

Evaluation of Future
JEE Technologies
Project Report

Nicolas Lanquetin
0604918@abertay.ac.uk



UNIVERSITY
of
ABERTAY DUNDEE

University of Abertay Dundee
School of Computing & Creative Technologies

May 2007

University of Abertay Dundee

Permission to copy

Author: Nicolas Lanquetin

Title: Evaluation of Future JEE Technologies - Project Report

Degree: BSc (Hons) Web Design & Development

Year: 4

- i) I certify that the above mentioned project is my original work.
- ii) I agree that this dissertation may be reproduced, stored or transmitted, in any form and by any means without the written consent of the undersigned.

Signature:

Date:

Contents

1. Introduction	1
2. Project Prerequisites	2
2.1. Eclipse	2
2.2. Jboss	2
2.2.1. Securing JBoss	3
2.2.2. Apache and JBoss on the Same Server	3
2.3. Environment Variables	4
2.4. Maven 2	5
2.5. Struts 2.0	5
3. Demonstration Project	6
3.1. The Project Structure	6
3.1.1. Java classes	6
3.1.2. File Descriptors and Resource Bundles	7
3.1.3. The Front-End Files	8
3.2. The Welcome Action	8
3.3. The ValidationDemo Action	9
3.4. The AnnotationValidationDemo Action	11
3.5. Summary	12
4. Load-Testing	13
4.1. Setting Up the Test Environment	13
4.2. The Tested Web Application	15
4.3. The Load-Tester	15
4.4. Building a Test Plan	15
4.5. The Load-Tests	17
5. Conclusion	18

6. Resources	19
A. Project Files	20
A.1. Java Files	20
A.2. JSP Files	23
A.3. Validation Files	27
B. The JMeter Test Plan	29
C. Honours Project Class Documentation	33
C.1. Honours Project Class Hierarchy	33
C.2. Honours Project Class List	33
C.3. com.psbaseshp.action.AnnotationValidationDemo Class Reference	34
C.3.1. Detailed Description	34
C.4. com.psbaseshp.Constants Class Reference	35
C.4.1. Detailed Description	35
C.4.2. Member Data Documentation	35
C.5. com.psbaseshp.model.ModelObject Interface Reference	37
C.5.1. Detailed Description	37
C.6. com.psbaseshp.exception.ResultNotSetException Class Reference	38
C.6.1. Detailed Description	38
C.7. com.psbaseshp.action.SupportBase Class Reference	39
C.7.1. Detailed Description	39
C.8. com.psbaseshp.model.TestObject Class Reference	40
C.8.1. Detailed Description	40
C.9. com.psbaseshp.action.ValidationDemo Class Reference	41
C.9.1. Detailed Description	41
C.10.com.psbaseshp.action.Welcome Class Reference	42
C.10.1. Detailed Description	42
Index of Honours Project Classes	43
Glossary	44
Bibliography	46

List of Figures

2.1. Apache and JBoss, Solution 1	3
2.2. Apache and JBoss, Solution 2	4
3.1. UML class diagram of the action package	7
3.2. Welcome Action	8
3.3. Welcome Action	9
3.4. ValidationDemo Action	10
3.5. ValidationDemo Action	11
3.6. AnnotationValidationDemo Action	11
4.1. Summary report	13
4.2. CPU History of javaw.exe process owned by JBoss.	15
4.3. CPU History of javaw.exe process owned by JMeter.	15
4.4. Screenshot of the JMeter application	16
B.1. Screenshot of the Welcome screen	29
B.2. Screenshot of Login screen	29
B.3. Screenshot of the main menu	29
B.4. Screenshot of the registration screen, showing the subscriptions	30
B.5. Screenshot of the screen to add a new subscription	30
B.6. Screenshot of registration screen with a new subscription	31
B.7. Screenshot of the screen to delete a subscription	31
B.8. Screenshot of a submitted form where validation failed.	32

List of Tables

4.1. Configuration of test systems	14
4.2. Configuration of JBoss	14

Listings

2.1. Configuration of a proxy and reverse proxy in Apache's httpd.conf	4
2.2. Output environment variables on console (Windows)	5
2.3. Output environment variables on console (GNU/Linux)	5
3.1. Extract of hp.xml	7
3.2. Extract of web.xml	8
3.3. Extract of ValidationDemoShort.jsp	9
3.4. Extract of AnnotationValidationDemo.java	12
A.1. Welcome.java	20
A.2. ValidationDemo.java	20
A.3. TestObject.java	21
A.4. AnnotationValidationDemo.java	22
A.5. Welcome.jsp	23
A.6. ValidationDemo.jsp	24
A.7. ValidationDemoSuccess.jsp	25
A.8. AnnotationValidationDemo.jsp	25
A.9. AnnotationValidationDemoSuccess.jsp	26
A.10. ValidationDemo-validation.xml	27

Chapter 1.

Introduction

In theory, Maven 2 and Struts 2.0 are both great tools to develop enterprise web applications. The practical part of the research is to use these tools to develop a small application and evaluate some of the features Struts 2.0 is offering. This report will therefore give a more practical view of the framework and support the key arguments of the Honours dissertation.

Additionally, the research included performance as key criteria for a good web application framework. Therefore, an open-source web application using Struts 2.0 will undertake some load-tests. The outcome of the tests will be evaluated in the Honours dissertation.

Chapter 2.

Project Prerequisites

To evaluate the frameworks, some programs and utilities are required. Since the process of setting up the environment is a large part of this project, this chapter lists which of the technologies are required for the project, gives some useful information about each of the used tool and some important advice setting them up.

Since JEE projects are built on Java, a *Java Development Kit* (JDK) and *Java Runtime Environment* (JRE) are required. In this project the JDK version 1.5.0 Update 10 is used, since it performs faster than its predecessor, and introduces the concept of annotations which are used in this project for validation.

2.1. Eclipse

Eclipse (Eclipse, 2007) is one of the most used development environment. It is an open source IDE written in Java, which goal is to incorporate the most used development tools needed in the lifecycle of software development. Its success results of the platform independency and its powerful API, allowing the integration of plug-ins. Eclipse can be extended by anyone for any needs. Two plug-ins in particular were required in this project:

1. *Subclipse*, a plug-in to enable the functionality of Subversion in the IDE (Tigris, 2007), and
2. *m2eclipse*, which helps in the building and deployment of the project.

2.2. Jboss

JBoss is a very powerful and popular application server used to deploy J2EE and JEE web applications. It comes with the not less popular Tomcat, a Servlet container which does the

actual processing of the web application.

The latest JBoss application server can be found on the JBoss website (JBoss, 2007a). It is recommended to install a version labeled 'Production', as they are stable and secure. JBoss 4.0.5 is the most recent stable version at the time of writing, and was chosen to run the Struts 2.0 application.

The Jboss application server offers three pre-defined servers that can be deployed: `all`, `default` and `minimal`. Depending on the needs, one of these three can be chosen. For the project, the default server is a good choice, as it offers a minimum of basic management and analysis tools to administrate the server over the web.

2.2.1. Securing JBoss

The tools mentioned above, in particular the web- and jmx-console, must be secured. JBoss (2007b) provides information on how to proceed and offers additional security advices. One important security measure is to run the JBoss server with as least privileges as possible. On Linux, a system user named 'jboss' can be created for this purpose. This ensures that, in case of an intrusion, only data can be lost, which the jboss user had access to.

