;loggedInUser&quot;)</td>
<td></td>
</tr>
<tr>
<td>@Out(&quot;loggedInUser&quot;)</td>
<td></td>
</tr>
<tr>
<td>private User user = null;</td>
<td>loggedInUser</td>
</tr>
<tr>
<td></td>
<td>private String name = &quot;Verhulst&quot;</td>
</tr>
<tr>
<td></td>
<td>private String firstName = &quot;Jeroen&quot;</td>
</tr>
<tr>
<td>@In</td>
<td></td>
</tr>
<tr>
<td>@Out</td>
<td></td>
</tr>
<tr>
<td>private Account account = null;</td>
<td>account</td>
</tr>
<tr>
<td></td>
<td>private String number = &quot;321-7654321-21&quot;</td>
</tr>
</tbody>
</table>

public void updateAccount() {
    account.setNumber("321-7654321-21");
    user = null;
}
Events

- **raise**
  ```java
  @In
  private Events events;

  events.raiseEvent("matchStarted");  // immediate
  events.raiseTransactionSuccessEvent("matchStarted", match);  // tx completed
  ```

- **observe**
  ```java
  @Observer("matchStarted")
  public void watchMatch(Match match) {
    // same parameters
    ...
  }
  ```

- **built-in events**
  ```java
  org.jboss.seam.preSetVariable.newPlayer
  org.jboss.seam.postCreate.matchBean  // matchBean passed as arg
  org.jboss.seam.preDestroyContext.APPLICATION
  org.jboss.seam.afterTransactionSuccess
  org.jboss.seam.notLoggedIn
  ```
DEMO

• soccer application
Testing Seam: Overview

- Unit Testing
- Integration Testing
  - Mocks
  - Mock Data (DBUnit)
- Integration Testing User Interactions
- Testing Seam

Supported by Seam
Too difficult to write?
Testing Seam: Component

- Unit Testing: Seam Components → PoJo's
- Integration Testing
Testing Seam: Component

- Unit Testing: Seam Components → PoJo's
- Integration Testing

```java
@Test
public void testVersion() throws Exception {
   new ComponentTest() {
      protected void testComponents() throws Exception {
         EntityManager em = (EntityManager) getValue("#{entityManager}");
         Assert.assertNotNull(em);

         Commentator commentator = new CommentatorImpl();
         commentator.setName("Annoying person");
         commentator.setUsername("username");
         commentator.setPassword("password");

         em.persist(commentator);

         Assert.assertEquals(0, commentator.getVersion().intValue());

         commentator.setVersion(1);
         Assert.assertEquals(1, commentator.getVersion().intValue());
      }
   }.run();
}
```
EntityManager

- Integration Testing: EntityManager injection
EntityManager

Integration Testing: EntityManager injection

```xml
<persistence:entity-manager-factory name="soccer" persistence-unit-name="soccer"/>

<persistence:managed-persistence-context name="entityManager" entity-manager-factory="#{soccer}" auto-create="true" persistence-unit-jndi-name="java:/SoccerDS" />
```
Integration Testing: EntityManager injection

```xml
<persistence:entity-manager-factory name="soccer" persistence-unit-name="soccer"/>

<persistence:managed-persistence-context name="entityManager" entity-manager-factory="#{soccer}" auto-create="true" persistence-unit-jndi-name="java:/SoccerDS"/>
```

```java
protected void testComponents() throws Exception {
    EntityManager em = (EntityManager) getValue("#{entityManager}");
    Assert.assertNotNull(em);
```
Who's going to pay for the extra test server?
JBoss Embedded

- Bootstrap on classpath
  http://www.jboss.org/community/docs/DOC-9690
JBoss Embedded

