JobTracker with Struts2

Eduard Gamonal
College of Engineering
Northeastern University
Boston, MA
gamonale@husky.neu.edu

December 13, 2012

Abstract

This report explains the development of JobTracker. First it offers an overview of the definition and scope of the project. Then details about the architecture and the internal design. After that, a description of some of the challenges faced. To conclude we offer a summary of the development process and the product.

Introduction

Development of complex websites requires some tedious tasks that are not related to business logic, like security checks, control of parameters, restrict access to certain sections, etc. Struts2 is a framework developed by the Apache Foundation that can help on these tasks.

A jobs crisis affects young graduates. Can software help them find a job? Most of them apply to different companies at the same time. Keeping track of what resume and cover letter is sent, to who and when, guarantees that this person will not get confused during an interview. The user can review the tracked activity before meeting the employer.

This project uses Struts2 to create an application to track job applications and opportunities.

Technology has penetrated in all sectors of the society. People use these new tools in many different ways depending on the task they want to accomplish, their education, their environment or even their physical conditions, amongst others.

We open this report with an overview of the definition and scope of the project. Then, details about the architecture and the internal design. After that, a description of some of the challenges
faced. To conclude we offer a summary of the development process and the product.

**Problem**

Job applicants often use different resumes and tailored cover letters for each job. They also have to deal with overlapping dates and all kinds of memory issues, like remembering the posts and companies that seem appealing to boost their career, the answers received, etc. Whereas it is possible to set up a rudimentary system with a spreadsheet or a text processor, an specialized program will meet the requirements in a much better way.

![Business Use Cases](image)

**Figure 1: Business Use Cases**

JobTracker can store resumes in different formats and versions, applications and opportunities. Users add opportunities to apply to them. An opportunity is a post in a company, with a salary, dates and comments. An application is a resume, a cover letter, a date and other relevant details, for a given opportunity.

JobTracker has two roles for the user: regular and premium. A regular user can access most of the contents. A premium user can access the premium area, where special services can be offered. These might include, but are out of the scope of this version of the project, connection with LinkedIn, Montser or custom services to offer jobs, like the university career service.
The uses cases are shown in 1.

Design

Architecture

Struts2 architecture is MVC. The MVC design pattern identifies three distinct concerns: model, view and controller. In Struts 2 these concerns are implemented by the action, result and Filter-Dispatcher, respectively [?].

This project is built to fit in this architecture and uses some software patterns that are proven to work well in this situation.

View

The presentation tier is written in JSP files. These components use Expression Language to show data dynamically and Bootstrap by Twitter to make a simple layout easily. The strong part of this project is data communication and persistence, not the view.

In 2 there is a list of the JSP files and the structure of the web folder.

Listing 1: Libraries used in JSP files

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>&lt;%@page contentType=&quot;text/html&quot; pageEncoding=&quot;UTF-8&quot;%&gt;</code></td>
</tr>
<tr>
<td>2</td>
<td><code>&lt;%@taglib prefix=&quot;s&quot; uri=&quot;/struts-tags&quot; %&gt;</code></td>
</tr>
<tr>
<td>3</td>
<td><code>&lt;%@taglib prefix=&quot;sb&quot; uri=&quot;/struts-bootstrap-tags&quot; %&gt;</code></td>
</tr>
<tr>
<td>4</td>
<td><code>&lt;%@taglib uri=&quot;http://java.sun.com/jsp/jstl/core&quot; prefix=&quot;c&quot; %&gt;</code></td>
</tr>
<tr>
<td>5</td>
<td><code>&lt;%@taglib prefix=&quot;fmt&quot; uri=&quot;http://java.sun.com/jsp/jstl/fmt&quot; %&gt;</code></td>
</tr>
<tr>
<td>6</td>
<td><code>&lt;%@taglib prefix=&quot;eg&quot; tagdir=&quot;/WEB-INF/tags&quot; %&gt;</code></td>
</tr>
</tbody>
</table>

Some of these views require to manipulate the data from the model to be displayed correctly. Thus, the use of iterators, conditionals and other elements of the Struts2 Tag library are used, as in 1.

