



Wicket + CDI + JPA (with JBoss AS 7)

Web developer's dream

Ondřej Žižka

Feb 2012

Quick survey

- Do you work with
 - Wicket?
 - Zend framework?
 - JSF?
 - Struts?
 - Spring framework?
 - EJB 3.x?
 - CDI / Weld?



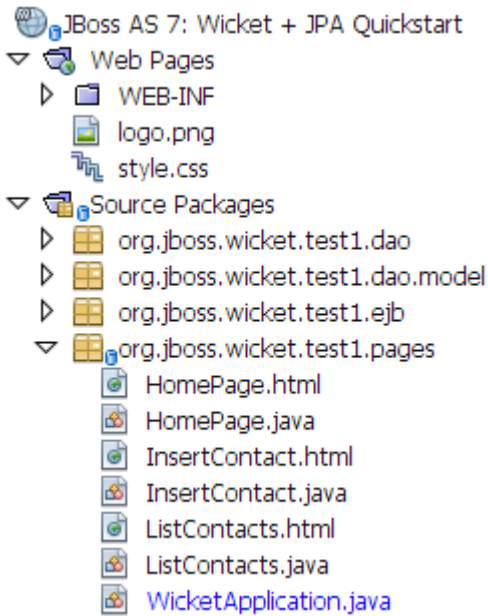
Wicket characteristics



- Pros:
 - Component-oriented
 - Java-oriented
 - No XML configuration
 - Lightweight – small memory footprint
 - Easily modifiable
 - Clusterable
- Cons:
 - No spec
 - Different way of thinking
 - Need to undo brain damage done by JSF et al.



Wicket basic: Pages



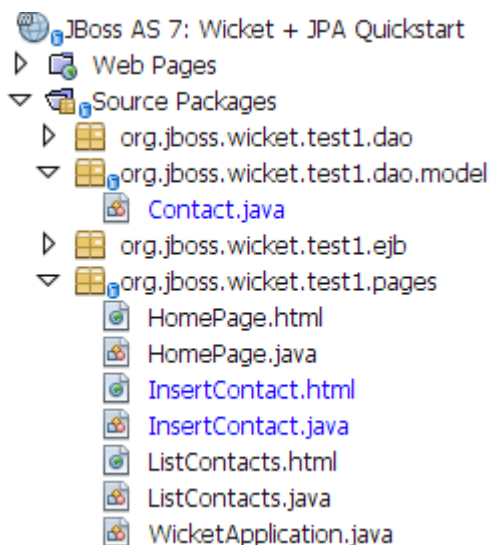
```
public class HelloWorldPage extends WebPage {  
    public HelloWorldPage() {  
        add(new Label("label", "Hello world"));  
    }  
}
```

```
<html>  
<body>  
    <h1>Wicket example page</h1>  
    <div wicket:id="label">Replaced text</div>  
</body>  
</html>
```

```
<html>  
<body>  
    <h1>Wicket example page</h1>  
    <div>Hello world</div>  
</body>  
</html>
```



Wicket basic: Models & Binding to components

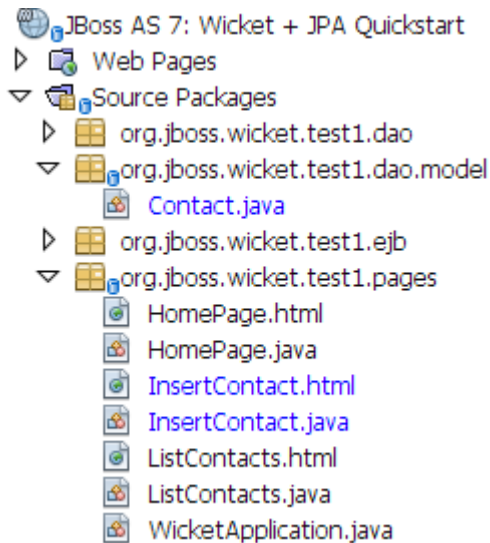


```
@Entity
public class Contact implements Serializable
{
    private Long id;
    private String name;
    private String email;

    public Contact(Long id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }
    Get/set, equals...
}
```

```
<body>
    <span wicket:id="feedback"></span>
    <form method="post" action="#" wicket:id="insertForm">
        <table border="0" cellpadding="5">
            <tr>
                <td width="200">Name</td>
                <td><input type="text" wicket:id="name" size="20"/></td>
            </tr>
            <tr>
                <td width="200">Email</td>
                <td><input type="text" wicket:id="email" size="20"/></td>
            </tr>
            <tr>
                <td width="200">&nbsp;</td>
                <td><input type="submit" value="Insert"/></td>
            </tr>
        </table>
    </form>
</body>
```