- Bootstrap on classpath
  http://www.jboss.org/community/docs/DOC-9690

```xml
<core:init debug="true" jndi-pattern="${jndi.pattern}"/>
```
JBoss Embedded

```xml
<profiles>
  <profile>
    <!-- "Profile package" is used for production -->
    <id>package</id>
    <activation>
      <activeByDefault>false</activeByDefault>
      <property>
        <name>embedded</name>
        <value>false</value>
      </property>
    </activation>
    <properties>
      <jndi.pattern>soccer-ear-#{version}/#{e jbName}/local</jndi.pattern>
    </properties>
  </profile>
  <profile>
    <!-- "Profile test" is used for embedded jboss -->
    <id>test</id>
    <activation>
      <activeByDefault>true</activeByDefault>
      <property>
        <name>embedded</name>
        <value>true</value>
      </property>
    </activation>
    <properties>
      <jndi.pattern>#{ejbName}/local</jndi.pattern>
    </properties>
  </profile>
</profiles>
```
Jboss Embedded

```xml
<profiles>
  <profile>
    <!-- "Profile package" is used for production -->
    <id>package</id>
    <activation>
      <activeByDefault>false</activeByDefault>
    </activation>
    <properties>
      <jndi.pattern>soccer-ear-${version}/#{ejbName}/local</jndi.pattern>
    </properties>
  </profile>
  <profile>
    <!-- "Profile test" is used for embedded jboss -->
    <id>test</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
      <jndi.pattern>#{ejbName}/local</jndi.pattern>
    </properties>
  </profile>
</profiles>
```
Why should I write mockups?
TestingMocks

- 3rd party systems.

```java
/**
 * @author Joris De Winne – RealDolmen
 */
@Name("homeBean")
@Install(precedence = Install.MOCK)
@Scope(ScopeType.CONVERSATION)
public class HomeBeanMockImpl implements HomeBean {

/**
 * @see com.realdolmen.soccer.business.home.HomeBean#getNextMatch()
 */
@Factory("nextMatch")
public Match getNextMatch() {
```

3rd party systems.
Testing Mocks

- 3rd party systems.

Put src/test/java on classpath

```java
/**
 * @author Joris De Winne – RealDolmen
 */
@Name("homeBean")
@Install(precedence = Install.MOCK)
@Scope(ScopeType.CONVERSATION)
public class HomeBeanMockImpl implements HomeBean {

/**
 * @see com.realdolmen.soccer.business.home.HomeBean#getNextMatch()
 */
@Factory("nextMatch")
public Match getNextMatch() {

```
Mocks: continued

```java
@Factory("nextMatch")
public Match getNextMatch() {
    Match matchMock = new MatchImpl();
    matchMock.setAway(null);
    matchMock.setAwayScore(10);
    matchMock.setCommentator(null);
    matchMock.setEvents(null);
    matchMock.setHome(null);
    matchMock.setHomeScore(1);
    matchMock.setId(666L);
    matchMock.setMatchDate(Calendar.getInstance().getTime());
    matchMock.setStartDate(Calendar.getInstance().getTime());
    matchMock.setVersion(999);
    return matchMock;
}
```
Mocks: continued

- And how do we test it?
Mocks: continued

- And how do we test it?

```java
@Test
public void testgetNextMatch() throws Exception {
    new ComponentTest() {

        protected void testComponents() throws Exception {
            Match match = (Match) invokeMethod("#{homeBean.getNextMatch()}");
            Assert.assertNotNull(match);
            Assert.assertEquals(666, match.getId().intValue());
        }

        }.run();
    }
```
DBUnit

- We want some dummy data inserted!
- We want some dummy data inserted!

```java
@Test(testName = "authentication", suiteName = "all", groups = "business")
public class AuthenticationManagerTest extends DBUnitSeamTest {
```
DBUnit

- We want some dummy data inserted!

```
@Test(testName = "authentication", suiteName = "all", groups = "business")
public class AuthenticationManagerTest extends DBUnitSeamTest {

    @Override
    protected void prepareDBUnitOperations() {
        beforeTestOperations.add(new DataSetOperation(
            "soccer/datasets/AuthenticationTestData.xml"));
    }
```
Test everything!
Also JSF pages!
User Interactions

- Test web application ≈ Functional Testing
- Test web application ≈ Functional Testing

```java
/**
 * Test correct signon services contact login.
 * @throws Exception
 */
@Test
public void testCorrectSignon() throws Exception {
    final FacesRequest fr = new FacesRequest("/login.xhtml");
```
final FacesRequest fr = new FacesRequest("/login.xhtml") {