2.2.2. Apache and JBoss on the Same Server

Most servers run *Apache* in addition to JBoss on the same machine. If both servers share the same external IP address, they cannot listen on the same port. In this case one of the server needs to run on the HTTP port (80) and the other server on another port (per default: 8080). Two solutions are possible:

1. The first solution is to make Apache listen on port 80 and JBoss on port 8080. Apache can then be used as external web server and proxy requests to JBoss using the `mod_proxy` module for Apache (Laurie and Murcko, 1996). (see Figure 2.1)

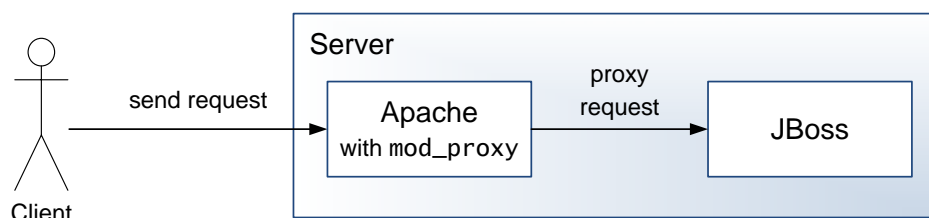


Figure 2.1: Apache and JBoss, Solution 1

2. Another more common solution is to make Apache listen on port 8080 and JBoss

on port 80. This solution can be implemented, when Apache should be used to exclusively deliver static content. To acquire data from Apache, the `mod_jk2` module of JBoss' Tomcat container must be installed (UCBerkeley, 2006). (see Figure 2.2)

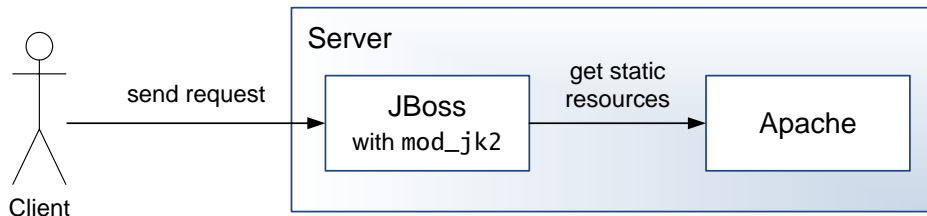


Figure 2.2: Apache and JBoss, Solution 2

In the honours project, the first solution was implemented, as the already installed Apache hosts far more web applications than JBoss. Listing 2.1 shows an extract of the configuration file.

Listing 2.1: Configuration of a proxy and reverse proxy in Apache's `httpd.conf`

```
1 NameVirtualHost 87.230.20.205:80
2
3 <VirtualHost 87.230.20.205:80>
4   Include ssl/sslsetup
5   ServerName jboss.psbase.com
6   ProxyPass / http://localhost:8080/
7   ProxyPassReverse / http://localhost:8080/
8   TransferLog /var/log/apache2/jboss.todofixme.com/access.log
9   ErrorLog /var/log/apache2/jboss.todofixme.com/error.log
10 </VirtualHost>
```

2.3. Environment Variables

Environment variables can be set in the operating system and made available to any program requesting them. In particular, two variables are important for the project to run:

1. `JAVA_HOME`: This variable must be set to the installation path of the used JDK. Amongst others, the variable is required to inform JBoss which JDK to use. It is possible to install more than one JDK, for instance in order to run JBoss with another Java version, such as Java 1.4.x or Java 1.6.x. Note that the environment variable can also be overridden in JBoss configuration files, so that the change does not affect other programs.
2. `JBoss_HOME`: This variable tells the location of the used JBoss server. However, JBoss will set this variable automatically if it is not specified.

To test if the variables are correctly set, the latter can be output on console (see Listing 2.2 and 2.3).

Listing 2.2: Output environment variables on console (Windows)

```
1 C:\>echo %JAVA_HOME%
2 E:\applications\Java\jdk1.5.0_10\
```

Listing 2.3: Output environment variables on console (GNU/Linux)

```
1 $ echo $JAVA_HOME
2 /usr/lib/java
```

2.4. Maven 2

Maven 2 is a project management framework (Massol et al., 2006, p. 22) which facilitates the lifecycle of an application development, viz. the compilation, distribution and documentation. Maven 2 could soon become a replacement for *Ant*¹, as it is based on best practices acquired from the long experience and previous mistakes of Ant projects.

However, Maven 2 needs to run with a basic configuration to package the web application. This configuration concerns some project meta-information and the listing of all required dependencies.

2.5. Struts 2.0

The first final version of Struts 2.0 has been released in late 2006. Using Struts 2.0 is therefore developing on the cutting edge. At the moment of writing, Struts 2.0.6 is the latest version (Struts, 2007c).

Struts 2.0 relies on many different key technologies, which must be understood before one can start developing a Struts application. McCuaig (2003) therefore describes the most important technologies and where the user can delve deeper into the subjects. The greatest difficulty of Struts 2.0, is that it assumes that the user knows a great amount of different tools and framework. Therefore, Struts 2.0 require a lot of reading for developers not familiar with JEE.

¹Ant is a very powerful and extensible build tool comparable to 'Make' on Linux systems. It is used in almost all J2EE projects to build and deploy web applications

Chapter 3.




Demonstration Project

A demonstration project using Maven 2 and Struts 2.0 was produced, in order to evaluate some of the features and how long it takes to apply the theory to the practice. In order to understand the project, it is recommended to read the Honours dissertation first, as it covers the framework architecture and different concepts used in Struts 2.0.

The project consists only of three Actions demonstrating internationalisation, client-side validation and validation using Java 5 annotations. However, getting the Actions to work is very time-consuming because most errors can only be discovered during runtime. However, once the first Action implemented, it is easier and faster to develop another one.

3.1. The Project Structure

The project consists of three distinct parts:

-  `src/main/java` Contains all Java classes.
-  `src/main/resources` Contains all file descriptors and resource bundles.
-  `src/main/webapp` Contains the web resources for the front-end.

3.1.1. Java classes

For better maintenance, the Java files are split into three packages: `action`, which contains the Action classes; `exception`, which contains all the Exception classes; and `model`, which contains the JavaBeans. Figure 3.1 shows the `action` package with three Actions which all extends the `SupportBase`. The latter is an abstract class which extends the `ActionSupport` class from the Struts 2.0 framework. It is a good practice to have a common base class, such as the `SupportBase` class, to implement common functionality required by all Actions.

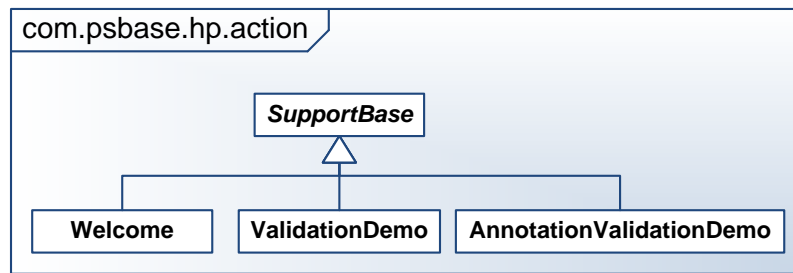


Figure 3.1: UML class diagram of the action package

3.1.2. File Descriptors and Resource Bundles

File descriptors and resources bundles are all stored together in the resources folder. The most important file descriptors for the project are `struts.xml` and `hp.xml`, as they contain the mapping of URLs to Actions. Listing 3.1 shows the configuration for the three Action classes. The file shows that every Action name is bound to a class and one or more Results. Which Result to show is decided by the Action.

