WTP Update

Developing Java EE 5 Applications with WTP

Kaloyan Raev
SAP Labs Bulgaria

kaloyan.raev@sap.com
WTP Update

Developing Java EE 5 Applications with WTP

WTP and Java EE 5

Demo

A look into the future
WTP Update

Developing Java EE 5 Applications with WTP

WTP and Java EE 5

Demo

A look into the future
The Eclipse Web Tools Platform project

- **Web Tools**
  - Web and Java EE applications development
  - Editors
  - Wizards
  - API frameworks

- **Platform**
  - Provides infrastructure
  - Extensible for adopters
Java EE 5

- **Benefits**
  - Simplified business logic development
  - Simplified testing and dependency management
  - Simplified O/R persistence
  - Enhanced Web services programming model

- **Achieved by**
  - Use of annotated POJOs.
  - Reduced need for deployment descriptors
  - Use of resource injections
Java EE 5

Java EE 5 Compatible Implementations

- **Apache Geronimo-2.0.1**
- **WebLogic Server v10.0**
- **IBM WASCE 2.0**
- **Kingdee**
- **Oracle Application Server 11**
- **SAP NetWeaver 7.1**
- **Sun Microsystems**
- **TmaxSoft JEUSS 6**
- **GlassFish Application Server**
WTP and Java EE 5

- Major application server providers already have a runtime solution

- Eclipse WTP should satisfy the need for a design time solution
WTP and Java EE 5

- Java EE 5 in WTP Europa
  - Basic support
  - Extensible for adopters

- Adopters
  - Provide full featured support on top of WTP Europa
Java EE Perspective

- New name of the perspective
  - J2EE → Java EE
  - Other “J2EE” labels are not changed yet
Java EE Perspective

- Facets
  - New Java EE module facets and extensible models
    - EJB 3.0
    - Web 2.5
    - EAR 5.0
    - AppClient 5.0
  - XDoclet facet conflicts Java EE 5 module facets
Java EE Perspective

- Optional deployment descriptor
  - Chosen during project creation
  - Set on only for Web projects by default
EJB 3.0 Projects

- Project Navigator content
  - No meta data overview in the Project Navigator
  - Could be provided by adopters

- Enterprise Java Beans
  - No XDoclet EJBs
  - No wizards for creating EJBs
  - Wizards could be provided by adopters

- EJB 3.0 development utilizes JDT tools
  - Code completion
  - Quick fix
EJB 3.0 Projects

- Create new Session Bean
  - Create a new POJO
  - Annotate the class with @Stateless or @Stateful
  - Add bean attributes
EJB 3.0 Projects

- Create new Message Driven Bean
  - Create a new POJO
  - Annotate the class with @Message Driven
  - Add bean attributes

```java
package mdb;

import javax.ejb.MessageDriven;

@MessageDriven()
public class MD
```
EJB 3.0 Projects

- Add business interfaces
  - Declare that the bean *implements* the interface
  - Use quick fix to create the interface
EJB 3.0 Projects

- Add business interfaces – cont.
  - Annotate the interface with `@Local` or `@Remote`

```java
package test;

@Local - javax.ejb
@Local - org.jboss.annotation.ejb
@LocalBinding - org.jboss.annotation.ejb
@LocalHome - javax.ejb
@LocalHomeBinding - org.jboss.annotation.ejb
long

package test;

@Remote - javax.ejb
@RemoteBinding - org.jboss.annotation.ejb
@RemoteBindings - org.jboss.annotation.ejb
@RemoteHome - javax.ejb
@RemoteHomeBinding - org.jboss.annotation.ejb
@Resource - javax.annotation
```
EJB 3.0 Projects

- Add business methods
  - Declare the method in the business interface

```java
@Remote
public interface MyRemoteIFace {

    public int myBusinessMethod(String param);
}
```
EJB 3.0 Projects

- Add business methods – cont.
  - Go back to the bean class
  - Use quick fix to add the method declared in the interface

```java
@Stateless(name="MyBean")
public class StatelessBean implements MyRem
{
    // Add unimplemented methods
    // Make type 'StatelessBean' abstract

    @Override
    public int myBusinessMethod(String para)
    {
        // TODO Auto-generated method stub
        return 0;
    }
}
```
Java Persistence API

