CSE 135

The MVC Design Pattern & The Struts Framework

Previous Attempts: Model 1 Design Pattern

for every JSP page \( p \)
for every type of request \( r \) to \( p \)
insert in \( p \) code to implement the action requested by \( r \)

students.jsp
If request to insert student perform SQL INSERT
If request to delete student perform SQL UPDATE
If request to update student perform SQL DELETE

HTML part of the JSP
INSERT STUDENT
UPDATE STUDENT
DELETE STUDENT

http://.../students.jsp?action=insert&...
http://.../students.jsp?action=update&...
http://.../students.jsp?action=delete&...)
The MVC Design Pattern: Separating Model, View & Controller

- Development “Best Practice”
- Known well before web items
  - Smalltalk pioneered
- **Model**: Access to Underlying Databases and Info Sources
- **Controller**: Control Flow of Web App
- **View**: Look-and-Feel

The MVC Design Pattern

- MVC originated as Model 2 in web developers community

- **Model 1**: Application logic is attached to JSPs
  - Similar to previous attempts of students.jsp

- **Model 2**: Data access and control flow decisions in Java Beans
**Data Entry Example – MVC Attempt**

**students.jsp**

HTML part of the JSP

- INSERT STUDENT
- UPDATE STUDENT
- DELETE STUDENT

Controller/Actions

- Delete Student
- Update Student
- Insert Student

Model Java classes export methods that encapsulate SQL access

DB

---

**The Process and the Frictions**

**Business Process Owner (Client)**

**COMMUNICATION**

business process and specification of Web application

Target of Phase 1

**Chief Architect/Technical Project Leader**

**COMMUNICATION**

technical specification and development

Target of Phase 3

**Developer**

Problem is even worse in evolution phase when application logic is hidden in thousands of lines of code
Struts

- **Black-Box Framework Implementing MVC**
  - Framework: reusable “partial” application
- **Struts** `ActionServlet` provides high level control of workflow (i.e., “what happens next”)
- You provide Beans and files to customize framework according to your application needs
  1. JSPs provide HTML presentation (View)
  2. `ActionForm` Beans “collect” form data (Part of Controller)
  3. Action Beans provide details of flow (Part of Controller)
  4. `struts-config.xml` declares Beans and JSPs

How To Develop Struts Applications

**From 10 Miles High:**
- Pass high-level control to `ActionServlet`
  - By appropriate URL mapping in `web.xml`
- Design “workflow” in diagrams and then code it in `struts-config.xml`
- Develop `ActionForm` Beans responsible for collecting data
- Develop `Action` Beans responsible for changing the state of the application and deciding next step
- Develop Model Beans (not part of Struts) responsible for encapsulating database access
- Develop HTML and JSP pages for user interface
Struts application lifetime: How is a browser request processed?

Struts Single Request Processing (cont’d)

1. When web app is loaded, ActionServlet parses struts-config.xml and associates URL paths with Action and ActionForm Beans
   - Location of struts-config.xml is given in web.xml
2. The user issues an HTTP request from an initiating page to the ActionServlet
Struts Single Request Processing (cont’d)

3 The ActionServlet instantiates the ActionForm Bean associated with the HTTP request URL in struts-config.xml, and sets its properties using the HTTP request parameters (user-submitted data)

4 The ActionForm Bean validates its property values and if validation fails, ActionServlet responds with the initiating page P displaying appropriate error messages for the user to correct his/her form data

Struts Single Request Processing (cont’d)

5 If validation succeeds, the ActionServlet instantiates the Action Bean associated with the HTTP request URL in struts-config.xml, and calls its execute method passing as parameters the ActionForm Bean, the HTTP request and the HTTP response objects
**Struts Single Request Processing (cont’d)**

6. Within its **execute** method, the **Action Bean**
   instantiates/calls Model Beans, which open a connection to the database, execute SQL operations, and return sets of tuples.
   The **Action Bean** places the sets of tuples in the session so that JSP pages (View components) can access them.