Listing 3.1: Extract of `hp.xml`

```

1 <action name="Welcome" class="com.psbases.hp.action.Welcome">
2   <result>/Welcome.jsp</result>
3 </action>
4
5 <action name="ValidationDemo!*" method="{1}" class="com.psbases.hp.action.
6   ValidationDemo">
7   <result name="input">/ValidationDemo.jsp</result>
8   <result>/ValidationDemoSuccess.jsp</result>
9 </action>
10
11 <action name="AnnotationValidationDemo!*" method="{1}" class="com.psbases.hp.action.
12   AnnotationValidationDemo">
13   <result name="input">/AnnotationValidationDemo.jsp</result>
14   <result>/AnnotationValidationDemoSuccess.jsp</result>
15 </action>
  
```

Regarding the resource bundles, three types were considered: a global resource bundle (`globalMessages.properties`) which is accessible from everywhere, a package-wide resource bundle (`package.properties`) which is only accessible from the package and its sub-packages, and Action resource bundles which are only accessible from the Action with the same name (e.g. `Welcome.properties` for the `Welcome` Action). To support internationalisation, a suffix of the country code can be added to the properties files, such as `Welcome_de.properties` for Germany.

3.1.3. The Front-End Files

The webapp folder contains all files related to the web application and the Servlet container. In this project only JSPs, images and style sheets are contained in this folder. There is one file descriptor in the WEB-INF folder which is however of interest: web.xml (see Listing 3.2). This file tells the Servlet container for instance, that all requests `/*` (line 8) should be handled by the `FilterDispatcher` of Struts 2.0 (line 3).

Listing 3.2: Extract of web.xml

```
1 <filter>
2   <filter-name>struts2</filter-name>
3   <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
4 </filter>
5
6 <filter-mapping>
7   <filter-name>struts2</filter-name>
8   <url-pattern>/*</url-pattern>
9 </filter-mapping>
10
11 <welcome-file-list>
12   <welcome-file>index.html</welcome-file>
13   <welcome-file>index.jsp</welcome-file>
14 </welcome-file-list>
```

3.2. The Welcome Action

The Welcome Action is the first Action a visitor will see. Figure 3.2 shows a screenshot of the Action.



Figure 3.2: Welcome Action

Except the text, everything is hard coded into the `Validation.jsp` (see Listing A.5, p. 23).

This class tries to use some of the Struts 2.0 tags, such as `s:url`, `s:param` and `s:text` and should demonstrate that even with a short JSP code, it is possible to generate a dynamic page, with support for internationalisation. All the text messages are hold in the resource bundle files mentioned in Section 3.1.2 and are dynamically retrieved depending on the `request_locale` parameter stored in the user's session via a cookie or URL-rewriting. When 'Englisch' or 'German' is chosen from the Language menu (see Figure 3.2), `request_locale` is set to 'en' or 'de' accordingly.

3.3. The ValidationDemo Action

The ValidationDemo Action shown in Figure 3.3 was developed to test client-side validation and the most used validation rules in Struts 2.0.

<http://jboss.psbases.com/hp/ValidationDemo!input.action>



The screenshot shows a web form titled "Validation Demonstration". It contains the following fields and controls:

- Nickname:
- Birthday (DD/MM/YY):
- Age:
- Email Address:
- Website:
- Preferred Cities:
- A "Save" button.
- A link: [Back to the main page](#)

Figure 3.3: Welcome Action

The form is realised with a short and simple JSP code shown in Listing 3.3. All the data in this form will be stored in the `TestObject` JavaBean (Listing A.3, p. 21) which is made accessible by the `ValidationDemoAction` (Listing A.2, p. 20).

Listing 3.3: Extract of `ValidationDemoShort.jsp`

```
1 <s:form action="ValidationDemo" validate="true">
2   <s:textfield label="%{getText('testObject.nickName')}" name="testObject.nickName"/>
3   <s:textfield label="%{getText('testObject.birthDay')}" name="testObject.birthDay"/>
4   <s:textfield label="%{getText('testObject.age')}" name="testObject.age"/>
5   <s:textfield label="%{getText('testObject.emailAddress')}" name="testObject.
   emailAddress"/>
6   <s:textfield label="%{getText('testObject.webSite')}" name="testObject.webSite"/>
```



```
7 <s:textfield label="%{getText('testObject.preferedCities')}" name="testObject.preferedCities"/>
8 <s:submit value="%{getText('save')}" />
9 </s:form>
```

If some test data is entered into the form and the 'Save' button is clicked, the form is validated on the client-side. A possible result is shown in Figure 3.4. Of course the validation fails because the entered data is not valid.

http://jboss.psbases.com/hp/ValidationDemo.action

Validation Demonstration

Nickname is required

Nickname:

Birthday (DD/MM/YY):

Your Age must be between 18 and 65

Age:

The email address you entered is not valid

Email Address:

Invalid website url

Website:

Please write between 5 and 50 characters for your prefered cities

Preferred Cities:

[Back to the main page](#)

Figure 3.4: ValidationDemo Action

A validation file, `ValidationDemo-validation.xml` (Listing A.10, p. 27) must be used together with the Action in order to validate the form. This file holds an entry for every field. Inside each `field` tag, it is possible to configure as many field-validator as required. The messages for invalid fields are obtained by the resource bundles and can therefore be set in any language.

Only when all validation rules are respected, the form is sent to the server. The `ValidationInterceptor` checks the submitted data again for security reason. Next, the

ConversionInterceptor verifies if all received fields can be transformed to the data types defined in TestObject. If the latter fails, the form is sent back with the appropriate conversion error messages. However none of this must be configured, since these Interceptors are already included in the default interceptor stack, which every Action uses by default.

Finally, if no errors were found, the ValidationDemoSuccess.jsp (Listing A.7, p. 25) is used to display the result. Figure 3.5 shows the result in the browser.

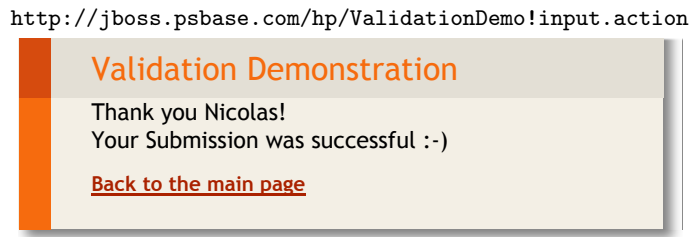


Figure 3.5: ValidationDemo Action

3.4. The AnnotationValidationDemo Action

The problem with the previous Action was that three files had to be managed to make the validation work: the Action, the resource bundles and the validation file. However, with annotations it is possible to handle the validation inside the Action. An Action called AnnotationValidationDemo has been developed for this purpose. Figure 3.6 displays a browser screenshot of the Action.

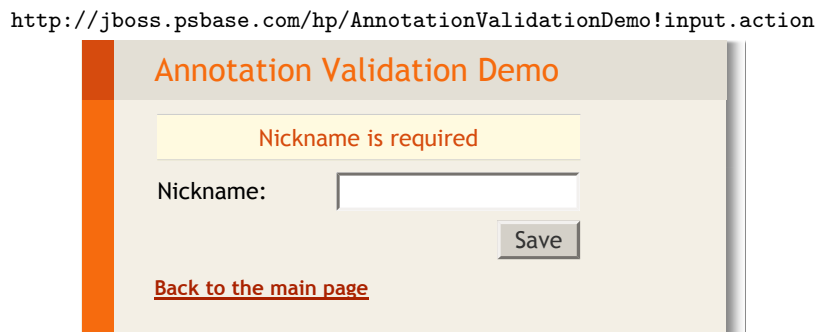


Figure 3.6: AnnotationValidationDemo Action

The Action contains only one field, which is required. By adding a `@RequiredFieldValidator` annotation above the setter method of that field, Struts 2.0 validates the field (see Listing 3.4). As shown on Figure 3.6, validation works.

Listing 3.4: Extract of AnnotationValidationDemo.java

```
1 @RequiredFieldValidator(  
2     type = ValidatorType.FIELD,  
3     message = "Nickname is required")  
4 public void setNickName(String nickName) {  
5     this.nickName = nickName;  
6 }
```

3.5. Summary

This chapter covered some practical example of what can be done with Struts 2.0. Even though the examples might look simple, it takes a great amount of effort and patience to implement and debug the application. However, with more practice and experience, developer should be able to rapidly implement a working enterprise web application.

