

# JBoss Seam ...and beyond



Jeroen Verhulst  
Joris De Winne  
Karel Maes

**REALDOLMEN**

# 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



# RealDolmen Java Competence Center

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



# RealDolmen Java Competence Center



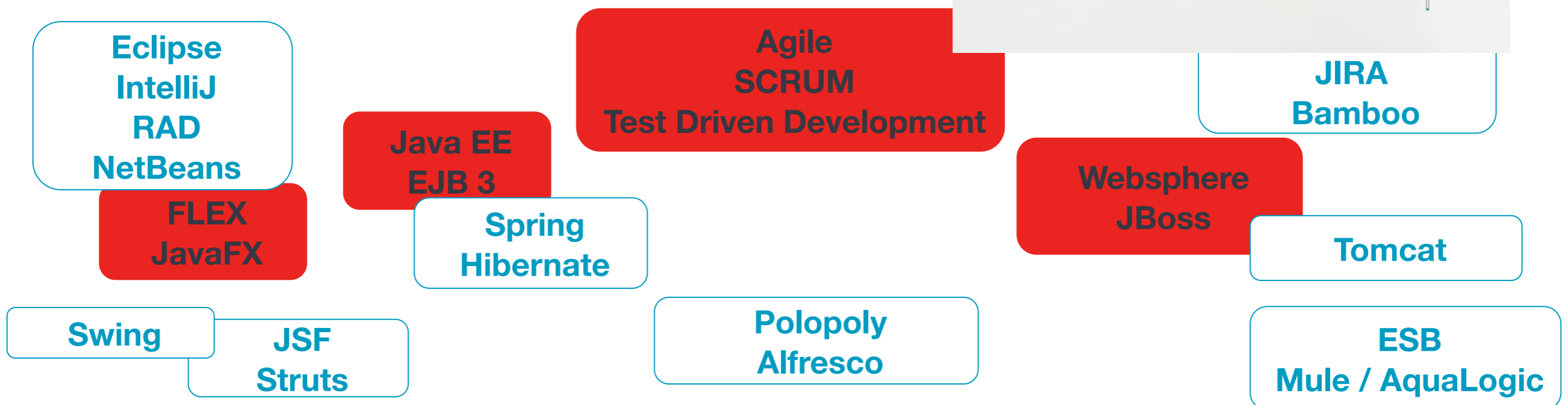
- **Rock solid passion for JAVA**
- **200 Java Super Powers**
- **10 years of experience in  
Java projects**
- **Proven approach:  
Java Project Platform**