Wicket basic: Models & Binding to components



```
public class InsertContact extends WebPage {

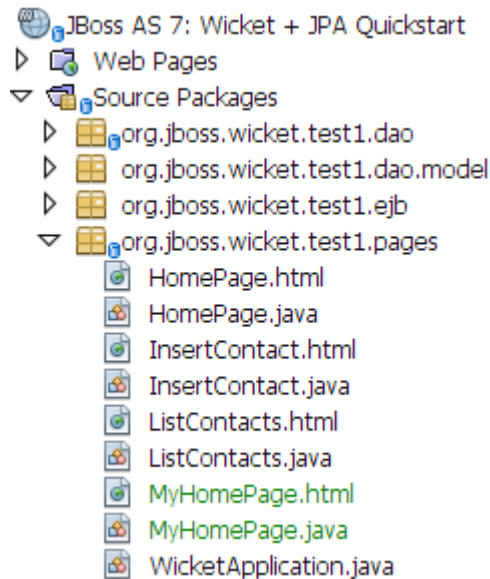
    private String name;
    private String email;

    @EJB private ContactDao contactDao;

    public InsertContact() {
        add( new Form<Contact>("insertForm") { {
            add(new TextField("name", new PropertyModel(this, "name")));
            add(new TextField("email", new PropertyModel(this, "email")));
        }
        @Override protected void onSubmit() {
            contactDao.addContact(name, email);
            setResponsePage(ListContacts.class);
        }
        });
        add(new FeedbackPanel("feedback"));
    }
    Get/Set
}
```



Wicket basic: Page update & navigation



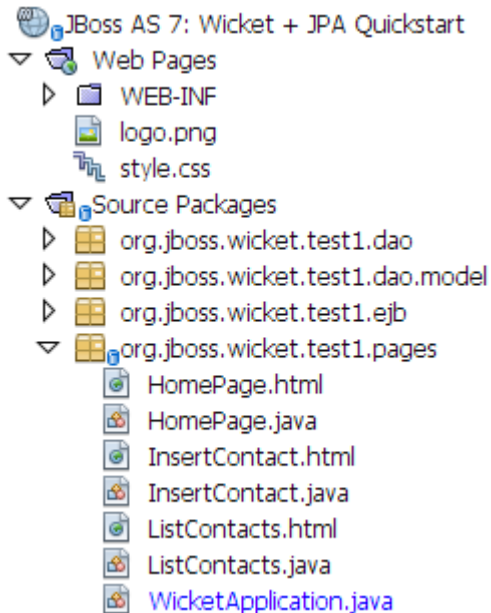
```
class MyHomePage extends WebPage {

    public MyHomePage() {
        add(new Link("changeState", new Model()) {
            public void onClick() {
                // this changes link state and therefore page state
                this.setModelObject(new Random().nextInt());
            }
        });

        add(new Link("gotoNewPage") {
            public void onClick() {
                setResponsePage(new MyHomePage());
            }
        });
    }
}
```



Wicket basic: Application object



```
package org.jboss.wicket.test1.pages;

import org.apache.wicket.protocol.http.WebApplication;

/**
 * Application object for your web application.
 */
public class WicketApplication extends WebApplication
{
    @Override
    public Class<HomePage> getHomePage() {
        return HomePage.class;
    }
}
```