Chapter 4.

Load-Testing

4.1. Setting Up the Test Environment

To analyse the performance of Struts 2.0 applications, a distributed load-test environment was set up. Figure 4.1 illustrates the used network topology. One system runs the JBoss application server, with a test application, and an additional system runs the JMeter software to load-test the application. Both Systems are on a dedicated network, connected together with a 100 Mbit switch.

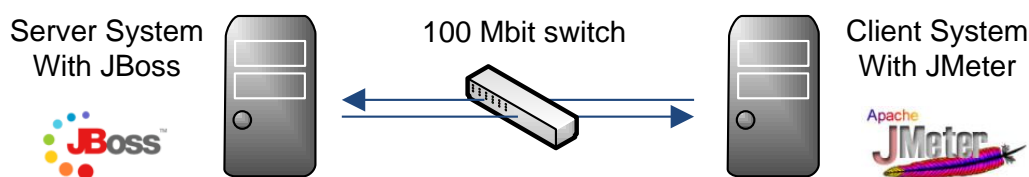


Figure 4.1: Summary report

Table 4.1 lists the most important settings for the used systems. Both run the Windows XP operating system with SP2. Windows XP is the minimum configuration for load-testing, as older Windows systems do not allow more than 50-60 simultaneous connections. Also, both servers have enough memory installed, which means that the server's limits will probably be reached because of the CPU speed and/or the hard drive access time.

Another important factor is the configuration regarding JBoss. Table 4.2 shows the most important settings. JBoss comes with three different pre-built servers, which already contains some basic web applications to monitor and manage the application server. Mentioning which server was chosen is important, because JBoss will use a certain amount of memory to deploy the server's applications. In the test case, the 'default' server was chosen.

Additionally, the test client accesses the server directly with the server's IP address, to avoid the overhead of a DNS lookup. Furthermore the server is accessed directly on its

	Server System	Test Client System
OS	Windows XP Pro SP2	Windows XP Pro SP2
JDK/JRE	jdk1.5.0_10	jre1.5.0_10
NIC	100 MBit	100 MBit
CPU	Athlon XP 3200+, 2.01 GHz	Intel Pentium M, 1.60 Ghz
Memory total	2048 MB	1280 MB
Memory free	> 1024 MB	> 320 MB
Hard drives	Mirror Raid: 2 x 300GB WD Caviar with 7200 rpm and 16MB buffer on NVidia nForce SATA2 Raid Controller	n/a

Table 4.1: Configuration of test systems

	JBoss
Min memory for JBoss	128 MB
Max memory for JBoss	512 MB
Used server	default + struts2-mailreader-2.0.6.war

Table 4.2: Configuration of JBoss

port 8080, instead of being tunnelled through the Apache Proxy. This is a precautionary measure to ensure that the Apache server won't be a bottle-neck. Finally, an initial test will be executed once prior to data gathering, because the application might need to compile the JSPs to Servlets, which is a relatively slow one-time process.

Both systems were configured to run a minimum of processes to guaranty that the CPU is fully dedicated to JBoss on the server and JMeter on the client system accordingly. To further guaranty that JBoss and JMeter get most of the CPU time, both run with a high priority (windows priority 13). During a first test, JBoss and JMeter were monitored regarding their CPU usage. Figure 4.2 and 4.3 shows the result of that monitoring. JBoss uses up to 91.89% of the CPU power available, which is an acceptable value. Figure 4.3, in turn, shows that JMeter only uses a total of 7.81% of the available CPU, which ensures that JMeter will not be the cause for suffered performance losses.

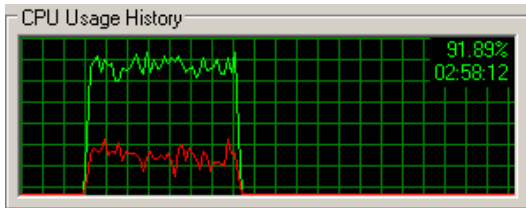


Figure 4.2: CPU History of javaw.exe process owned by JBoss.

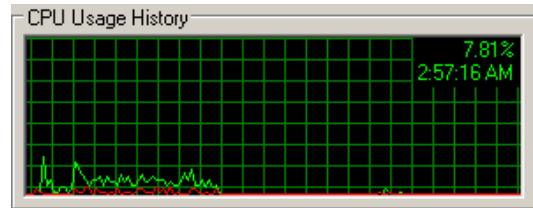


Figure 4.3: CPU History of javaw.exe process owned by JMeter.

4.2. The Tested Web Application

The application, which will be load-tested, is the open source Struts MailReader 2. The latter allows to create, login and logout users; edit a user profile; and add, edit and remove subscriptions to IMAP or POP3 hosts. The application scope (usually a database) is implemented as XML file and permanently stores all the user data for the time JBoss is running. It is possible that the single XML file will be a bottle neck, as many request will need to access this file simultaneously.

4.3. The Load-Tester

For the load-testing, Apache JMeter (Apache, 2007) will be used. It is a free Java desktop application designed to load test functional behaviour and measure performance. The program was originally designed for testing web application. It is possible to execute HTTP requests on a web server and supply the requests with parameters.

4.4. Building a Test Plan

To create a so-called test plan, JMeter provides an easy to use GUI application as shown in Figure 4.4. The test plan consists of HTTP requests displayed in the left frame of the GUI. By adding enough HTTP requests it is possible to build a scenario, where the user logs in, add and removes a subscription, and logs out. Appendix B shows the screenshots of that scenario.

The JMeter application provide a proxy feature which allows the 'recording' of a scenario. To take advantage of that feature, JMeter must be configured as proxy, which will allow the saving of all requests sent to the web application. This is a huge advantage, because JMeter also saves all the parameters used in a request.

