



JavaEE Seams Easy

JBoss Seam: A Web Application Component Framework for
EJB3, JPA, JSF and AJAX



Agenda

- The JavaEE 5 programming model: Remaining issues
- Introducing Seam
 - Contextual Components
 - Seam Conversations
 - Bijection
 - Remoting
 - Testability
- Short demo
- Summary
- Q&A

Cadec 2007, JBoss Seam, Slide 2
Copyright 2006-2007, Callista Enterprise AB



Java EE 5 programming model

- JSF 1.2
 - Template language
 - extensible component model for widgets
 - “Managed bean” component model
 - JavaBeans with dependency injection
 - XML-based declaration
 - Defines interactions between the page and managed beans
 - Expression Language (EL) for binding controls to managed beans

Cadec 2007, JBoss Seam, Slide 3
Copyright 2006-2007, Callista Enterprise AB



Java EE 5 programming model (contd.)

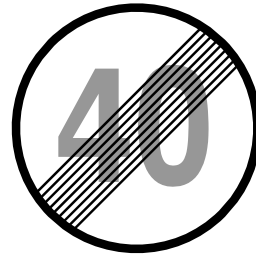
- EJB 3.0
 - Component model for transactional components
 - dependency injection
 - declarative transaction and persistence context demarcation
 - sophisticated state management
 - JPA ORM for persistence
 - Annotation-based programming model

Cadec 2007, JBoss Seam, Slide 4
Copyright 2006-2007, Callista Enterprise AB



Java EE 5 Compared to J2EE

- Simpler programming model
 - Fewer artifacts (look Mum, no DTOs!)
 - Less noise (EJB boilerplate, Struts boilerplate)
 - Much simpler ORM
 - Finer grained components
- More powerful for complex problems
 - Powerful ORM engine
 - EJB interceptors support lightweight AOP
 - JSF is amazingly flexible and extensible
- Much better testability
 - All components (except JSP pages) can be unit tested in isolation



Cadec 2007, JBoss Seam, Slide 5
Copyright 2006-2007, Callista Enterprise AB



So what is the problem?

- Incoherent programming models
- Remaining JSF complexity
 - Managed beans are just unnecessary plumbing
 - XML configuration
- Lack of power
 - Enemy of the State:
No Flows
 - Dependency Injection is not enough
- What about Ajax?

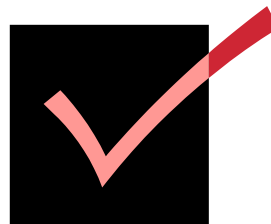


Cadec 2007, JBoss Seam, Slide 6
Copyright 2006-2007, Callista Enterprise AB



Seam Goals

- Unify the two component models
 - Simplify Java EE 5, filling a gap
 - Improve usability of JSF
- Deprecate stateless architectures
- Decouple technology from execution environment
- Enable richer user experience



Cadec 2007, JBoss Seam, Slide 7
Copyright 2006-2007, Callista Enterprise AB



Fix the mismatch between JSF and EJB 3.0 component programming models

- No more excessive XML configuration
 - **@Annotations everywhere!**
 - **#{EL} expressions everywhere!**

- No more awkward "Managed Beans".

(Want to know what gets between my JSFs and my EJBs?

- Nothing ...)



Cadec 2007, JBoss Seam, Slide 8
Copyright 2006-2007, Callista Enterprise AB



Seam Components in Action!

The `@Name` annotation binds a component to a contextual variable

```
@Stateful
@Name("docEdit")
public EditDocumentBean implements
    EditDocument {
    @PersistenceContext
    private EntityManager em;
    private Long id;
    public void setId(Long id) {
        this.id = id;
    }
    private Document document;
    public Document getDocument() {
        return document;
    }
    public void save() {
        ...
    }
}
```

Cadec 2007,
Copyright 20

Seam components can be
Session Beans, Entity Beans
or simply POJOs

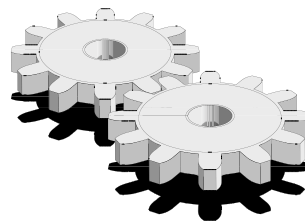
```
<f:form>
<table>
<tr>
<td>Title</td>
<td>
<h:inputText
    value="#{docEdit.document.title}"
    >
<f:validateLength maximum="100"/>
</h:inputText>
</td>
</tr>
<tr>
<td>Password</td>
<td><h:inputText
    value="#{docEdit.document.content
    }"/></td>
</tr>
</table>
<h:messages/>
<h:commandButton type="submit"
    value="Save"
    action="#{docEdit.save}"/>
</f:form>
```