Wicket basic: Application object

```
public class MyApplication extends WebApplication {

    protected void init() {
        boolean inProduction =
            Application.DEPLOYMENT.equals(getConfigurationType());

        getMarkupSettings().setStripComments(inProduction);

        getMarkupSettings().setStripXmlDeclarationFromOutput(true);
        getMarkupSettings().setStripWicketTags(true);

        getResourceSettings()
            .setStripJavascriptCommentsAndWhitespace(inProduction);

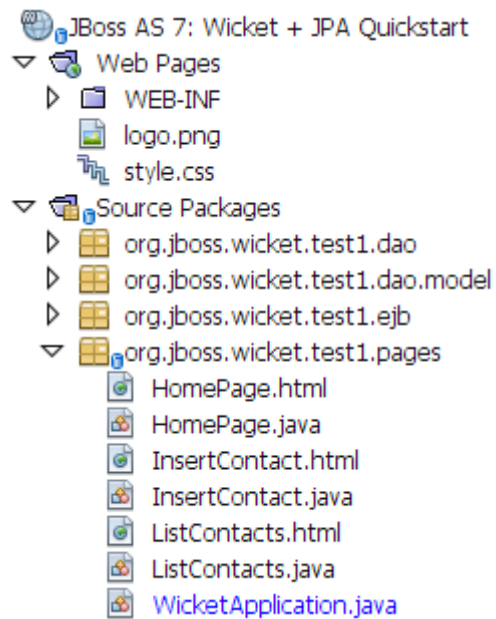
        // Expiry page and error page
        if (inProduction) {
            getApplicationSettings()
                .setPageExpiredErrorPage(MyExpiredPage.class);

            // show internal error page rather than default developer page
            getApplicationSettings().setInternalErrorPage(MyErrorPage.class);

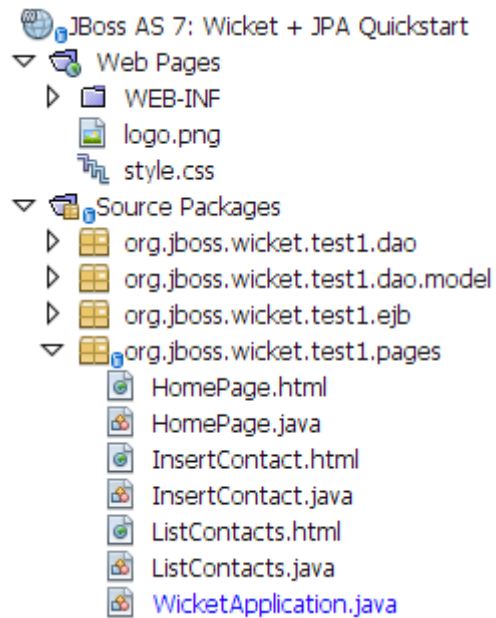
            getExceptionSettings().setUnexpectedExceptionDisplay(
                IExceptionSettings.SHOW_INTERNAL_ERROR_PAGE);
        }
    }
}
```