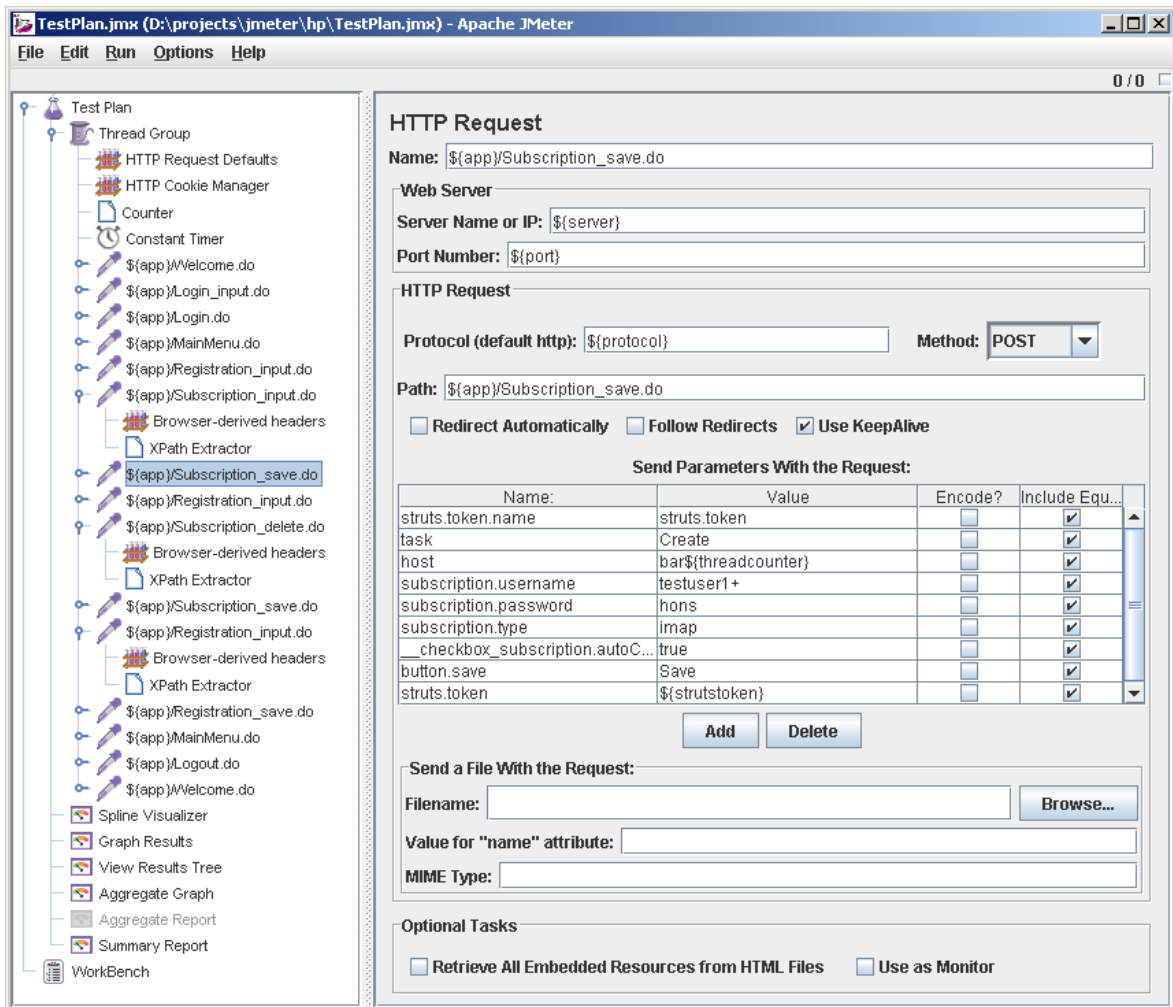


Figure 4.4: Screenshot of the JMeter application. Current view: Configuration of Subscription_save.do

The configuration of a test plan is however time-consuming. A *Test Plan* usually consists of a *Thread Group*, where it is possible to define how many times the *samples* inside the Thread Group should be executed. The Thread Group also hold the settings about the number of simultaneous users and in which time period (ramp-up period) the users should start the thread. As shown on Figure 4.4, it is possible to configure other elements, such as the *HTTP Cookie Manager*, which enables the request to accept cookies. Additionally, a *Counter* was added to allow simultaneous threads to use unique data. The right frame of Figure 4.4 shows the request parameters which will be used for the currently highlighted Subscription_save.do Action. The 'host' parameter has the value of 'bar\${threadcounter}', which means that the value of this parameter will differ for every user, thereby preventing two users to send the same host parameter. Another element

in the Thread Group is the *Constant Timer*, which can be configured to allow a certain time to elapse between each sample. Finally Struts 2.0 uses tokens to prevent double submits. If the sent token does not match the token written in the html form, Struts 2.0 will assume that the form has been already submitted and will not take the request into account. Because these tokens change for every new request, it is necessary to extract the token from the HTML form. This can be done with the *XPath Extractor* which looks for `'/html/body/form/input[@name='struts.token']/@value'` in the XHTML response and makes the value available as `'${strutstoken}'` variable.

Once the test plan works without any errors, it is possible to add *Listeners*. Many different exist, but the most important are the *Graph Results*, which show a graphical representation of the gathered data, and the *Summary Report*, which represent overall results in a table.

4.5. The Load-Tests

Two load-tests have been realised. The first is a single-user load-test to find out about the different response times of each tested Struts 2.0 features. The second, in turn, is a multi-user load-test which should clarify how many simultaneous users can be supported and were Struts 2.0 has its limits. The results of the load-tests are analysed and interpreted in the Honours dissertation.

Chapter 5.

Conclusion

The practical part of the research covered an implementation of a web application using Struts 2.0, and a load-test of an open-source web application.




In theory, Struts 2.0 is an excellent framework, which separates concerns and offers a lot of great concepts and functionality. However, the implementation of a small web application has shown that in practice a lot of time can be consumed in debugging the application and searching for errors, as they occur only during runtime. This is mainly because files are maintained in different location as explained in Section 3.1, The Project Structure. An Action consists of several different parts. If one of these parts is changed, it required some adaptation in the other parts. In the currently available IDEs, refactoring over different types of files is not perfect yet, which means that all parts of an Action must be adapted by hand. However, as mentioned in Section 3.4, The AnnotationValidationDemo Action, it is possible to use annotations to decrease the number of files to manage.

In addition, the load-tests were a necessary part of this research, as they would show if Struts 2.0 is capable of handling many simultaneous connections. JMeter is a great software to realise the tests: It is free and offers a lot of functionality to handle cookies, URL-rewriting, counters and useful ways to extract text out of the response. Results are however analysed and evaluated in the dissertation, and therefore not covered in this report.





Chapter 6.

Resources

Content on the Web

-  <http://jboss.psbases.com/hp/>
Hosts the demonstration application.
-  <http://www.psbases.com/hp/>
Hosts the Java documentation of the project.
-  https://guest:guest@svn.psbases.com/ca1033a_project/
Host the demonstration project Subversion repository.

Content on the CD

-  **Documentation** Contains this project report, its source code, and the Java documentation of the project.
-  **Project Files** Contains the JMeter test plan and the source code of the demonstration web application.
-  **Software** Contains all the software used to implement the project.
-  **Web Archives** Contains the web applications as deployable WAR files. `hp.war` is the demonstration web application, and `struts2-mailreader-2.0.6.war` is the Struts 2.0 Mailreader application used for the load-tests.

Appendix A.

Project Files

A.1. Java Files

Listing A.1: Welcome.java

```
1 package com.psbases.hp.action;
2
3 /**
4  * This is the Welcome Page of the Demonstration Site.
5  *
6  * @author Nicolas Lanquetin (0604918)
7  * @version $Id: Welcome.java 20 2007-05-08 14:37:54Z ps $
8  */
9 public class Welcome extends SupportBase {
10 }
```

Listing A.2: ValidationDemo.java

```
1 package com.psbases.hp.action;
2
3 import com.psbases.hp.model.TestObject;
4
5 /**
6  * This Action is a Demo, used to validate all fields of the
7  * TestObject.
8  *
9  * @author Nicolas Lanquetin (0604918)
10 * @version $Id: ValidationDemo.java 20 2007-05-08 14:37:54Z ps $
11 * @see TestObject
12 */
13 public class ValidationDemo extends SupportBase {
14
15     private static final long serialVersionUID = -7548250779091041743L;
16
17     private TestObject testObject = null;
18
19     public TestObject getTestObject() {
20         return testObject;
21     }
22 }
```

```
21 }
22
23 public void setTestObject(TestObject testObject) {
24     this.testObject = testObject;
25 }
26 }
```