# RealDolmen Java Expertise



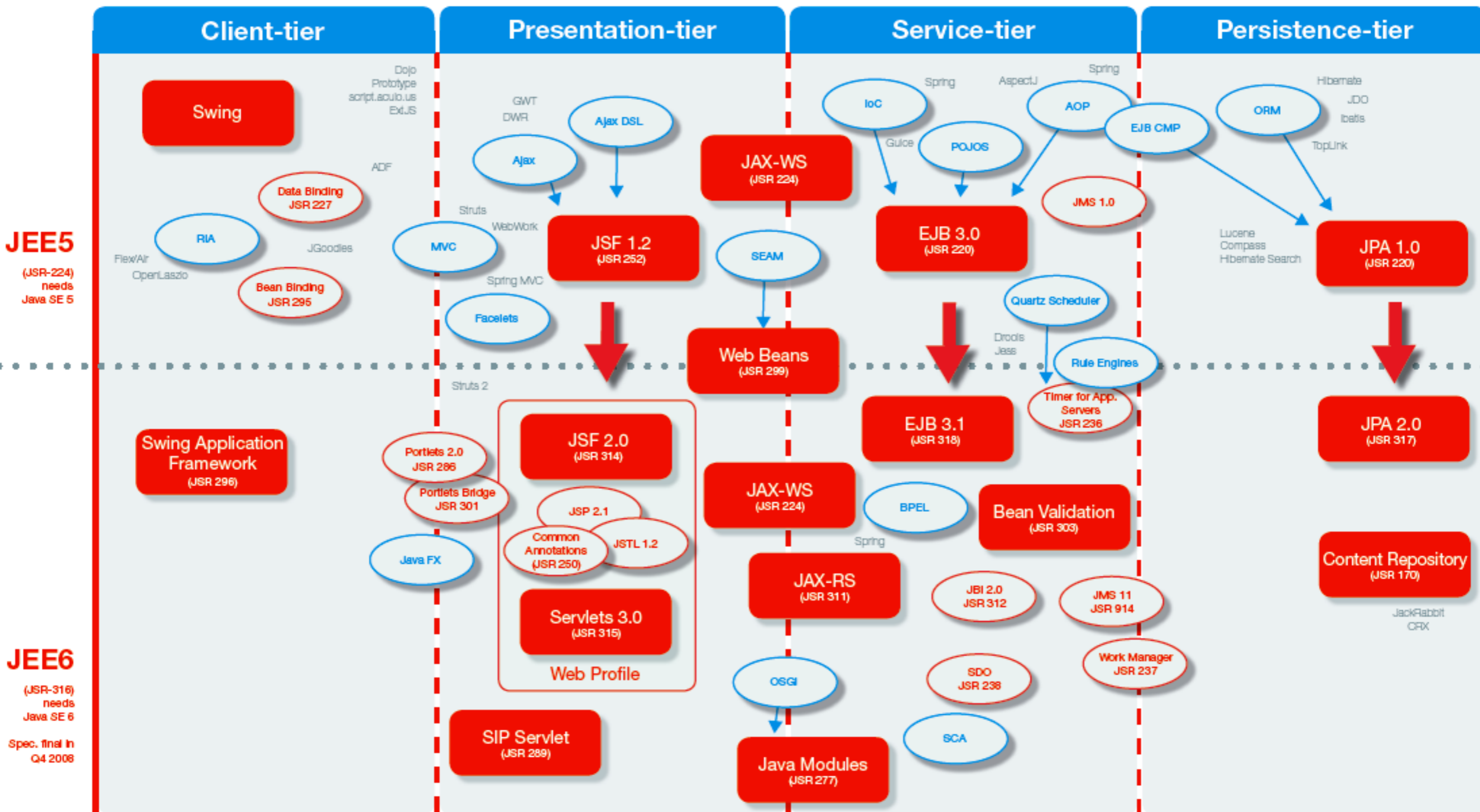
# RealDolmen Java Expertise

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