Wicket basic: Reusable components

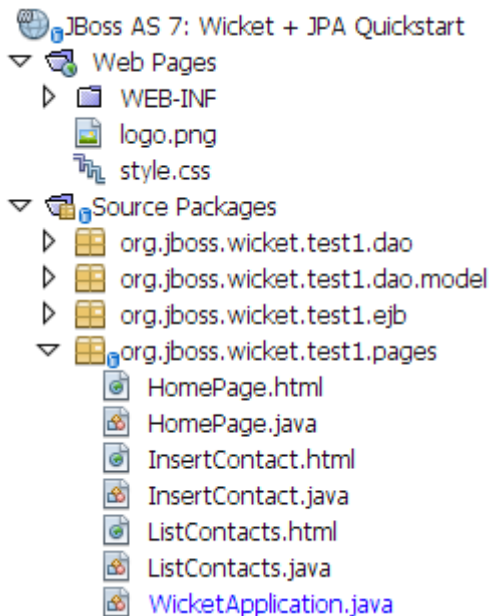


Wicket basic: Reusable components



Wicket basic: Creating reusable components

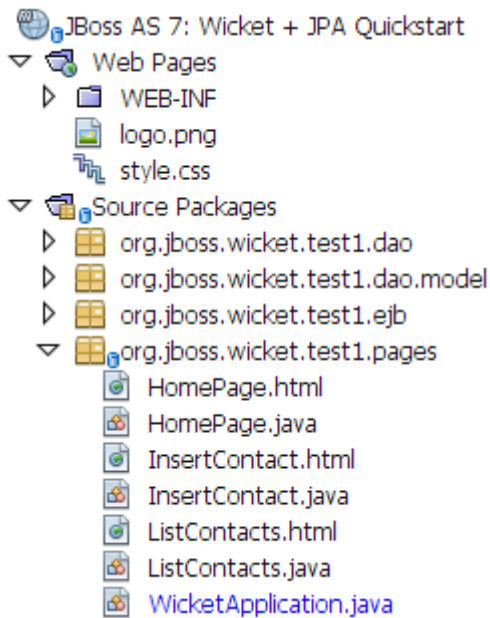
Navigation between pages



Wicket basic: Creating reusable components

Navigation between pages

AJAX



Wicket basic: Creating reusable components

Navigation between pages

AJAX

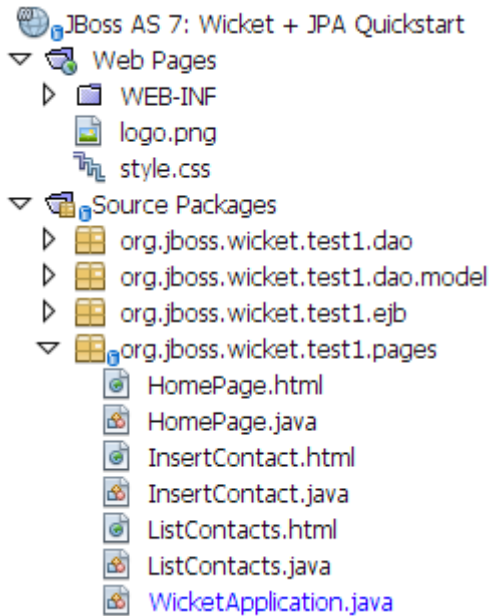


Sessions

- JBoss AS 7: Wicket + JPA Quickstart
 - Web Pages
 - WEB-INF
 - logo.png
 - style.css
 - Source Packages
 - org.jboss.wicket.test1.dao
 - org.jboss.wicket.test1.dao.model
 - org.jboss.wicket.test1.ejb
 - org.jboss.wicket.test1.pages
 - HomePage.html
 - HomePage.java
 - InsertContact.html
 - InsertContact.java
 - ListContacts.html
 - ListContacts.java
 - WicketApplication.java



