



NEW FEATURES OF JAVA ENTERPRISE EDITION 6

Zdeněk Troníček

Faculty of Information Technology

Czech Technical University in Prague



Java Enterprise Edition 6

- Dependency Injection
- Java Server Faces 2.0
- Bean Validation
- Enterprise Java Beans 3.1
- Java Persistence API 2.0
- JAX-RS (RESTful web services)

Dependency Injection

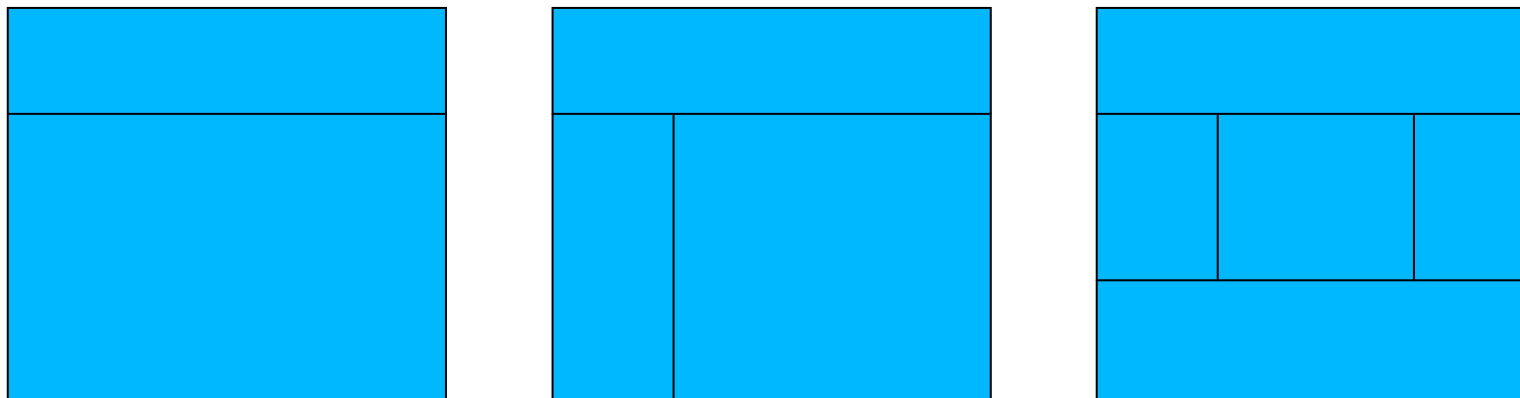
- Inject
- Qualifier
- Scope
- Singleton
- Named

Java Server Faces 2.0

- Templating
- Composite Components
- Ajax support
- State saving
- Implicit navigation
- GET support
- Bookmarkable URLs
- New scopes
- Configuration

Templating (Facelets)

- Use of XHTML
- Support for Facelets Tag Libraries
- Support for unified expression language
- Templating for components and pages



Template

...

```
<h:body>
```

```
  <div id="top" class="top">
```

```
    <ui:insert name="top">Library</ui:insert>
```

```
  </div>
```

```
  <div id="content" class="center_content">
```

```
    <ui:insert name="content">Content</ui:insert>
```

```
  </div>
```

```
</h:body>
```

...

Template Client

```
<ui:composition xmlns:ui="http://java.sun.com/jsf/facelets"
    template="BasicTemplate.xhtml"
    xmlns:h="http://java.sun.com/jsf/html">
  <ui:define name="content">
    <h:form>
      Author: <h:inputText value="#{searchBean.author}"/>
      Title: <h:inputText value="#{searchBean.title}"/>
      <h:commandButton value="search"
        action="#{searchBean.search}"/>
    </h:form>
  </ui:define>
</ui:composition>
```

Composite Components

Composite Component = a special type of template that acts as component

```
<!-- INTERFACE -->
<composite:interface>
  <composite:attribute name="mailto" />
  <composite:attribute name="submit" method-signature="java.lang.String submit()"/>
</composite:interface>
<!-- IMPLEMENTATION -->
<composite:implementation>
  <h:form>
    <h:inputText value="#{cc.attrs.mailto}"/>
    <h:commandButton value="submit" action="#{cc.attrs.submit}"/>
  </h:form>
</composite:implementation>
```

Use of Composite Component

```
<ui:composition xmlns:ui="http://java.sun.com/jsf/facelets"
  template="template1.xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:my="http://java.sun.com/jsf/composite/mycomp">
  <ui:define name="content">
    <my:email mailto="#{user.email}" submit="#{user.submit}"/>
  </ui:define>
</ui:composition>
```