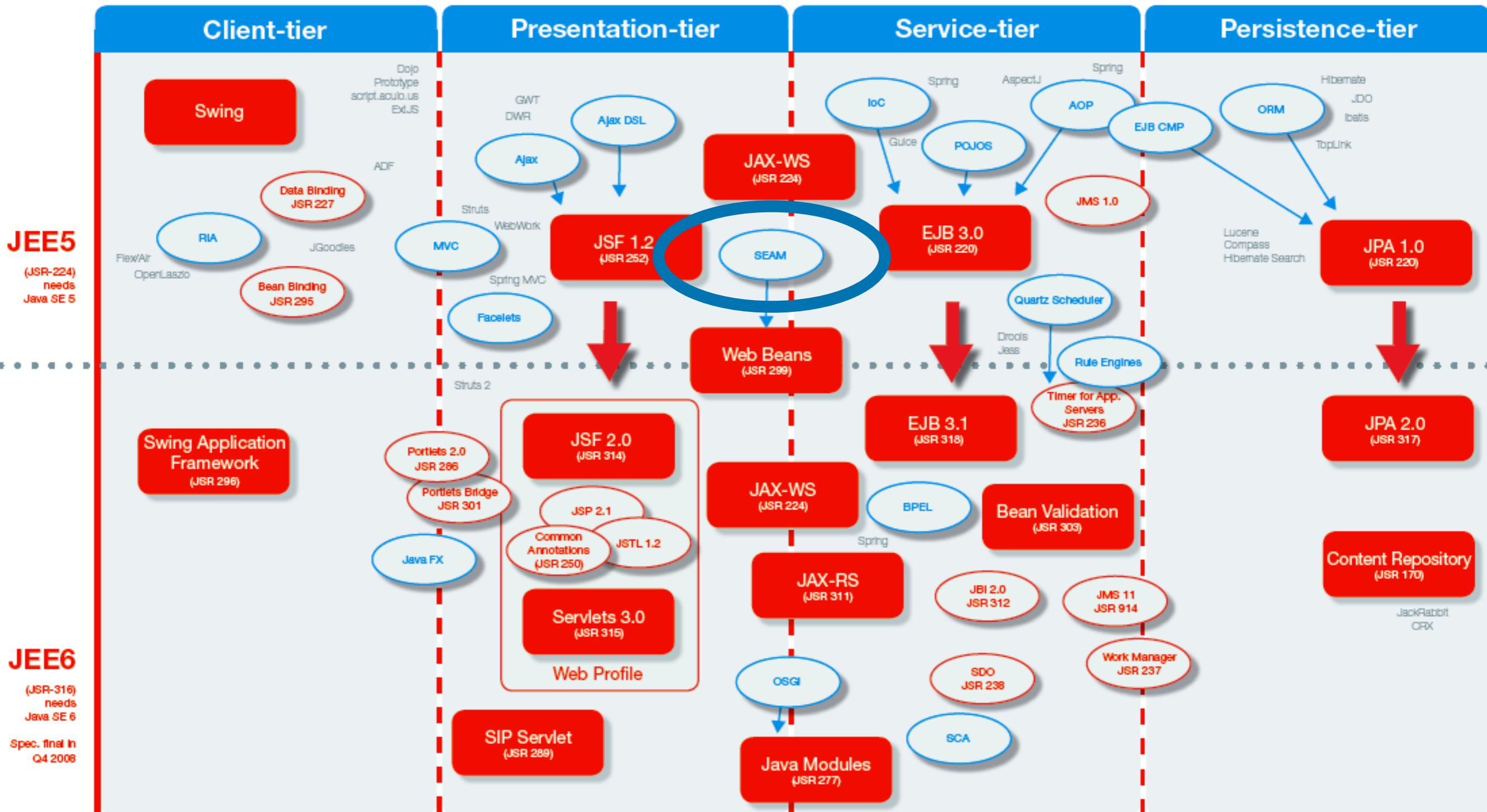


# The Java Enterprise Edition Roadmap 2008



Ask your **free** poster here!

# The Java Enterprise Edition Roadmap 2008



Ask your **free** poster here!

