



Web Engineering

Implementation

Prof. Dr. Alexander Knapp

Dr. Nora Koch

SS 07 (6) – 18.06.07

Ludwig-Maximilians-Universität München

- **Echo 2**
- **Google Web Toolkit**

- **Struts**

- **Java Persistence API**

- **Development of dynamic Web sites and Web applications**

- **Integration on all layers**
 - model-view-controller
 - database access (object-relational mapping)

- **Features**
 - templates
 - validation
 - different renderers
 - automatic configuration
 - introspection
 - security (authentication, authorization)
 - Ajax
 - internationalisation, localisation
 - bookmarking

http://en.wikipedia.org/wiki/List_of_web_application_frameworks

- **Pure Java development**
 - servlet-based (“Web application container” layer)
 - application code executed on the server
 - client-side rendering engine in JavaScript and CSS

- **Event-driven Web applications**
 - based on Ajax technology
 - incremental updates for reducing bandwidth consumption

- **User interface component library**
 - comparable to Swing, SWT, ...
 - thin-client UI technology

<http://www.nextapp.com/platform/echo2/echo/>

Echo 2: Example (1)

```
package echo2helloworld;
import nextapp.echo2.app.ApplicationInstance;
import nextapp.echo2.webcontainer.WebContainerServlet;
public class Servlet extends WebContainerServlet {
    @Override
    public ApplicationInstance newApplicationInstance() {
        return new App();
    }
}
```

```
package echo2helloworld;
import nextapp.echo2.app.*;
public class App extends ApplicationInstance {
    private Window mainWindow;
    @Override
    public Window init() {
        mainWindow = new Window(); mainWindow.setContent(new Form());
        return mainWindow;
    }
}
```

Echo 2: Example (2)

```
class Form extends ContentPane {
    private TextField helloText;
    private Label helloOut;
    private Button submit;
    public Form() {
        super(); initForm();
    }
    private void initForm() {
        Column col = new Column(); this.add(col);
        helloOut = new Label("Hello World!"); col.add(helloOut);
        Row row = new Row();
        Label nameLabel = new Label("Enter your name:");
        helloText = new TextField(); row.add(nameLabel);
        row.add(helloText); col.add(row);
        submit = new Button("Submit"); col.add(submit);
        submit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                helloOut.setText("Hello " + helloText.getText() + "!"); } });
    }
}
```

Echo 2: Evaluation

- **Application domain**
 - intra-net applications
 - rich client interface
- **Open source** (Mozilla Public License)
 - commercial visual development environment EchoStudio2
- **Separation of concerns left to developer**
 - content, navigation, presentation
 - open to various middle-tier technologies
- **Client dependency**
 - support for Internet Explorer, Mozilla, Safari
 - but different rendering
- **Proprietary UI component framework**

Google Web Toolkit (GWT)

- **Pure Java development**
 - application code executed on the client
 - translation of Java into JavaScript
 - restrictions on Java library classes
 - thick client
- **User-interface component library**
 - comparable to Swing, SWT, ...
- **Not fully open-source**
 - binary only Java-JavaScript cross-compiler

`http://code.google.com/webtoolkit/`

- **Open-source Java Web development framework**
 - based on servlets
 - encourages use of MVC 2
 - support for localisation, validation, ...

- **History**
 - originally developed by Craig R. McClanahan
 - donated to the Apache Software Foundation in 2000
 - top-level Apache project since 2005
 - Struts joined forces with WebWork in 2005: Struts 2

`http://struts.apache.org`

Struts 2: Features

- **Model-view-controller** (front controller, Model 2)
 - action-based framework
 - POJO-based actions (facilitating testing)