Listing A.3: TestObject.java

```
1 package com.psbaseshp.model;
2
3 import java.sql.Date;
4
5 /**
6  * This is JavaBean Test Object which holds different kinds
7  * of types, which can be used for validation.
8  *
9  * @author Nicolas Lanquetin (0604918)
10 * @version $Id: TestObject.java 20 2007-05-08 14:37:54Z ps $
11 */
12 public class TestObject implements ModelObject {
13
14     private static final long serialVersionUID = 3724482471979777317L;
15     private String nickName = null; // requiredStringValidatorField
16     private Date birthDay = null; // integerValidatorField
17     private Integer age = null; // dateValidatorField
18     private String emailAddress = null; // emailValidatorField
19     private String webSite = null; // urlValidatorField
20     private String preferredCities = null; // stringLengthValidatorField
21
22     public Integer getAge() {
23         return age;
24     }
25
26     public void setAge(Integer age) {
27         this.age = age;
28     }
29
30     public Date getBirthDay() {
31         return birthDay;
32     }
33
34     public void setBirthDay(Date birthDay) {
35         this.birthDay = birthDay;
36     }
37
38     public String getEmailAddress() {
39         return emailAddress;
40     }
41
42     public void setEmailAddress(String emailAddress) {
43         this.emailAddress = emailAddress;
44     }
45 }
```

```
46 public String getNickName() {
47     return nickName;
48 }
49
50 public void setNickName(String nickName) {
51     this.nickName = nickName;
52 }
53
54 public String getPreferredCities() {
55     return preferredCities;
56 }
57
58 public void setPreferredCities(String preferredCities) {
59     this.preferredCities = preferredCities;
60 }
61
62 public String getWebSite() {
63     return webSite;
64 }
65
66 public void setWebSite(String webSite) {
67     this.webSite = webSite;
68 }
69 }
```

Listing A.4: AnnotationValidationDemo.java

```
1 package com.psbaseshp.action;
2
3 import com.opensymphony.xwork2.ActionSupport;
4 import com.opensymphony.xwork2.validator.annotations.RequiredFieldValidator;
5 import com.opensymphony.xwork2.validator.annotations.Validation;
6 import com.opensymphony.xwork2.validator.annotations.ValidatorType;
7
8 /**
9  * This class is to demonstrate validation using Java 1.5 Annotations
10  *
11  * @author Nicolas Lanquetin (0604918)
12  * @version $Id: AnnotationValidationDemo.java 23 2007-05-09 05:58:30Z ps $
13  */
14 @Validation
15 public class AnnotationValidationDemo extends ActionSupport {
16
17     private static final long serialVersionUID = -5738948001604664556L;
18     private String nickName = null;
19
20     public String getNickName() {
21         return nickName;
22     }
23
24     @RequiredFieldValidator(
25         type = ValidatorType.FIELD,
26         message = "Nickname is required")
27     public void setNickName(String nickName) {
```

```
28     this.nickname = nickname;
29   }
30 }
```

A.2. JSP Files

Listing A.5: Welcome.jsp

```
1 <!--
2   @author Nicolas Lanquetin
3   @version $Id: Welcome.jsp 23 2007-05-09 05:58:30Z ps $
4 --%>
5
6 <%@ page contentType="text/html; charset=UTF-8" %>
7 <%@ taglib prefix="s" uri="/struts-tags" %>
8
9 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
10    xhtml1/DTD/xhtml1-transitional.dtd">
11 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
12   <head>
13     <title><s:text name="sitetitle"/> - <s:text name="pagetitle"/></title>
14     <link href="css/styles.css" rel="stylesheet" type="text/css"/>
15   </head>
16   <body>
17     <h1><s:text name="sitetitle"/></h1>
18     <div id="container">
19       <div id="table-wrapper">
20
21         <h2><s:text name="welcome.language.choose"/></h2>
22         <div id="content">
23           <ul>
24             <li>
25               <s:url id="url" action="Welcome">
26                 <s:param name="request_locale">en</s:param>
27               </s:url>
28               <s:a href="{url}"><s:text name="welcome.language.en"/></s:a>
29             </li>
30             <li>
31               <s:url id="url" action="Welcome">
32                 <s:param name="request_locale">de</s:param>
33               </s:url>
34               <s:a href="{url}"><s:text name="welcome.language.de"/></s:a>
35             </li>
36           </ul>
37         </div>
38
39         <h2><s:text name="welcome.validation.test"/></h2>
40         <div id="content">
41           <ul>
```

Appendix A. Project Files

```
42     <li><s:a href="ValidationDemo!input.action"><s:text name="welcome.
43         validation.client"/></s:a></li>
44     <li><s:a href="AnnotationValidationDemo!input.action"><s:text name="
45         welcome.validation.annotation"/></s:a></li>
46 </ul>
47 </div>
48 </div>
49 </body>
50 </html>
```

Listing A.6: ValidationDemo.jsp

```
1 <%--
2   @author Nicolas Lanquetin
3   @version $Id: ValidationDemo.jsp 20 2007-05-08 14:37:54Z ps $
4 --%>
5
6 <%@ page contentType="text/html; charset=UTF-8" %>
7 <%@ taglib prefix="s" uri="/struts-tags" %>
8
9 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
10    xhtml1/DTD/xhtml1-transitional.dtd">
11 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
12   <head>
13     <title><s:text name="sitetitle"/> - <s:text name="pagetitle"/></title>
14     <link href="css/styles.css" rel="stylesheet" type="text/css"/>
15   </head>
16   <body>
17     <h1><s:text name="sitetitle"/></h1>
18     <div id="container">
19       <div id="table-wrapper">
20         <h2><s:text name="pagetitle"/></h2>
21
22         <s:form action="ValidationDemo" validate="true">
23           <s:textfield label="%{getText('testObject.nickName')}" name="testObject.
24             nickName"/>
25           <s:textfield label="%{getText('testObject.birthDay')}" name="testObject.
26             birthDay"/>
27           <s:textfield label="%{getText('testObject.age')}" name="testObject.age"/>
28           <s:textfield label="%{getText('testObject.emailAddress')}" name="testObject.
29             emailAddress"/>
30           <s:textfield label="%{getText('testObject.webSite')}" name="testObject.
31             webSite"/>
32           <s:textfield label="%{getText('testObject.preferedCities')}" name="
33             testObject.preferedCities"/>
34           <s:submit value="%{getText('save')}" />
35         </s:form>
36
37         <div id="content">
38           <p><a href="<s:url action="Welcome"/>"><s:text name="back2main"/></a></p>
39         </div>
```

Appendix A. Project Files

```
35     </div>
36   </div>
37 </body>
38 </html>
```

Listing A.7: ValidationDemoSuccess.jsp

```
1 <%--
2   @author Nicolas Lanquetin
3   @version $Id: ValidationDemoSuccess.jsp 17 2007-05-08 10:57:55Z ps $
4 --%>
5
6 <%@ page contentType="text/html; charset=UTF-8" %>
7 <%@ taglib prefix="s" uri="/struts-tags" %>
8
9 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
10    xhtml1/DTD/xhtml1-transitional.dtd">
11 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
12   <head>
13     <title><s:text name="sitetitle"/> - <s:text name="pagetitle"/></title>
14     <link href="css/styles.css" rel="stylesheet" type="text/css"/>
15   </head>
16   <body>
17     <h1><s:text name="sitetitle"/></h1>
18     <div id="container">
19       <div id="table-wrapper">
20         <h2><s:text name="pagetitle"/></h2>
21
22         <div id="content">
23           <p><s:text name="thankyou"/> <s:property value="testObject.nickName"/>!<br/>
24           <s:text name="validation.successful"/></p>
25           <p><a href="<s:url action="Welcome"/>"><s:text name="back2main"/></a></p>
26         </div>
27       </div>
28     </div>
29   </body>
30 </html>
```

Listing A.8: AnnotationValidationDemo.jsp

```
1 <%--
2   @author Nicolas Lanquetin
3   @version $Id: AnnotationValidationDemo.jsp 23 2007-05-09 05:58:30Z ps $
4 --%>
5
6 <%@ page contentType="text/html; charset=UTF-8" %>
7 <%@ taglib prefix="s" uri="/struts-tags" %>
8
9 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
10    xhtml1/DTD/xhtml1-transitional.dtd">
11 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
12   <head>
```



```

12 <title><s:text name="sitetitle"/> - <s:text name="pagetitle"/></title>
13 <link href="css/styles.css" rel="stylesheet" type="text/css"/>
14 </head>
15
16 <body>
17 <h1><s:text name="sitetitle"/></h1>
18 <div id="container">
19 <div id="table-wrapper">
20 <h2><s:text name="pagetitle"/></h2>
21
22 <s:form action="AnnotationValidationDemo" validate="true">
23 <s:textfield label="%{getText('nickName')}}" name="nickName"/>
24 <s:submit value="%{getText('save')}}" />
25 </s:form>
26
27 <div id="content">
28 <p><a href="<s:url action="Welcome"/>"><s:text name="back2main"/></a></p>
29 </div>
30 </div>
31 </div>
32 </body>
33 </html>