# JBoss Seam



# JBoss Seam

**JSF**



**EJB**

# JBoss Seam



# JBoss Seam

AJAX

Drools

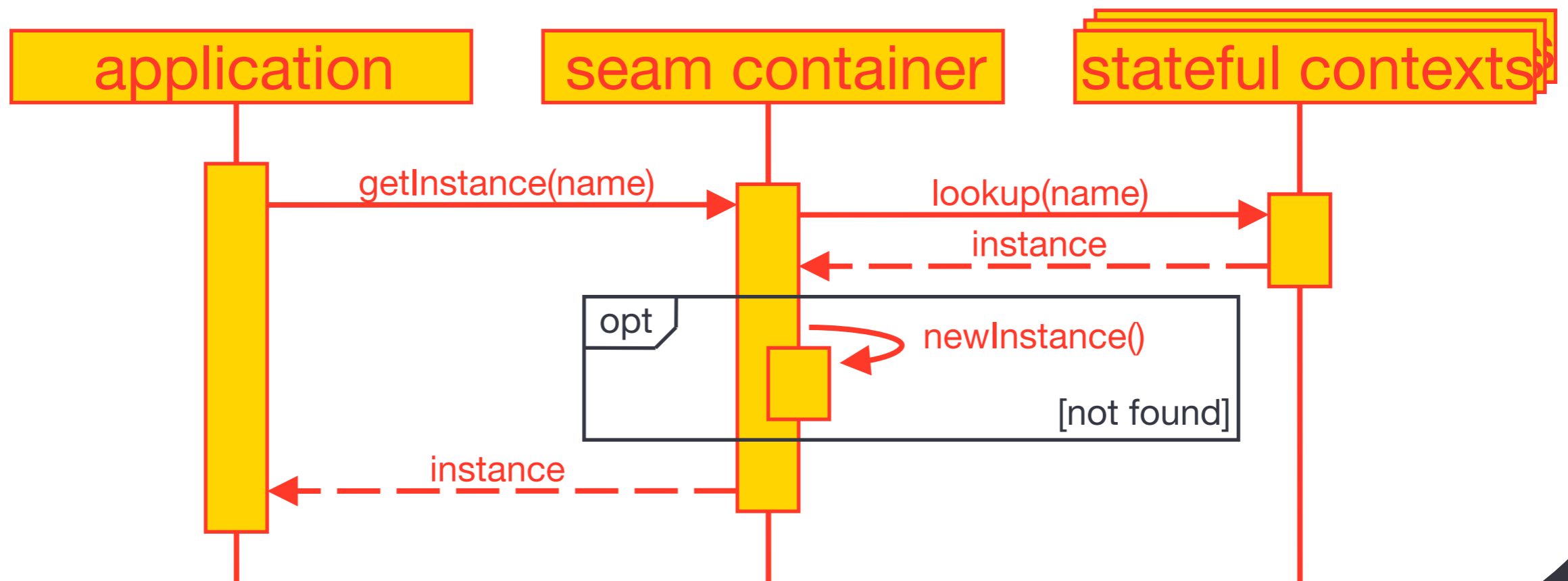


jBPM

Spring

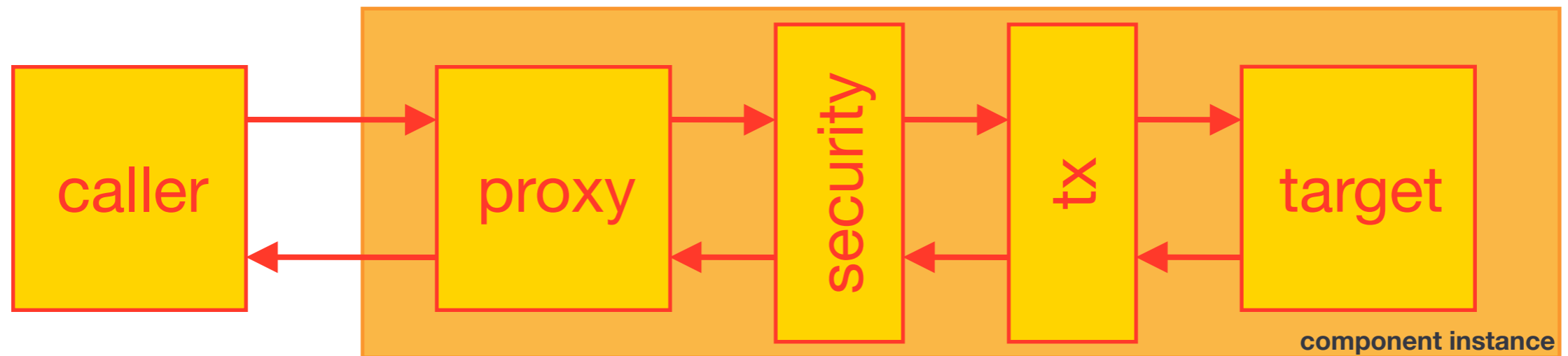
# Components

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



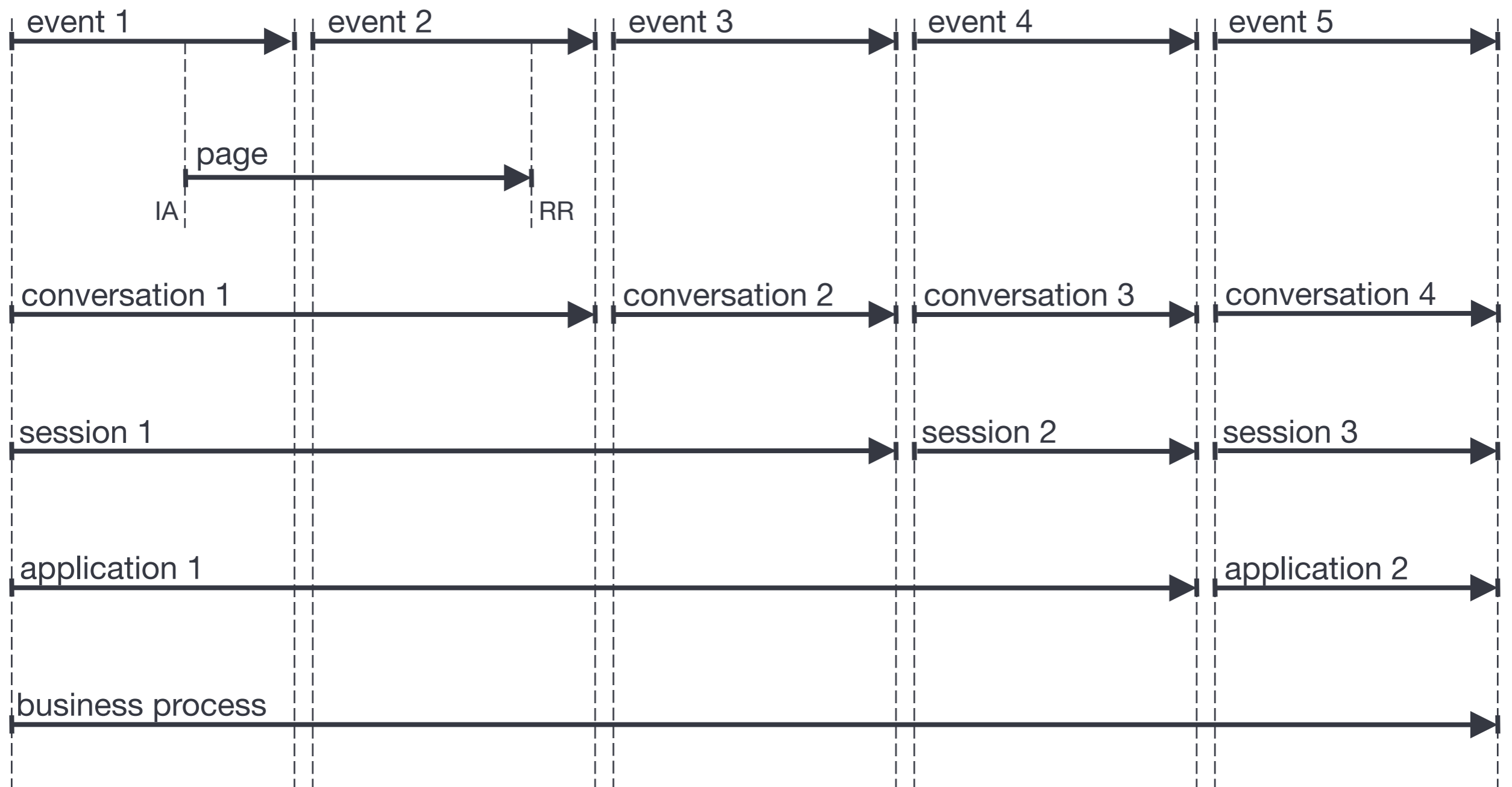
# Components

- component instance
- proxy object



- contextual

# Context model



# 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

```
Component.getInstance("componentName");  
Component.getInstance(ComponentClass.class)
```


- expression language (EL)

- can be used almost everywhere in Seam

- seam enhancements

```
#{seamComponentName.methodName(parameter1, parameter2)}  
#{componentName.getProperty(parameter1)}  
#{someList.size}  
#{someList.{ listItem | listItem.property}}
```

```
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"
  si="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

```
@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

```
@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

seam component instance

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

@In
@Out
private Account account = null;

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

- 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

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 = "321-7654321-21"
```

# Bijection


- disinjection

seam component instance

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

@In
@Out
private Account account = null;

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 = "321-7654321-21"
```

# Events

- raise

```
@In
private Events events;

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

- observe

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

- built-in events

```
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 with External Actions
- 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

```
@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

```
<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" />
```

# EntityManager

- Integration Testing: EntityManager injection

```
<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" />
```

```
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

# 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>

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

# JBoss Embedded

```
<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}/#{ejbName}/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

```
<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}/#{ejbName}/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>
```

# Why should I write mockups?

# Testing Mocks

- → 3rd party systems.

```
/**
 * @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() {
```



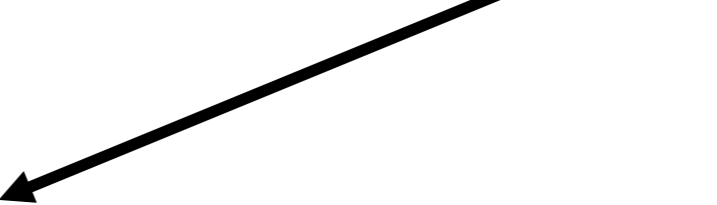
# Testing Mocks

- → 3rd party systems.