Context variables are
accessible from EL

CALLISTA

Contextual Components

- Most of the problems in Web Application development relate directly or indirectly to state management
 - Servlet Spec contexts (Page, Request, Session, Application) are not meaningful in terms of the application
- We want a richer context model providing meaningful, "logical" contexts
- What's missing: **Conversations**



Cadec 2007, JBoss Seam, Slide 10
Copyright 2006-2007, Callista Enterprise AB

CALLISTA

Conversations In Action!

Components are assigned to a scope using the **@Scope** annotation

```
@Stateful
@Name("documentEditor")
@Scope(ScopeType.CONVERSATION)
public EditDocumentBean implements EditDocument {
    @PersistenceContext
    private EntityManager em;
    ...
    @Begin
    public String get() {
        document = em.find(Document.class, id);
        return document==null ? "notFound" : "success";
    }
    @End
    public String save(Document doc) {
        document = em.merge(doc);
        return "success";
    }
}
```

Conversations are demarcated using **@Begin** and **@End** annotations

Cadec 2007, JBoss Seam, Slide 11
Copyright 2006-2007, Callista Enterprise AB



Conversations helps solve real, inherent problems in Web Applications

- Multi-Window operations
- Back Button support
- "Workspace Management"
- Nested conversations
 - multiple concurrent inner conversations within an outer conversation - a stack of continuable states

• Time for a demo!

Cadec 2007, JBoss Seam, Slide 12
Copyright 2006-2007, Callista Enterprise AB



Conversations and Persistence

- Transaction-scoped persistence contexts have problems if you (re)use objects across transactions
 - not held open for entire request (while rendering view), hence can result in `LazyInitializationException` while navigating lazy associations
 - `NonUniqueObjectException` reassociating detached instances
 - Less opportunity for caching
- Conversation-scoped persistence contexts solves these problems
 - Much cleaner than well-known (and highly questionable) “open session in view” idiom!

Cadec 2007, JBoss Seam, Slide 13
Copyright 2006-2007, Callista Enterprise AB



Dependency Injection Sucks! Here's *Bijection*!

- Dependency injection is broken for stateful components
 - Rigid and static
 - Does not support the notion of contexts
- What we really need is *Bijection*, which is
 - Dynamic
 - Bidirectional
 - Injection
 - “Outjection”
 - Contextual



Cadec 2007, JBoss Seam, Slide 14
Copyright 2006-2007, Callista Enterprise AB



Bijection in Action!

```
@Stateless
@Name("changePassword")
public class ChangePasswordBean implements Login {
    @PersistenceContext
    private EntityManager em;
    @In @Out
    private User currentUser;
    public String changePassword() {
        currentUser = em.merge(currentUser);
    }
}
```

@In and **@Out** annotations are used to inject and "outject" variables into the context

Cadec 2007, JBoss Seam, Slide 15
Copyright 2006-2007, Callista Enterprise AB



Bijection in Action!

```
@Stateful
@Name("hotelSearch")
@Scope(ScopeType.SESSION)
@LoggedIn
public class HotelSearchingAction implements HotelSearching
{
    @PersistenceContext private EntityManager em;
    private String searchString;
    private int pageSize = 10;
    @DataModel private List<Hotel> hotels;
    @DataModelSelection private Hotel selectedHotel;
    public String find() {...}
}
```

@DataModel annotation outjects search result into the context

Cadec 2007, JBoss Seam, Slide 16
Copyright 2006-2007, Callista Enterprise AB



Bijection in Action – Referring to context variable

```
<h:dataTable value="#{hotels}" var="hot"
  rendered="#{hotels.rowCount>0}">
  <h:column>
    <f:facet name="header">Name</f:facet> #{hot.name}
  </h:column>
  <h:column>
    <f:facet name="header">Address</f:facet> #{hot.address}
  </h:column>
  <h:column>
    <f:facet name="header">City, State</f:facet> #{hot.city},
    #{hot.state}
  </h:column>
</h:dataTable>
```

Cadec 2007, JBoss Seam, Slide 17
Copyright 2006-2007, Callista Enterprise AB



What about Ajax?