```

Listing A.9: AnnotationValidationDemoSuccess.jsp

```

1 <!--
2 @author Nicolas Lanquetin
3 @version $Id: AnnotationValidationDemoSuccess.jsp 21 2007-05-08 14:41:31Z ps $
4 --%>
5
6 <%@ page contentType="text/html; charset=UTF-8" %>
7 <%@ taglib prefix="s" uri="/struts-tags" %>
8
9 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
10 xhtml1/DTD/xhtml1-transitional.dtd">
11 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
12 <head>
13 <title><s:text name="sitetitle"/> - <s:text name="pagetitle"/></title>
14 <link href="css/styles.css" rel="stylesheet" type="text/css"/>
15 </head>
16 <body>
17 <h1><s:text name="sitetitle"/></h1>
18 <div id="container">
19 <div id="table-wrapper">
20 <h2><s:text name="pagetitle"/></h2>
21
22 <div id="content">
23 <p><s:text name="thankyou"/> <s:property value="nickName"/>!<br/>
24 <s:text name="validation.successful"/></p>
25 <p><a href="<s:url action="Welcome"/>"><s:text name="back2main"/></a></p>
26 </div>
27 </div>
28 </div>

```

```
29 </body>
30 </html>
```

A.3. Validation Files

Listing A.10: ValidationDemo-validation.xml

```
1 <!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN" "http:
  //www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
2
3 <validators>
4   <field name="testObject.nickName">
5     <field-validator type="required">
6       <message key="requiredstring"/>
7     </field-validator>
8   </field>
9
10  <field name="testObject.birthDay">
11    <field-validator type="required">
12      <message key="requiredstring"/>
13    </field-validator>
14    <field-validator type="date">
15      <param name="min">01/01/1942</param>
16      <param name="max">01/01/1989</param>
17      <message key="testObject.birthDay.invalid.date"/>
18    </field-validator>
19    <field-validator type="conversion">
20      <param name="repopulateField">true</param>
21      <message key="testObject.birthDay.invalid.conversion"/>
22    </field-validator>
23  </field>
24
25  <field name="testObject.age">
26    <field-validator type="required">
27      <message key="requiredstring"/>
28    </field-validator>
29    <field-validator type="conversion">
30      <param name="repopulateField">true</param>
31      <message key="testObject.age.invalid.conversion"/>
32    </field-validator>
33    <field-validator type="int">
34      <param name="min">18</param>
35      <param name="max">65</param>
36      <message key="testObject.age.invalid.int"/>
37    </field-validator>
38  </field>
39
40  <field name="testObject.emailAddress">
41    <field-validator type="required">
42      <message key="requiredstring"/>
43    </field-validator>
```

Appendix A. Project Files

```
44     <field-validator type="email">
45         <message key="testObject.emailAdress.invalid.email"/>
46     </field-validator>
47 </field>
48
49 <field name="testObject.webSite">
50     <field-validator type="required">
51         <message key="requiredstring"/>
52     </field-validator>
53     <field-validator type="url">
54         <message key="testObject.webSite.invalid.url"/>
55     </field-validator>
56 </field>
57
58 <field name="testObject.preferedCities">
59     <field-validator type="required">
60         <message key="requiredstrings"/>
61     </field-validator>
62     <field-validator type="stringlength">
63         <param name="minLength">5</param>
64         <param name="maxLength">50</param>
65         <param name="trim">true</param>
66         <message key="testObject.preferedCities.invalid.stringlength"/>
67     </field-validator>
68 </field>
69 </validators>
```

Appendix B.

The JMeter Test Plan

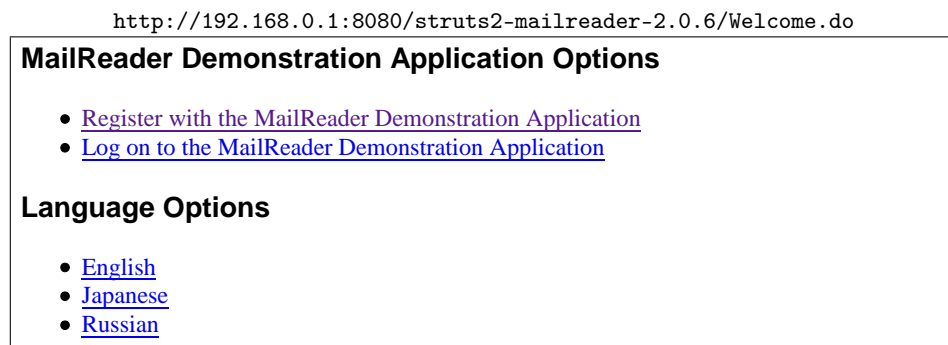


Figure B.1: Screenshot of the Welcome screen

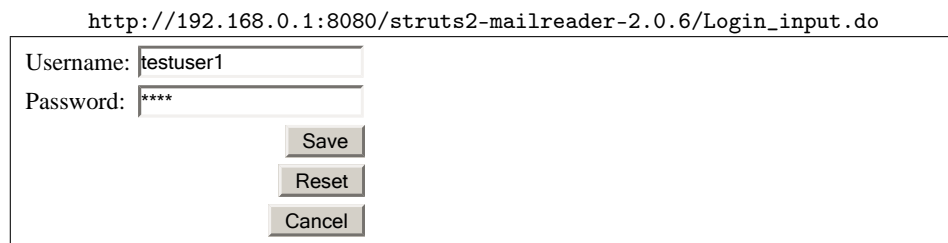


Figure B.2: Screenshot of Login screen

http://192.168.0.1:8080/struts2-mailreader-2.0.6/MainMenu.do



Figure B.3: Screenshot of the main menu

http://192.168.0.1:8080/struts2-mailreader-2.0.6/Registration_input.do

Username: testuser1
Password:
(Repeat) Password:
Full Name: testuser1
From Address: testusers@testers.com
Reply To Address: testusers@testers.com

Current Subscriptions

Host Name	User Name	Server Type	Auto	Action
-----------	-----------	-------------	------	--------

[Add](#)

Figure B.4: Screenshot of the registration screen, showing the subscriptions

http://192.168.0.1:8080/struts2-mailreader-2.0.6/Subscription_input.do

Username: testuser1
Mail Server: mail.yahoo.com
Mail Username: testuser1
Mail Password: hons
Server Type: POP3 Protocol ▾
 Auto Connect

Figure B.5: Screenshot of the screen to add a new subscription

http://192.168.0.1:8080/struts2-mailreader-2.0.6/Registration_input.do

Username: testuser1
Password:
(Repeat) Password:
Full Name: testuser1
From Address: testusers@testers.com
Reply To Address: testusers@testers.com

Current Subscriptions

Host Name	User Name	Server Type	Auto	Action
mail.yahoo.com	testuser1	pop3	false	Delete Edit

[Add](#)

Figure B.6: Screenshot of registration screen with a new subscription

http://192.168.0.1:8080/struts2-mailreader-2.0.6/Subscription_delete.do?host=mail.yahoo.com

Username: testuser1
Mail Server: mail.yahoo.com
Mail Username: testuser1
Mail Password: hons
Server Type: pop3
Auto Connect: false

Figure B.7: Screenshot of the screen to delete a subscription

http://192.168.0.1:8080/struts2-mailreader-2.0.6/Registration_save.do

Username: testuser1
Password:
(Repeat) Password:

Full Name is required
Full Name:

From Address is required
From Address:
Reply To Address:

Current Subscriptions

Host Name	User Name	Server Type	Auto	Action
-----------	-----------	-------------	------	--------

[Add](#)

Figure B.8: Screenshot of a submitted form where validation failed.