Put src/test/java on classpath

```
/**
 * @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

```
@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?

```
@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

# DBUnit

- We want some dummy data inserted!

# DBUnit

- We want some dummy data inserted!

```
@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  $\approx$  Functional Testing

# User Interactions

- Test web application  $\approx$  Functional Testing

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

# User Interactions: cont.

```
final FacesRequest fr = new FacesRequest("/login.xhtml") {

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

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

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

    @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

# DEMO

Time to run the tests!



# Seam References

- Corelio
  - Newspapers: 0.5 million / day
  - Freesheets: 4million 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: 4million 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: 4million copies / week
  - Magazines: CÃTCHY, 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

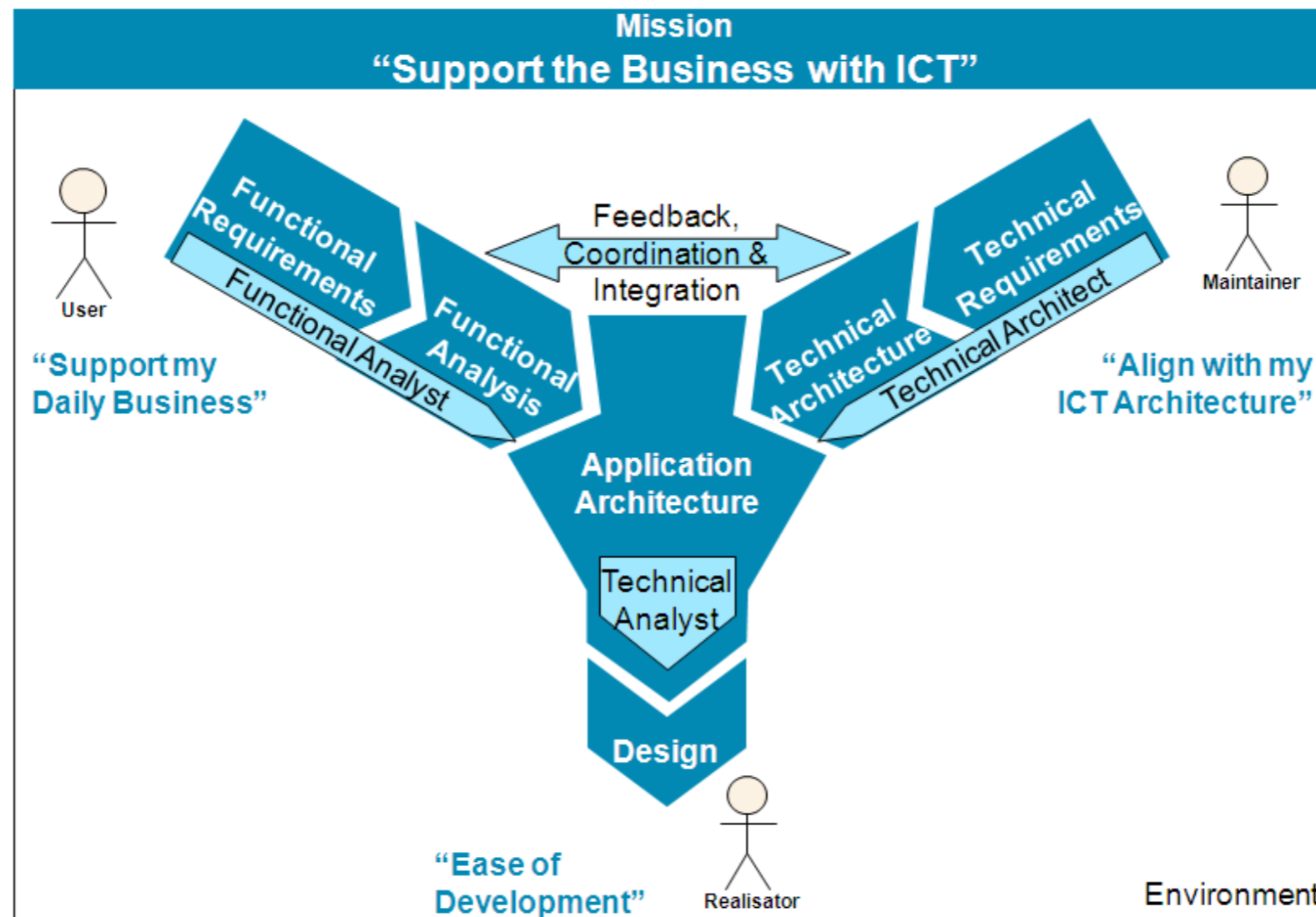
# BRIE – Project Goal



BRIE

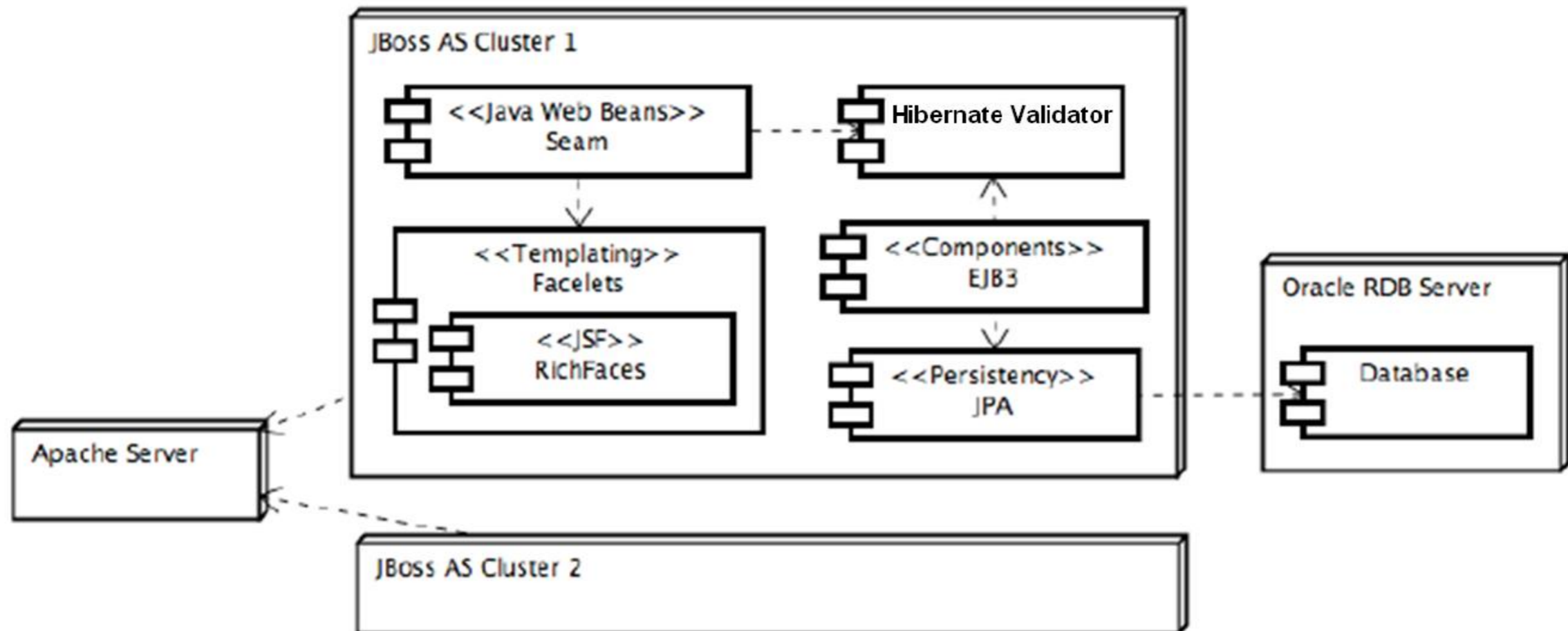
- Improve existing service of company visits
- Give the company visitor (400 users) a tool that helps him in his daily activities
  - Registration of observations in central DB
  - Easy creation of summary report for the customer

# BRIE – Global Design Phase



- Focus on JEE, Open Source, SOA