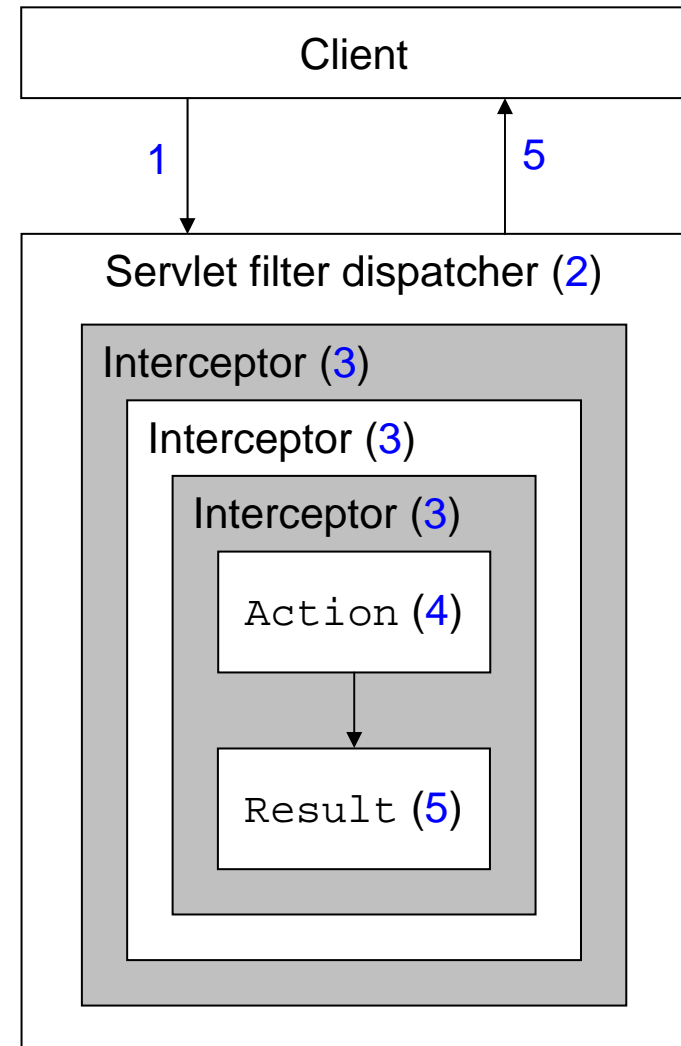
- **Middle-tier**
 - Spring, SiteMesh and Tiles integration

- **Presentation**
 - OGNL expression language integration
 - themes based tag libraries and Ajax tags
 - multiple view options (JSP, Freemarker, Velocity and XSLT)
 - validators

- Plug-ins to extend and modify framework features
- Annotation and XML configuration options

Struts 2: Basic Architecture

1. Web browser requests a resource (e.g., `/mypage.action`, `/reports/myreport.pdf`)
2. filter dispatcher looks at the request and determines the appropriate `Action`
3. interceptors automatically apply common functionality to the request, like workflow, validation, and file upload handling
4. `Action` method executes, usually storing and/or retrieving information from a database
5. result renders the output to the browser, be it HTML, images, PDF, ...



Struts 2: Example (1)

index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
  <meta http-equiv="Refresh" content="0;URL=demo/StrutsDemo.action">
</head>
<body><p>Loading ...</p></body>
</html>
```

Struts 2: Example (2)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_9" version="2.4"
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>Struts Demo</display-name>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

web.xml

**master filter**

- executing actions
- serving static content
- kicking off interceptor chain for the request lifecycle
- should serve /*

Struts 2: Example (3)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <package name="demo" namespace="/demo" extends="struts-default">
    <action name="StrutsDemo"
      class="de.lmu.ifi.pst.strutsdemo.StrutsDemo">
      <result>demo.jsp</result>
    </action>
  </package>
</struts>
```

struts.xml
(on Web application's class path)

in /demo

default result type: Dispatcher result; handles JSP


Struts 2: Example (4)

```
<%@ page language="java"
    contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
    <meta http-equiv="Content-Type"
        content="text/html; charset=ISO-8859-1">
    <title>Struts Demo</title>
</head>
<body>
    <h2><s:property value="message"/></h2>
    <p>Current date and time is: <b>
    <s:property value="currentTime"/></b>
</body>
</html>
```

demo.jsp

access to Java bean property



Struts 2: Example (5)

```
package de.lmu.ifi.pst.strutsdemo;
import com.opensymphony.xwork2.ActionSupport;
public class StrutsDemo extends ActionSupport {
    public static final String MESSAGE = "Struts Demo";
    public String execute() throws Exception {
        setMessage(MESSAGE);
        return SUCCESS;
    }

    private String message;

    public void setMessage(String message){
        this.message = message;
    }
    public String getMessage() {
        return message;
    }
    public String getCurrentTime(){
        return new java.util.Date().toString();
    }
}
```

StrutsDemo.java

can also be a POJO

Java bean properties
naming convention

Struts 2 Validation: Example (1)

```
<%@ page language="java" contentType="text/html;
    charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <title>Login</title>
</head>
<body>
<s:form action="Logon">
    <s:textfield label="User Name" name="username" />
    <s:password label="Password" name="password" />
    <s:submit />
</s:form>
</body>
</html>
```