- Ajax seems to be here to stay, whether we like it or not. And the JavaScript programming model comes straight from Hell ...
- JSF components in themselves can encapsulate JavaScript code, but that is a closed model
 - What about those fancy JS libraries that you might have to use (Rico, Prototype, script.aculo.us, ...)?
 - What about access to the server-side, contextual components?
- Seam Remoting gives DWR-like Javascript access to any Seam component



Cadec 2007, JBoss Seam, Slide 18
Copyright 2006-2007, Callista Enterprise AB



Seam Remoting in Action

```
@Local
public interface Manager {
    ... ..

    @WebRemote
    public boolean checkName (String
        name);
}

@Name("manager")
public class ManagerAction
    implements Manager {

    ... ..

    public boolean checkName (String
        name) {
        ...
    }
}

<script language="javascript">
    // Get the "manager" Seam component
    var manager =
        Seam.Component.getInstance("manager");

    // Make the async call with a callback
    handler
    function checkName () {
        var e =
            document.getElementById("form:name");
        var inputName = e.value;
        manager.checkName(inputName,
            checkNameCallback);
    }

    function checkNameCallback (result) {
        if (result) {
            hideCheckNameError ();
        } else {
            showCheckNameError ();
        }
    }
}
</script>
```

Cadec 2007, JBoss Seam, Slide 19
Copyright 2006-2007, Callista Enterprise AB



Runtime Requirements

- Java SE 5
- JSF 1.x
- Servlet 2.4+ container (i.e. Tomcat 5.x)
- JPA implementation only required if persistent contextual objects are used
- EJB 3 container only required if the Contextual objects are EJBs
 - Also works with JBoss Embeddable EJB3 container

Cadec 2007, JBoss Seam, Slide 20
Copyright 2006-2007, Callista Enterprise AB



Conclusions

- Seam provides a unified programming model that brings together JSF, JPA, EJB 3 and AJAX
- Relies on two powerful tools:
 - Annotations
 - Expression Language
- Greatly simplifies development of stateful Web Applications with complex flows
- ... but is not yet a standard, which might lead to single vendor lock-in. You might want to wait for WebBeans (JSR 299).



Cadec 2007, JBoss Seam, Slide 21
Copyright 2006-2007, Callista Enterprise AB



Time for Questions!

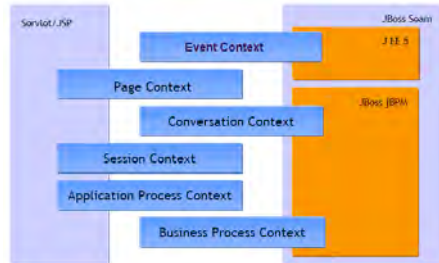


Cadec 2007, JBoss Seam, Slide 22
Copyright 2006-2007, Callista Enterprise AB



Seam Contexts

- **EVENT**
- PAGE
- **CONVERSATION**
- SESSION
- **PROCESS**
- APPLICATION



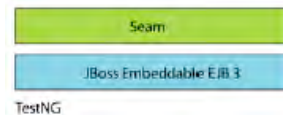
- Components are assigned to a scope using the `@scope` annotation

Cadec 2007, JBoss Seam, Slide 23
Copyright 2006-2007, Callista Enterprise AB



Support for Test Driven Development

```
public class BookingTest extends SeamTest {
    @Test
    public void testBookHotel() throws Exception {
        new FacesRequest() {
            @Override
            protected void invokeApplication() throws Exception {
                Contexts.getSessionContext().set("loggedIn", true);
                Contexts.getSessionContext().set("user", new User("Gavin King", "foobar"));
            }
        }.run();
        new FacesRequest("/main.xhtml") {
            @Override
            protected void updateModelValues() throws Exception {
                setValue("#{hotelSearch.searchString}", "Union Square");
            }
            @Override
            protected void renderResponse() {
                DataModel hotels = (DataModel) Contexts.getSessionContext().get("hotels");
                assert hotels.getRowCount() == 1;
                assert ( (Hotel) hotels.getRowData() ).getCity().equals("NY");
                assert getValue("#{hotelSearch.searchString}").equals("Union Square");
                assert !Manager.instance().isLongRunningConversation();
            }
        }.run();
    }
}
```



Cadec 2007, JBoss Seam, Slide 24
Copyright 2006-2007, Callista Enterprise AB