- Users are not familiar with web applications
- Changing business processes of company visits

# BRIE – Architecture



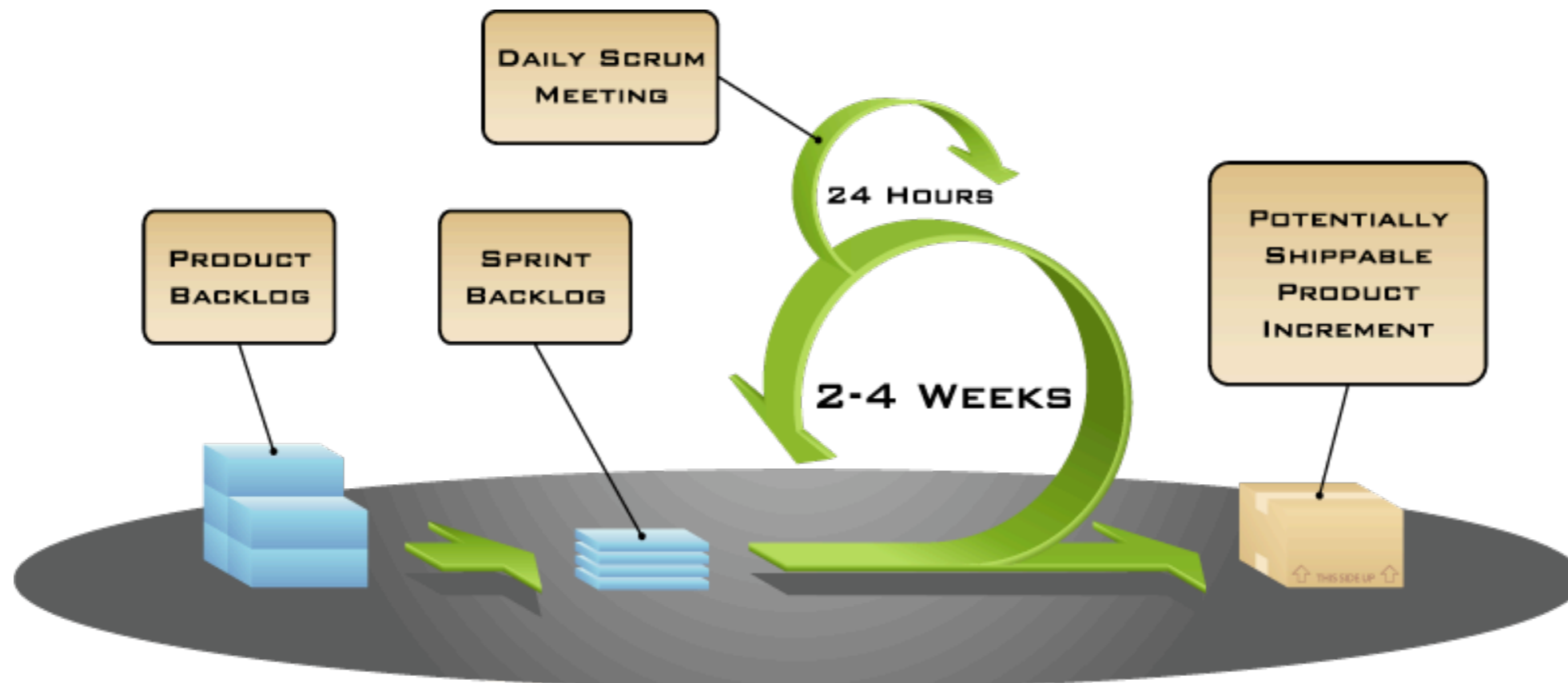
Scalable infrastructure

“Rich” look and feel

JEE6 in mind (Web Beans)

Learning Curve developers

# BRIE – Scrum



COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE



Ease of development



Automatic testing not straightforward

# BRIE – Testing

- **Unit & Automated Functional Tests**

- Selenium

- SeamTest

- JSF, Ajax

- Maven

- **Performance & Stress Tests**

- JMeter  
cld

- JSF, Seam

- **Maven Site & Bamboo**

- project dashboard

- **Maven Archetype**

- jumpstart future

projects

# Summary

# 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

# Q&A



Thanks for your attention!