- Create a JPA enabled project
  - Activate the *Java Persistence* facet
  - Configure JPA settings
  - Any Java EE module with Java nature can be enabled with JPA

<table>
<thead>
<tr>
<th>Project Facet</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJB Module</td>
<td>3.0</td>
</tr>
<tr>
<td>EJBDocet (XDoclet)</td>
<td>1.2.3</td>
</tr>
<tr>
<td>Java</td>
<td>5.0</td>
</tr>
<tr>
<td>Java Persistence</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Web 2.5 Projects

- Project Navigator content
  - No meta data overview in the Project Navigator
  - Could be provided by adopters

- Servlet 2.5
  - Reused Servlet wizard
  - No XDoclet servlets
EJB 3.0 Projects

- Inject an EJB in a Servlet
  - Add build reference from the Web to the EJB project
  - Add reference in the Servlet to the business interface of the bean
  - Annotate the reference with `@EJB`
Web Services

- Creating and consuming Web Services
  - No UI changes
  - Meta data generation depends on the selected runtime
Java Server Faces

- Create a JSF enabled Web project
  - Activate the *JavaServer Faces* facet
  - Configure JSF settings

<table>
<thead>
<tr>
<th>Project Facet</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Axis2 Web Services</strong></td>
<td></td>
</tr>
<tr>
<td>Dynamic Web Module</td>
<td>2.5</td>
</tr>
<tr>
<td>Java</td>
<td>5.0</td>
</tr>
<tr>
<td><strong>Java Persistence</strong></td>
<td>1.0</td>
</tr>
<tr>
<td>JavaServer Faces</td>
<td>1.2</td>
</tr>
<tr>
<td>WebDdoclet (XDoclet)</td>
<td>1.2.3</td>
</tr>
</tbody>
</table>
Server Adapters

- Java EE 5 enabled server adapters
  - Open Source adapters bundled with WTP
  - Proprietary adapters from update sites
WTP Update

Developing Java EE 5 Applications with WTP

WTP and Java EE 5

Demo

A look into the future
Demo

- **Molecular mass** calculator
  - The *molecular mass* is computed as the sum of the individual masses of all the atoms in the molecule
  - *Atomic masses* are consumed from a public Web Service
    - [http://www.webservicex.net/periodictable.asmx](http://www.webservicex.net/periodictable.asmx)
  - Application UI will be build with Java Server Faces.

- Example
  - \( A_r(H) = 1.00797 \)
  - \( A_r(O) = 15.9994 \)
  - \( M_r(H_2O) = 18.01534 \)
Demo

- *periodictable* web service definition
  - **GetAtoms**
    - Get element
  - **GetAtomicWeight**
    - Get atomic weight by element name
  - **GetAtomicNumber**
    - Get atomic number by element name
  - **GetElementSymbol**
    - Get atomic symbol by element name
Demo

**EAR project**

**Web project**
- Presentation logic on JSF
- Consumes the EJB bean

**EJB project**
- Consumes the web service
- Consumes the util classes
- Feeds the atomic weights

**Utility project**
- Calculation and parsing algorithms
- Axis libraries

Application design

**periodictable web service**
WTP Update

Developing Java EE 5 Applications with WTP

WTP and Java EE 5

Demo

A look into the future
WTP 3.0 enhancements

- Java EE 5
  - New and improved wizards
  - Simplified models
  - Project Navigator content
  - Better experience with deployment descriptors

- [WTP 3.0 requirement wiki page](#)
WTP 3.0 enhancements

- Committed
  - Allow up-level spec version changes for Java EE facets
  - Simplified Java EE deployment descriptor model APIs
  - EJB 3.0 Session Bean and Message Driven Bean wizards
  - Generation of deployment descriptors for Java EE 5 projects
WTP 3.0 enhancements

- Proposed
  - Deployment descriptor editors for all Java EE 5 DD files
  - Java EE 5 module-specific validation support
  - Project Navigator content for Java EE models
  - Filter and Listener Wizards
  - Classpath entry publish/export support to handle most J2EE dependencies
WTP 3.0 enhancements

- Get involved!
  - Read and comment on the [WTP 3.0 requirement wiki page](#)
  - File new [enhancement requests](#) in Bugzilla