Wicket basic: Creating reusable components



Navigation between pages



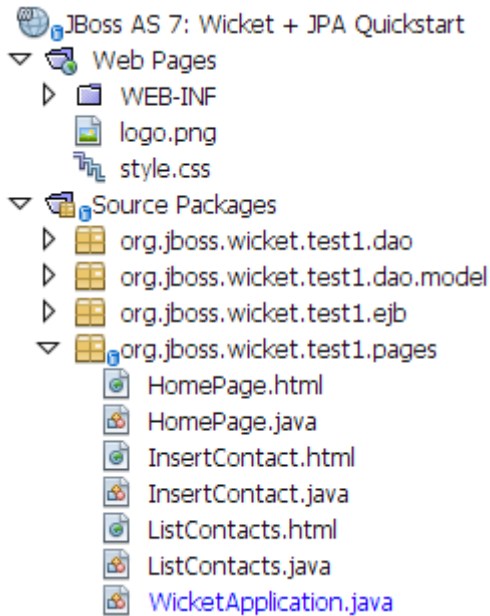
AJAX

Sessions

GWT-style AJAX-only apps



Wicket basic: Creating reusable components



Navigation between pages

Validators



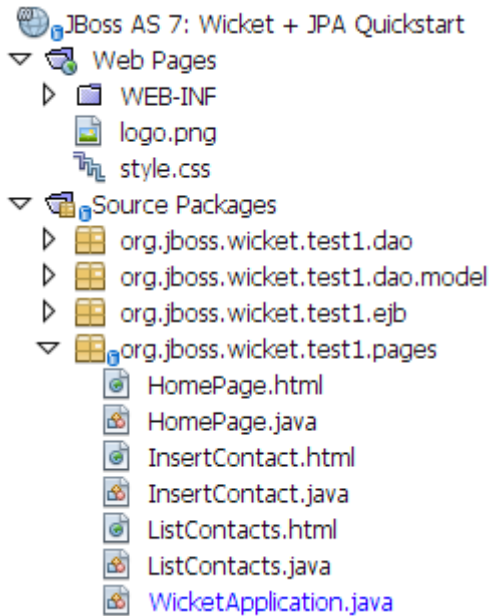
AJAX

Sessions

GWT-style AJAX-only apps



Wicket basic: Creating reusable components



Navigation between pages

Validators



Resources
(images, icons)

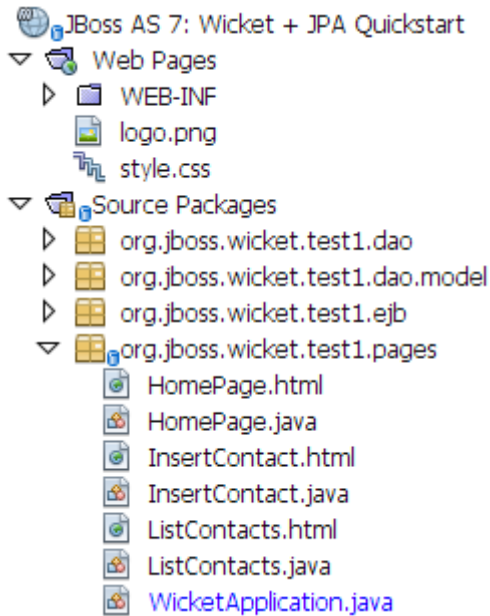
AJAX

Sessions

GWT-style AJAX-only apps



Wicket basic: Creating reusable components



Navigation between pages

Validators



Resources
(images, icons)

AJAX

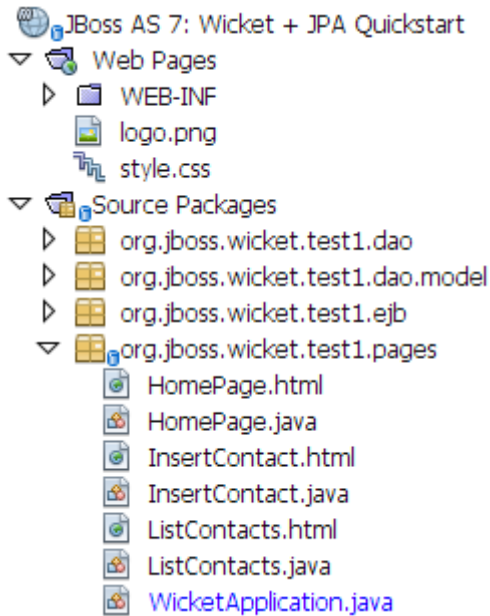
Sessions

GWT-style AJAX-only apps

Components hierarchy



Wicket basic: Creating reusable components



Navigation between pages

AJAX

Validators



Resources
(images, icons)

Sessions

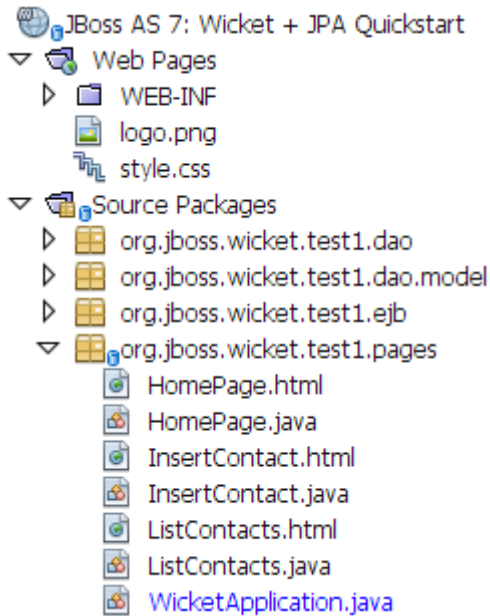
Pre-existing rich components

GWT-style AJAX-only apps

Components hierarchy



Wicket basic: Creating reusable components



Navigation between pages

CSS and JavaScript

AJAX

Validators



Resources
(images, icons)