logon.jsp



Struts 2 Validation: Example (2)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators
PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
"http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
  <field name="username">
    <field-validator type="requiredstring">
      <message>Username is required</message>
    </field-validator>
  </field>
  <field name="password">
    <field-validator type="requiredstring" short-circuit="true">
      <message>Password is required</message>
    </field-validator>
    <field-validator type="fieldexpression">
      <param name="expression">
        <![CDATA[ password.length() > 5 ]]></param>
      <message>Password must have at least 6 characters.</message>
    </field-validator>
  </field>
</validators>
```

Logon-validator.xml
(in the same director as Logon class)

int, date, email, ...

stop validation early

OGNL expression

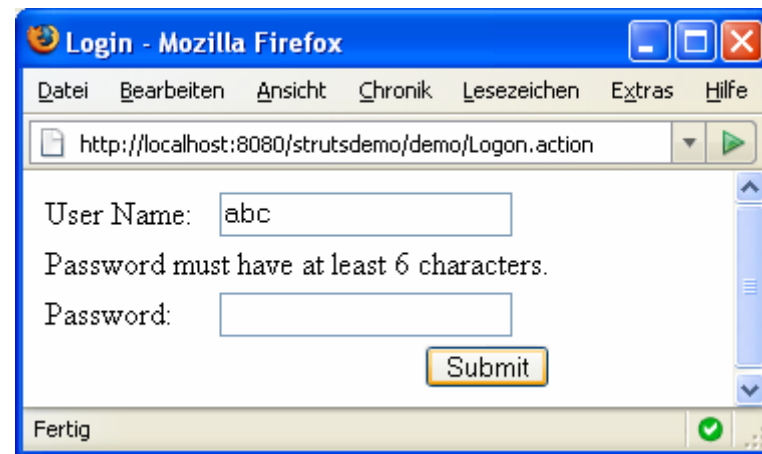
Struts 2 Validation: Example (3)

```
<action name="Logon_*" method="{1}"
        class="de.lmu.ifi.pst.strutsdemo.Logon">
  <result type="redirect-action">menu</result>
  <result name="input">logon.jsp</result>
</action>
```

struts.xml

```
<action name="*" >
  <result>{1}.jsp</result>
</action>
```

← default redirection



Java Persistence API (JPA)

- **Java framework for handling relational data (JSR 220)**
 - object/relational meta data
 - Java persistence query language
 - `javax.persistence`
- **Implementations**
 - Hibernate (www.hibernate.org)
 - Glassfish (glassfish.java.net)
 - Oracle TopLink (www.oracle.com/technology/products/ias/toplink)
 - ...

JPA: Entities

```

package model;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
@Entity
public class Person {
    @Id
    @GeneratedValue
    private Integer id;
    private String lastName;
    private String firstName;

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    ...
}

```

← declaring an entity (database table, override name by @Table)
 ← unique identifier (primary key), generated by database
 ← persistent fields (override by @Transient, override name by @Column)

```


CREATE TABLE Person (
    id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    firstName VARCHAR(45) NOT NULL,
    lastName VARCHAR(45) NOT NULL,
    PRIMARY KEY(id))

```

- Synchronisation with database

```
javax.persistence.EntityManager em;  
  
Person person = new Person();  
person.setFirstName("Norman");  
person.setLastName("Miller");  
em.persist(person);  
  
em.remove(person);  
  
Query query = em.createQuery(  
    "select p FROM Person p WHERE p.firstName = 'Norman'");  
query.getResultList();
```

JPQL query



JPA: Persistence Unit

- **Entity classes form a logical unit**

- database connection for each unit stored in `persistence.xml`
 - transaction management
- may also list entities; JPA implementations scan application for entities