**Struts Single Request Processing (cont’d)**

7. The **Action Bean** returns to the **ActionServlet**
   one of the **ActionForwards** with which the HTTP request URL is associated in **struts-config.xml**
   An **ActionForward** is a possible outcome of the **Action Bean** and represents either an JSP/HTML page or another **Action** that will be the response to the user’s request.
   Upon receiving the **ActionForward**, the **ActionServlet** responds to the user’s request with the corresponding JSP/HTML page or **Action**.
Install Struts

- We will use Struts 1.3 for Phase 2 of the project
  - Struts 2 will be covered later on and will not be used for the project
- Download **struts-1.3.10-all.zip**
- Struts is only a package containing:
  \doc, \src, \lib, \apps
- Within \apps is a set of *.war files
  - struts-blank-1.3.10.war
  - struts-examples-1.3.10.war
  - struts-cookbook-1.3.10.war

Struts Examples

- To play with Struts examples:
  - Copy struts-cookbook-1.3.10.war under \webapps
  - Access http://localhost:8080/struts-cookbook-1.3.10/
- To play with more Struts examples:
  - Copy struts-examples-1.3.10.war under \webapps
  - This automatically deploys a new web app directory
  - Access http://localhost:8080/struts-examples-1.3.10/
- To start your own Struts application:
  - Copy struts-blank-1.3.10.war under \webapps
  - Rename \struts-blank-1.3.10 to \your_app_name
Pass Control to ActionServlet

URLs of requests ending with ".do", pass all .do requests to ActionServlet

**web.xml**

```xml
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

Let's reconsider the “students” example

![Moodle interface with a table of student data]
showStudents.do Request Processing

showStudents.do Configuration

```
<struts-config>
  ...  
  <action-mappings>
    <action
      path="/showStudents"
      type="dataentry.actions.ShowStudentsAction">
      <forward
        name="success"
        path="/pages/students.jsp"/>
    </action>
    ...  
  </action-mappings>
  ...  
</struts-config>
```
showStudents.do Action Bean

ShowStudentsAction.java
package dataentry.actions;

import javax.sql.RowSet;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import dataentry.model.StudentModel;
...

public class ShowStudentsAction extends Action {
    ...

    public ActionForward execute(
        ActionMapping mapping, ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response)
        throws DBException {

        // retrieve all students
        RowSet crsStudents = StudentModel.getAllStudents();
        // store the RowSet in the request scope
        request.setAttribute("crsStudents", crsStudents);

        return mapping.findForward("success");
    }
}

showStudents.do Action Bean

ShowStudentsAction.java (cont’d)
...
public ActionForward execute(
    ActionMapping mapping, ActionForm form,
    HttpServletRequest request,
    HttpServletResponse response) throws DBException {

    // retrieve all students
    RowSet crsStudents = StudentModel.getAllStudents();
    // store the RowSet in the request scope
    request.setAttribute("crsStudents", crsStudents);

    return mapping.findForward("success");
}
showStudents.do Model Bean

StudentsModel.java
package dataentry.model;
...
public class StudentModel {
    private static String selectStr = "...;"
    private static String insertStr = "...;"
    private static String updateStr = "...;"
    private static String deleteStr = "...;"

    public static CachedRowSet getAllStudents() {...}
    public static void insertStudent(StudentBean student) {...}
    public static void updateStudent(StudentBean student) {...}
    public static void deleteStudent(StudentBean student) {...}
}

showStudents.do ActionForward

students.jsp
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
...
<%-- -------- Iteration Code -------- --%>
<%= RowSet crsStudents = (RowSet) request.getAttribute("crsStudents");
    while (crsStudents.next()) { %>
<tr>
    ...<td>
        <html:text property="middle" size="15"
            value="<%= crsStudents.getString("middleName") %>" />
    </td>
    ...</tr>
<% } %>
...
**insertStudent.do Request Processing**

1. **View**
   - Students.jsp
   - Form Validation Error

2. **ActionServlet**
   - InsertStudentAction
   - StudentFormInsertUpdate

3. **Controller**
   - ActionServlet
   - InsertStudentAction

4. **Model**
   - DB

5. **struts-config.xml**

6. **success**
   - students.jsp

7. **showStudents.do**

---

**insertStudent.do Configuration**