Sessions

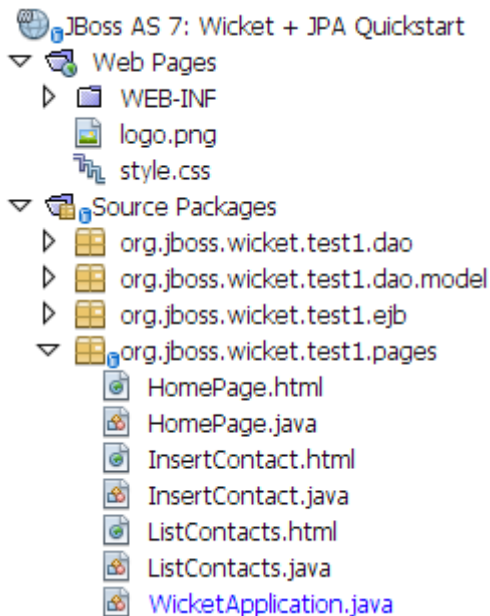
Pre-existing rich components

GWT-style AJAX-only apps

Components hierarchy



Wicket basic: Creating reusable components



Navigation between pages

CSS and JavaScript

Leveraging Java inheritance

AJAX

Validators



Resources
(images, icons)

Sessions

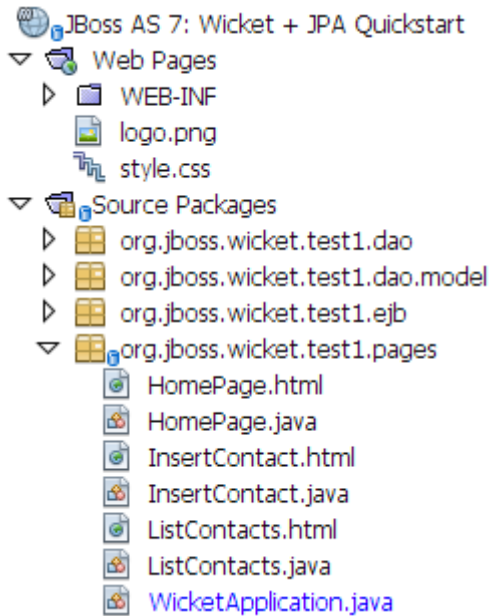
Pre-existing rich components

GWT-style AJAX-only apps

Components hierarchy



Wicket basic: Creating reusable components



Navigation between pages

CSS and JavaScript

Leveraging Java inheritance

AJAX

Validators



Resources
(images, icons)

Sessions

Pre-existing rich components

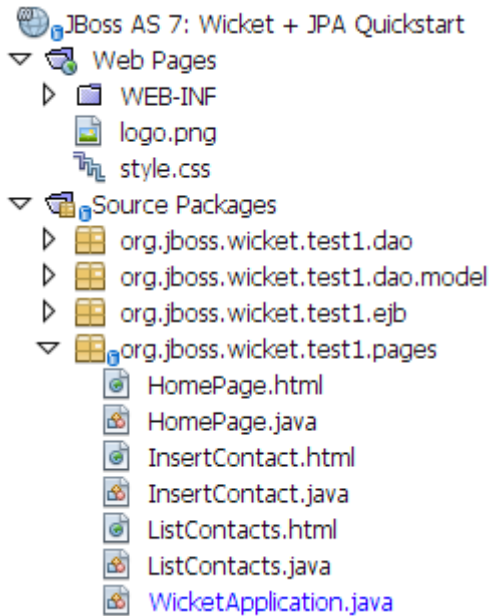
GWT-style AJAX-only apps

Localization

Components hierarchy



Wicket basic: Creating reusable components



Navigation between pages

Leveraging Java inheritance

Validators

(resources, icons)