    @Override
    protected void beforeRequest() {
        super.beforeRequest();
    }

    @Override
    protected void processValidations() throws Exception {
        validateValue("#{identity.username}", "pemae");
        validateValue("#{identity.password}", "pemae");
        assert !isValidationFailure();
    }

    @Override
    protected void updateModelValues() throws Exception {
        setValue("#{identity.username}", "pemae");
        setValue("#{identity.password}", "pemae");
    }

    @Override
    protected void invokeApplication() {
        invokeMethod("#{identity.login}");
    }

    @Override
    public String toString() {
        return result;
    }

};
fr.run();
They take too long to execute!
Run faster
Run faster

- JBoss Embedded: Kick out jms
Run faster

- JBoss Embedded: Kick out jms
- Use Seam 2.1.1.CR1 → CR2
Run faster

- JBoss Embedded: Kick out jms
- Use Seam 2.1.1.CR1 → CR2

- Example: compilation, packaging and testing (54 integration tests)
  → 1 minute 18 seconds
Time to run the tests!
Seam References

- Corelio
  - Newspapers: 0.5 million / day
  - Freesheets: 4 million copies / week
  - Magazines: CÃTCHY, PC Magazine
  - Printing (e.g.: 50,000 tons of newspaper per annum)
  - E-media: Online sites (> 300,000 visitors / day)
Seam References

- Corelio
  - Newspapers: 0.5 million / day
  - Freesheets: 4 million copies / week
  - Magazines: CÃTCHY, PC Magazine
  - Printing (e.g.: 50,000 tons of newspaper per annum)
  - E-media: Online sites (> 300,000 visitors / day)
Seam References

- Corelio
  - Newspapers: 0.5 million / day
  - Freesheets: 4 million copies / week
  - Magazines: CATCHY, PC Magazine
  - Printing (e.g.: 50,000 tons of newspaper per annum)
  - E-media: Online sites (> 300,000 visitors / day)
Reference Case

- External Service For Prevention and Protection at Work
- Well-being at work
- Company visits
BRIE – Project Goal

- Improve existing service of company visits
- Give the company visitor (400 users) a tool that helps him in his daily activities
  - Registration of observations in central DB
  - Easy creation of summary report for the customer
Focus on JEE, Open Source, SOA

Users are not familiar with web applications

Changing business processes of company visits
BRIE – Architecture

+ Scalable infrastructure

“Rich” look and feel

JEE6 in mind (Web Beans)

Learning Curve developers
Ease of development

Automatic testing not straightforward
BRIE – Testing

- Unit & Automated Functional Tests
  - Selenium
  - SeamTest

- Performance & Stress Tests
  - JMeter
  - cld

- Maven Site & Bamboo

- Maven Archetype

- JSF, Ajax
- Maven
- JSF, Seam
- project dashboard
- jumpstart future

projects
Summary

- Web applications == Seam
Summary

- Web applications == Seam
- Perfect integration: JSF, Facelets, Richfaces, EJB3, ...
Summary

• Web applications == Seam
  + Perfect integration: JSF, Facelets, Richfaces, EJB3, ...
  + Fast development
Summary

• Web applications == Seam
  • Perfect integration: JSF, Facelets, Richfaces, EJB3, ...
  • Fast development

• Issues:
Summary

• Web applications == Seam
  + Perfect integration: JSF, Facelets, Richfaces, EJB3, ...
  + Fast development

• Issues:
  - Mavenizing the project
Summary

• Web applications == Seam
  + Perfect integration: JSF, Facelets, Richfaces, EJB3, ...
  + Fast development

• Issues:
  - Mavenizing the project
  - JBoss Embedded configuration
Summary

• Web applications == Seam
  + Perfect integration: JSF, Facelets, Richfaces, EJB3, ...
  + Fast development

• Issues:
  - Mavenizing the project
  - JBoss Embedded configuration
  - Stress / Performance testing
Thanks for your attention!