Listing 2: Iterators and conditionals

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>&lt;s:if test=&quot;{%applications.isEmpty()}&quot;%&gt;</code></td>
</tr>
<tr>
<td>2</td>
<td>You haven’t added any job applications yet.</td>
</tr>
<tr>
<td>3</td>
<td><code>&lt;/s:if&gt;</code></td>
</tr>
</tbody>
</table>
A simplified demonstration of these iterators is in 2.

In the Applications (/applications) view there is a progress bar that is displayed in a different color depending on how far the application has gone. A tag was created to encapsulate this, in WEB-INF/tags/. It can be called as in 4.

Listing 3: Custom tag

```
<s:set name="statusCode" value="${#ap.statusCode}" />
<eg:jobapplicationstatusbar statusCode="${statusCode}" statusDescr="${ap.status}" />
```

Layout is done using Twitter Bootstrap, that provides widgets and default styles for most common cases, like column-grid, buttons and forms. It is included in the application using the Bootstrap plugin for Struts2.

The datepicker widget belongs to the jQuery library, which is also a plugin for Struts2.

All views are listed in 2.

**Controller**

A software engineer has to decide what patterns will be used in the domain layer. Normally, the problem would be:
In both cases the engineer has to decide how will the controller be. Many options are available, like a facade controller, an application controller, a transaction controller or others.

Most business applications can be thought of as a series of transactions. A transaction may view some information as organized in a particular way, another will make changes to it. Each interaction between a client system and a server system contains a certain amount of logic. In some cases this can be as simple as displaying information in the database. In others it may involve many steps of validations and calculations.

A Transaction Script organizes all this logic primarily as a single procedure, making calls directly to the database or through a thin database wrapper. Each transaction will have its own Transaction Script, although common subtasks can be broken into subprocedures.

A transaction controller normally uses the attributes to store input parameters. A special attribute is included to store the result. All of them have their corresponding set of getters and setters.

Struts2 is MVC. The Controller part is implemented normally with Actions. These actions fit very well the Transaction Script pattern (3). Struts2 actions look similar: they have an execute() method, parameters can be stored in the attributes and even one of the attributes is the result if the Action implements the ModelDriven interface.
Typical implementations of an Action contain light-weight logic. See the example of an insert (??) or a retrieve operation (??).

**Model**

This decision affects the patterns that can be applied to the Model. Because Transaction Script is used, the best options for the Model are:

- Table Data Gateway
- Row Data Gateway
- Active Record

Row data gateway fits best the needs of this application. For each table of the database there are two classes in the Model: a Finder and a Gateway (4)

A MyClassFinder to retrieve instances of MyClass. Typically operations are getAll(), getOne(id) and exists(id), but custom queries are also allowed, like getByName(), getByAge(), getByCompanyDistrict(), etc. MyClass abstracts a row of a table. All attributes match a column of the table. It typically has methods insert, delete, update.

These methods use JDBC to access the database. A wrapper for the Connection object is provided in DBConnection.java. A developer can configure there the connection to the database.
Figure 4: Model

Database logic design

The database schema is shown in 5.

Struts2 specific issues

This section discusses how Struts2 affects the implementation of certain components.

Login

Login works with a LoginInterceptor, a LoginAction and a login.jsp form. There are two users in the database:
• eduard: password eduard. Regular user.

• ozbek: password ozbek. Premium user.

**ModelDriven classes**

Some Actions extend ActionSupport and implement ModelDriven to ease passing parameters. It is specially useful for CRUD operations, despite the fact that update entities is out of the scope of this project. Some other classes don’t do that. The purpose was to try both methods.

**Download files**

Users can download their resumes in the same format that they were uploaded. To avoid writing one action per format, a parameter was declared in struts.xml that carries the contentType of the file.

### Listing 4: Custom tag

```xml
<action name="download" class="edu.neu.coe.csy6220.egamonal.jobTracker.action.DownloadAction">
    <result name="success" type="stream">
        <param name="contentType">contentType</param>
        <param name="inputName">fileInputStream</param>
    </result>
</action>
```
References