Sessions

Components

GWT-style AJAX-only apps

Localization

Components hierarchy

**Come to JBug Brno
for more Wicket!**



Wicket vs. JSF

- JSF:
 - Inspired in JSP taglibs and Struts
 - Bunch of XML configs
 - Tries to pretend there's no HTML
 - Logic and UI binding by “magic”
 - Intranet oriented
- Wicket:
 - Inspired by XUL, Swing
 - No XML (except Java EE's)
 - Tries to pretend there's no HTTP
 - Logic and UI binding next to each other
 - Doesn't need a container → easily testable
 - Modern web app oriented



(JSF page)

```
<f:view>
  <f:form formName="logonForm">
    <h:panel_grid columns="2">
      <h:output_text value="Username:"/>
      <h:input_text id="username" length="16"
        valueRef="logonBean.username"/>
      <h:output_text value="Password:"/>
      <h:input_secret id="password" length="16"
        valueRef="logonBean.password"/>
      <h:command_button type="submit"
        label="Log On"
        actionRef="logonBean.logon"/>
      <h:command_button type="reset"
        label="Reset"/>
    </h:panel_grid>
  </f:form>
</f:view>
```



Wicket + Java EE 6 (with JBoss AS 7.1)

- AS 7:
 - Java EE 6 container
 - Starts faster than Tomcat (in ~3 seconds)
 - Deploys faster than Tomcat
 - Small memory footprint
- Java EE 6:
 - Much simpler than Java EE 5
 - One class per component is enough – POJOs
 - CDI (similar to *Spring JavaConfig*)



Wicket + Java EE 6 (with JBoss AS 7.1)

- Make Wicket work with Java EE injection:

```
public class WicketApplication extends WebApplication
{
    @Override
    public Class<HomePage> getHomePage() {
        return HomePage.class;
    }

    @Override
    public void init() {
        super.init();

        getComponentInstantiationListeners()
            .add(new JavaEEComponentInjector(this));
    }
}
```



Wicket + Java EE 6 (with JBoss AS 7.1)

- Make Wicket work with Java EE injection:

```
<dependency>
  <groupId>org.wicketstuff</groupId>
  <artifactId>wicketstuff-javaee-inject</artifactId>
  <version>${wicket.version}</version>
</dependency>
```

```
<!-- Import the CDI API, we use provided scope as the API is included in JBoss AS 7 -->
```

```
<dependency>
  <groupId>javax.enterprise</groupId>
  <artifactId>cdi-api</artifactId>
  <scope>provided</scope>
</dependency>
```

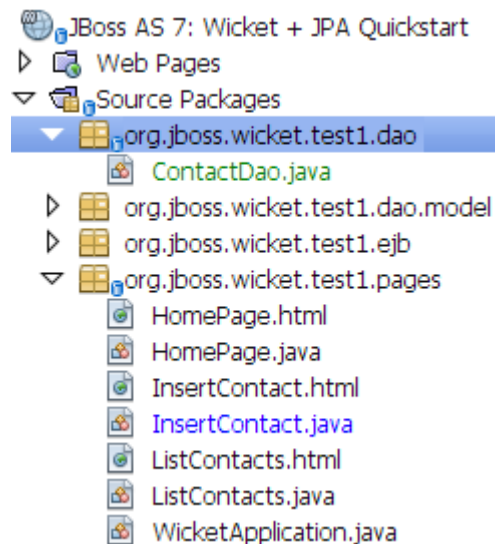
```
<!-- Import the Common Annotations API (JSR-250), we use provided scope as the API is included
```

```
<dependency>
  <groupId>org.jboss.spec.javax.annotation</groupId>
  <artifactId>jboss-annotations-api_1.1_spec</artifactId>
  <scope>provided</scope>
</dependency>
```

```
<!-- Import the EJB API, we use provided scope as the API is included in JBoss AS 7 -->
```

```
<dependency>
  <groupId>org.jboss.spec.javax.ejb</groupId>
  <artifactId>jboss-ejb-api_3.1_spec</artifactId>
  <scope>provided</scope>
</dependency>
```