Bookmarkable URLs

```
<h:link outcome="page2" value="go to page2">  
  <f:param name="username" value="#{user.name}"/>  
</h:link>
```

```
<h:button outcome="page2" value="go to page2">  
  <f:param name="username" value="#{user.name}"/>  
</h:button>
```

<http://localhost:8080/JSFApplication/faces/page2.xhtml?username=Eva>

New Scopes

- View scope is preserved until the user finishes interaction with the current view
- Flash scope is propagated across a single view transition

Configuration

```
<managed-bean>  
  <managed-bean-name>user</managed-bean-name>  
  <managed-bean-class>myjsf.UserBean</managed-bean-class>  
  <managed-bean-scope>session</managed-bean-scope>  
</managed-bean>
```

```
@ManagedBean  
@SessionScope  
public class UserBean {  
  //...  
}
```

Bean Validation

```
public class Account {  
    @NotNull  
    Long accountId;  
    @NotNull @Size(max=100)  
    String description;  
    //...  
}
```

- @NotNull, @Null
- @Size(min = 5, max = 10)
- @Max(255), @Min(1)
- @Past, @Future
- @Pattern(regexp = "...")

Enterprise Java Beans 3.1

- Session Beans
 - @Stateless, @Stateful, @Singleton
 - @Local, @Remote, web service, no interface
 - Timer service: @Schedule

Singleton Session Bean

@Singleton

```
//@ConcurrencyManagement(ConcurrencyManagementType.CONTAINER)
```

```
public class SingletonSessionBean {
```

```
    @Lock(LockType.READ)
```

```
    public void doSomething() {
```

```
        //...
```

```
    }
```

```
    @Lock(LockType.WRITE)
```

```
    public void doSomethingElse() {
```

```
        //...
```

```
    }
```

```
}
```

Singleton Session Bean (cont.)

@Singleton

@ConcurrencyManagement(ConcurrencyManagementType.BEAN)

```
public class SingletonSessionBean {
    public void doSomething() {
        //...
        synchronized(this) {
            //...
        }
    }
    public synchronized void doSomethingElse() {
        //...
    }
}
```

Timer Service

@Schedule arguments:

- year, month, dayOfMonth, dayOfWeek
- hour, minute, second
- timezone

```
@Schedule(hour = "2", minute = "30")  
public void cleanDatabase() {  
    //...  
}
```

Java Persistence API 2.0

- Lists and Maps
- Pessimistic locking
- Criteria API

Lists in Entities

```
@Entity
public class Book implements Serializable {
    @ElementCollection(fetch = FetchType.EAGER)
    @CollectionTable(name = "BOOK_AUTHORS")
    private List<String> authors;
    //...
}
```

BOOK_AUTHORS

BOOK_ID	AUTHORS

Maps in Entities

@Entity

```
public class Item implements Serializable {
```

```
    @ElementCollection @CollectionTable(name = "INFO")
```

```
    @MapKeyColumn(name = "NAME") @Column(name = "INFO")
```

```
    private Map<String, String> info;
```

```
    //...
```

```
}
```

INFO

ITEM_ID	NAME	INFO

Criteria API

```
// SELECT i FROM Item i WHERE i.weight > 50
CriteriaBuilder builder = em.getCriteriaBuilder();
CriteriaQuery<Item> query = builder.createQuery(Item.class);
Root<Item> root = query.from(Item.class);
Path<Integer> path = root.<Integer>get("weight");
Predicate cond = builder.gt(path, 50);
query.where(cond);
TypedQuery<Item> typedQuery = em.createQuery(query);
List<Item> result = typedQuery.getResultList();
```

Pessimistic Locking

```
Auction auction = em.find(Auction.class, auctionId);  
em.lock(auction, LockModeType.PESSIMISTIC_READ);
```

LockModeType

- PESSIMISTIC_READ
- PESSIMISTIC_WRITE
- OPTIMISTIC
- ...

RESTful Web Services

Representational State Transfer (REST):

- Resources identified through URI
- Uniform interface (GET, POST, PUT, DELETE)
- Self-descriptive messages
- Stateful interaction through hyperlinks

JAX-RS

- @Path, @GET, @POST, @PUT, @DELETE, @HEAD
- @PathParam, @QueryParam
- @Consumes, @Produces

```
@Path("/book/{id}")
public class Book {
    @GET @Produces("application/xml")
    public String getXml(@PathParam("id") int id) {
        ...
    }
    ...
}
```



QUESTIONS & ANSWERS

Zdeněk Troníček
tronicek@fit.cvut.cz