```xml
<form-bean name="studentFormInsertUpdate"
    type="dataentry.forms.StudentFormInsertUpdate"/>

<action
    path="/insertStudent"
    type="dataentry.actions.InsertStudentAction"
    validate="true"
    scope="request"
    input="/showStudents.do"
    name="studentFormInsertUpdate">
    <forward name="success" path="/showStudents.do"
        redirect="true"/>
</action>
```
**StudentFormInsertUpdate.java**

```java
package dataentry.forms;
...
public class StudentFormInsertUpdate extends ActionForm {

    private String id = null;
    private String first = null;
    private String middle = null;
    private String last = null;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }
    ...
```

/**
 * Reset all properties to their default values.
 */
public void reset(ActionMapping mapping,
    HttpServletRequest request) {
    setId(null);
    setFirst(null);
    setMiddle(null);
    setLast(null);
}
...
StudentFormInsertUpdate.java (cont’d)

```java
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {

    ActionErrors errors = new ActionErrors();

    if ((id == null) || (id.length() < 1))
        errors.add("idMsgTag1",
                new ActionMessage("errors.required", "ID"));
...

    return errors;
}
```

InsertStudentAction.java

```java
public class InsertStudentAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) throws DBException {

        // cast the form
        StudentFormInsertUpdate iForm = (StudentFormInsertUpdate) form;

        // insert the student
        StudentModel.insertStudent(iForm);

        return mapping.findForward("success");
    }
}
```
students.jsp

```html
<!-- in case form validation fails -->
<html:errors />

<!-- in case form validation fails -->
<html:form action="/insertStudent">
  <td><html:text property="id" size="10" /></td>
  <td><html:text property="first" size="15" /></td>
  <td><html:text property="middle" size="15" /></td>
  <td><html:text property="last" size="15" /></td>
  <td><html:submit value="Insert" /></td>
  <td><html:reset /></td>
</html:form>
```

struts-config.xml Structure

```xml
<struts-config>
  <!-- Form Bean Definitions -->
  <form-beans>...

  <!-- Global Exception Definitions -->
  <global-exceptions>...

  <!-- Global Forward Definitions -->
  <global-forwards>...

  <!-- Action Mapping Definitions -->
  <action-mappings>...

  <!-- Message Resources Definitions -->
  <message-resources parameter="MessageResources" />
</struts-config>
```
Global Exceptions

**struts-config.xml**

```xml
<!-- ------------------ Global Exception Definitions -->
<global-exceptions>
  <exception key="error.db"
    type="dataentry.db.DBException"
    path="/pages/dbException.jsp"/>
</global-exceptions>
```

**DBException.java**

```java
package dataentry.db;

public class DBException extends Exception {

  public DBException() {
    super();
  }

  public DBException(String message) {
    super(message);
  }
}
```
Global Exceptions

StudentModel.java

```java
public static void insertStudent(
    StudentFormInsertUpdate student) throws DBException {

    try {
        ...
    } catch (SQLException ex) {
        throw new DBException(ex);
    } catch (NamingException ex) {
        throw new DBException(ex);
    }
}
```

Global Exceptions

InsertStudentAction.java

```java
public class InsertStudentAction extends Action {

    public ActionForward execute(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response) throws DBException {

        ...
        StudentModel.insertStudent(...);
        ...
    }
}
```
Global Exceptions

**dbException.jsp**

```html
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>

<html>
<body>
  <h2>Database Exception</h2>
  ... 
  <h3>Here is the message generated by the thrown database exception:</h3>
  <p><html:errors /></p>
</body>
</html>
```

Global Forwards

**struts-config.xml**

```xml
<!-- ====================== Global Forward Definitions -->
<global-forwards>
  <forward name="showStudents" path="/showStudents.do"/>
</global-forwards>
```

**menu.jsp**

```jsp
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<b>Data Entry Menu</b>
<ul>
  <li><html:link forward="showStudents">Students</html:link></li>
  ... 
</ul>
```
Message Resources

MessageResources.properties

# -- app --
app.title=Struts Data Entry Application
...

students.jsp

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<html>
  <head>
    <title><bean:message key="app.title" /></title>
  </head>
  ...
</html>