Wicket + JPA (with JBoss AS 7.1)



```
@Stateless
public class ContactDao
{
    @PersistenceContext private EntityManager em;

    public List<Contact> getContacts() {
        return em.createQuery("SELECT c FROM Contact c").getResultList();
    }

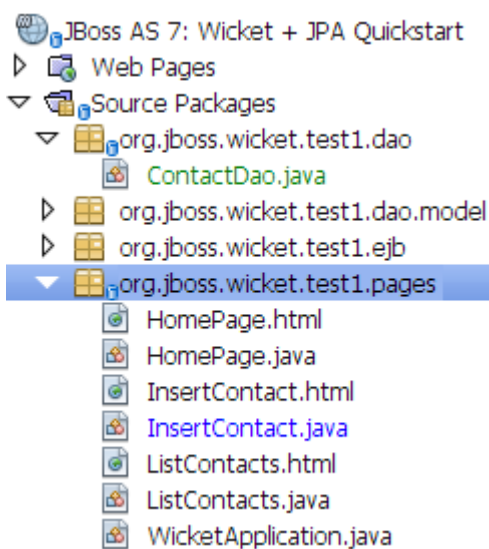
    public Contact getContact(Long id) {
        return em.find(Contact.class, id);
    }

    public void addContact(String name, String email) {
        em.merge(new Contact(null, name, email));
    }

    public void remove(Contact modelObject) {
        Contact managed = em.merge(modelObject);
        em.remove(managed);
        em.flush();
    }
}
```



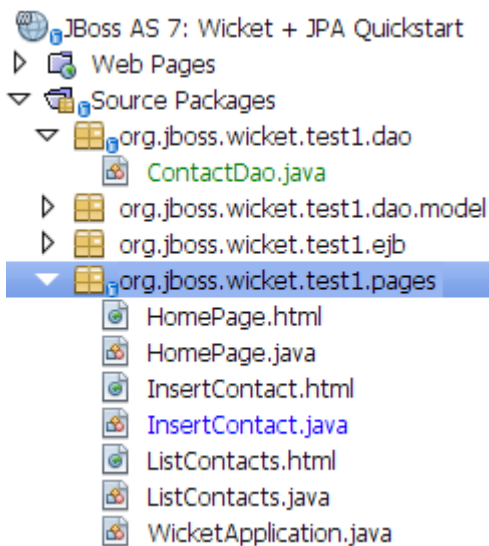
Wicket + EJB (with JBoss AS 7.1)



```
public class InsertContact extends WebPage {  
  
    private String name;  
    private String email;  
  
    @EJB private ContactDao contactDao;  
  
    public InsertContact() {  
        add( new Form<Contact>("insertForm") {  
            add(new TextField("name", new PropertyModel(this, "name")));  
            add(new TextField("email", new PropertyModel(this, "email")));  
        }  
        @Override protected void onSubmit() {  
            contactDao.addContact(name, email);  
            setResponsePage(ListContacts.class);  
        }  
    });  
    add(new FeedbackPanel("feedback"));  
}  
Get/Set  
}
```



Wicket page (expanded version)



```
public class InsertContact extends WebPage {

    private Form<Contact> insertForm;

    private String name;
    private String email;

    @EJB private ContactDao contactDao;

    public InsertContact() {

        this.insertForm = new Form<Contact>("insertForm") {
            @Override
            protected void onSubmit() {
                contactDao.addContact(name, email);
                setResponsePage(ListContacts.class);
            }
        };

        this.insertForm.add(new RequiredTextField<String>("name",
            new PropertyModel<String>(this, "name")))
            ;
        this.insertForm.add(new RequiredTextField<String>("email",
            new PropertyModel<String>(this, "email")));

        add(this.insertForm);
    }
    Get/Set
}
```



Questions?

ozizka@redhat.com



Thanks!

See you at JBug - monthly at FI MUNI.
Watch **JBoss.cz** for announcements.

Join *Czech & Slovak Wicket Users Group*
at [LinkedIn.com](https://www.linkedin.com/company/wicket-users-group/)



Wicket (slide template)