```

<persistence
  xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">
  <persistence-unit name="punit">
  <provider>org.apache.openjpa.persistence.PersistenceProviderImpl</provider>
  <properties>
    <property name="openjpa.ConnectionURL" value="jdbc:hsqldb:database"/>
    <property name="openjpa.ConnectionDriverName" value="org.hsqldb.jdbcDriver"/>
    <property name="openjpa.ConnectionUserName" value="sa"/>
    <property name="openjpa.ConnectionPassword" value=""/>
  </properties>
  </persistence-unit>
</persistence>

```

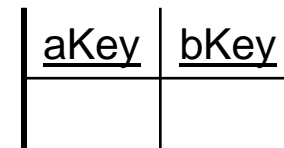
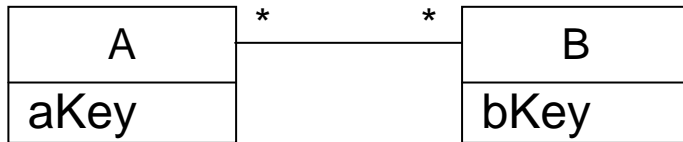
persistence.xml
(in META-INF/)

properties depending on provider

Object-Relational Mappings: Associations

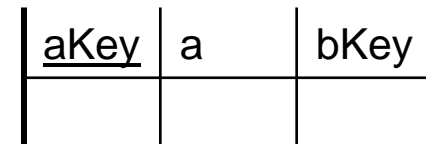
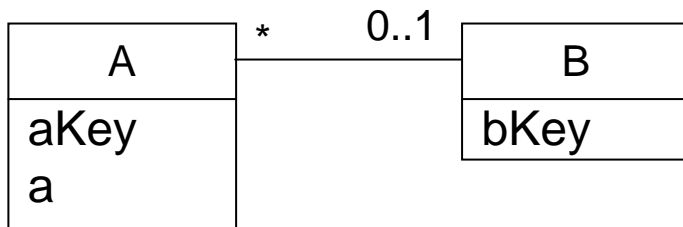
■ Many-to-many

- association in a separate table with primary keys
- JPA: `@ManyToMany` (details set with `@JoinTable`)

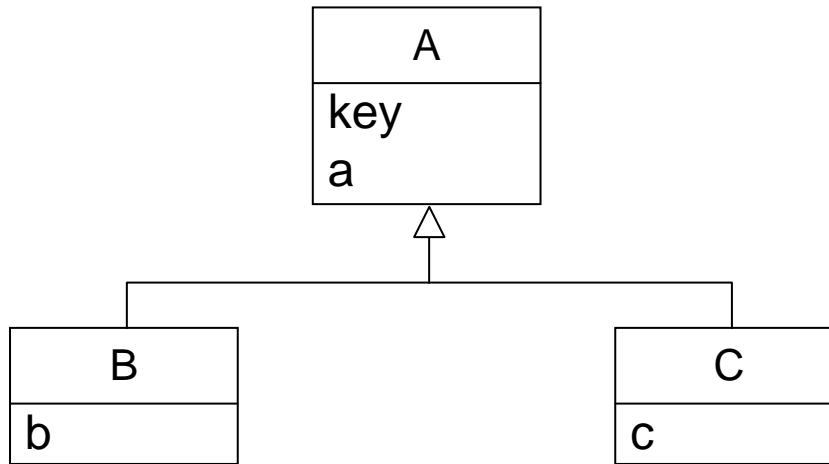


■ Many-to-one

- primary key of B as foreign key in table for A
- JPA: `@ManyToOne(optional=b)` (details set with `@JoinColumn`), opposite end specified by `@mappedBy`
- one-to-many, and one-to-one similar



Object-Relational Mappings: Inheritance (1)



1. Table per class

- several tables have to be visited for sub-classes
- JPA: `@Inheritance(strategy=JOINED)`

Table A

<u>key</u>	a

Table B

<u>key</u>	b

Table C

<u>key</u>	c

Object-Relational Mappings: Inheritance (2)

2. One table per class, inherited attributes added

- changes to super-class have to mirrored in all tables
- JPA: `@Inheritance(strategy=TABLE_PER_CLASS)`

Table A

<u>key</u>	a

Table B

<u>key</u>	a	b

Table C

<u>key</u>	a	c

3. Single table

- null values in unused columns
- JPA: `@Inheritance(strategy=SINGLE_TABLE), discriminator DTYPE`
 - default

Table ABC

<u>key</u>	a	b	c	discriminator


Dependency Injection (DI)

- **Inversion of control (IOC) for object creation and linking**
 - objects do not create other objects they depend on themselves
 - dependencies resolved by factories or frameworks
 - Spring, PicoContainer, ...
 - constructor, setter, and interface injection

- **Loose coupling**
 - facilitates testing (mock-ups, ...) and maintenance
 - alternative: [service locators](#)

Dependency Injection with Spring (1)

```
package service;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import model.Person;

public class PersonServiceImpl implements PersonService {
    private EntityManager em;
    @PersistenceContext  inject EntityManager on instantiation
    public void setEntityManager(EntityManager em) { this.em = em; }
    private EntityManager getEntityManager() { return em; }

    public void save(Person person) {
        if (person.getId() == null) { em.persist(person); }
        else { em.merge(person); }
    }
    public Person find(int id) {
        return em.find(Person.class, id);
    }
    ...
}
```

Dependency Injection with Spring (2)

- **Install context listener for enabling Spring functionality**
 - loads `WEB-INF/applicationContext.xml`

```
...                                     web.xml
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
...
```

Dependency Injection with Spring (3)

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans">
  <bean class=
    "org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor"/>
  <bean id="entityManagerFactory" class=
    "org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="jpaVendorAdapter">
      <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
        <property name="database" value="MYSQL" />
        <property name="showSql" value="true" />
        <property name="generateDdl" value="false" />
      </bean>
    </property>
  </bean>
  <bean id="dataSource" class=
    "org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost/quickstart" />
    <property name="username" value="root" />
    <property name="password" value="root" />
  </bean>
</beans>
```

applicationContext.xml

make Spring JPA container, inject EntityManager

Hibernate JPA implementation

do not create DDL script (CREATE TABLE)

database connection (MySQL)

References

- Ian Roughley. *Starting Struts2*. C4Media, 2007.
<http://www.infoq.com/minibooks/starting-struts2>
- Martin Fowler. *Inversion of Control Containers and the Dependency Injection pattern*. 2004.
<http://martinfowler.com/articles/injection.html>
- OpenJPA manual.
<http://openjpa.apache.org/docs/latest/manual/manual.html>
- OGNL Technology. <http://www.ognl.org/>