Appendix C.

Honours Project Class Documentation

C.1. Honours Project Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

com.psbases.hp.action.AnnotationValidationDemo	34
com.psbases.hp.Constants	35
com.psbases.hp.model.ModelObject	37
com.psbases.hp.model.TestObject	40
com.psbases.hp.exception.ResultNotSetException	38
com.psbases.hp.action.SupportBase	39
com.psbases.hp.action.ValidationDemo	41
com.psbases.hp.action.Welcome	42

C.2. Honours Project Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

com.psbases.hp.action.AnnotationValidationDemo	34
com.psbases.hp.Constants	35
com.psbases.hp.model.ModelObject	37
com.psbases.hp.exception.ResultNotSetException	38
com.psbases.hp.action.SupportBase	39
com.psbases.hp.model.TestObject	40
com.psbases.hp.action.ValidationDemo	41
com.psbases.hp.action.Welcome	42

C.3. com.psbases.hp.action.AnnotationValidationDemo Class Reference

C.3.1. Detailed Description

This class is to demonstrate validation using Java 1.5 Annotations

Author:

Nicolas Lanquetin (0604918)

Versions:

AnnotationValidationDemo.java 23 2007-05-09 05:58:30Z ps

The documentation for this class was generated from the following file:

- E:/projects/java/hp/src/main/java/com/psbase/hp/action/AnnotationValidationDemo.java

C.4. com.psbaseshp.Constants Class Reference

Static Public Attributes

- static final String CANCEL = "cancel"
- static final String CREATE = "Create"
- static final String EDIT = "Edit"
- static final String DELETE = "Delete"

C.4.1. Detailed Description

This Class holds all constants required in this application.

Author:

Nicolas Lanquetin (0604918)

VersIdn:

Constants.java 20 2007-05-08 14:37:54Z ps

C.4.2. Member Data Documentation

C.4.2.1. final String com.psbaseshp.Constants.CANCEL = "cancel" [static]

The token representing a "cancel" request.

C.4.2.2. final String com.psbaseshp.Constants.CREATE = "Create" [static]

The token representing a "create" task.

C.4.2.3. final String com.psbaseshp.Constants.EDIT = "Edit" [static]

The token representing a "edit" task.

C.4.2.4. final String com.psbaseshp.Constants.DELETE = "Delete" [static]

The token representing a "delete" task.

The documentation for this class was generated from the following file:

- `E:/projects/java/hp/src/main/java/com/psbase/hp/Constants.java`

C.5. com.psbases.hp.model.ModelObject Interface Reference

Inherited by com.psbases.hp.model.TestObject.

C.5.1. Detailed Description

ModelObject is an Interface for all ModelObjects.

Author:

Nicolas Lanquetin (0604918)

VersIdn:

ModelObject.java 20 2007-05-08 14:37:54Z ps

The documentation for this interface was generated from the following file:

- E:/projects/java/hp/src/main/java/com/psbase/hp/model/ModelObject.java

C.6. com.pibase.hp.exception.ResultNotSetException Class Reference

C.6.1. Detailed Description

The ResultNotSetException is thrown when no Action Result was set.

Author:

Nicolas Lanquetin (0604918)

Versions:

ResultNotSetException.java 17 2007-05-08 10:57:55Z ps

The documentation for this class was generated from the following file:

- E:/projects/java/hp/src/main/java/com/pibase/hp/exception/ResultNotSetException.java

C.7. com.psbaseshp.action.SupportBase Class Reference

Inherited by com.psbaseshp.action.ValidationDemo, and com.psbaseshp.action.Welcome.

C.7.1. Detailed Description

This is the Base Action Class for all Action Classes. It holds common functionality.

Author:

Nicolas Lanquetin (0604918)

VersIdn:

SupportBase.java 20 2007-05-08 14:37:54Z ps

The documentation for this class was generated from the following file:

- E:/projects/java/hp/src/main/java/com/psbase/hp/action/SupportBase.java

C.8. com.psbases.hp.model.TestObject Class Reference

Inherits com.psbases.hp.model.ModelObject.

C.8.1. Detailed Description

This is JavaBean Test Object which holds different kinds of types, which can be used for validation.

Author:

Nicolas Lanquetin (0604918)

VersIdn:

TestObject.java 20 2007-05-08 14:37:54Z ps

The documentation for this class was generated from the following file:

- E:/projects/java/hp/src/main/java/com/psbase/hp/model/TestObject.java

C.9. com.psbases.hp.action.ValidationDemo Class Reference

Inherits com.psbases.hp.action.SupportBase.

C.9.1. Detailed Description

This Action is a Demo, used to validate all fields of the TestObject.

Author:

Nicolas Lanquetin (0604918)

VersIdn:

ValidationDemo.java 20 2007-05-08 14:37:54Z ps

See also:

TestObject

The documentation for this class was generated from the following file:

- E:/projects/java/hp/src/main/java/com/psbase/hp/action/ValidationDemo.java

C.10. com.psbaseshp.action.Welcome Class Reference

Inherits com.psbaseshp.action.SupportBase.

C.10.1. Detailed Description

This is the Welcome Page of the Demonstration Site.

Author:

Nicolas Lanquetin (0604918)

Versions:

Welcome.java 20 2007-05-08 14:37:54Z ps

The documentation for this class was generated from the following file:

- E:/projects/java/hp/src/main/java/com/psbase/hp/action/Welcome.java

Index of Honours Project Classes

CANCEL

com::psbase:hp::Constants, 35

com::psbase:hp::action::AnnotationValidationDemo,
34

com::psbase:hp::action::SupportBase, 39

com::psbase:hp::action::ValidationDemo,
41

com::psbase:hp::action::Welcome, 42

com::psbase:hp::Constants, 35

CANCEL, 35

CREATE, 35

DELETE, 35

EDIT, 35

com::psbase:hp::exception::ResultNotSetException,
38

com::psbase:hp::model::ModelObject, 37

com::psbase:hp::model::TestObject, 40

CREATE

com::psbase:hp::Constants, 35

DELETE

com::psbase:hp::Constants, 35

EDIT

com::psbase:hp::Constants, 35

Glossary

Notation	Description
AJAX	Asynchronous JavaScript and XML: <i>AJAX is used to only refresh specific parts of a website instead of reloading the entire page.</i>
ASF	Apache Software Foundation: <i>The Apache Software Foundation is a non-profit corporation, operating many different web related projects and a wide community of members.</i>
EAR	Enterprise ARchive: <i>An EAR file packages one or more WARs.</i>
HTTP	Hyper Text Transfer Protocol: <i>HTTP is amongst others the protocol used to browse the World Wide Web.</i>
IRC	Internal Relay Chat: <i>HTTP is amongst others the protocol used to browse the World Wide Web.</i>
J2EE	Java 2 Platform, Enterprise Edition: <i>J2EE is a version of Java for developing and deploying enterprise applications.</i>
JAR	Java ARchive: <i>JAR is a file format used to package Java applications</i>
JDK	Java Development Kit: <i>The Java Development Kit is a collection of developer tools for Java developers provided by Sun Microsystems.</i>
JEE	Java Platform, Enterprise Edition: <i>Formerly known as J2EE up to version 1.4, the term JEE is now used for web application running on version 5 of the Java Platform.</i>

Notation	Description
JRE	Java Runtime Environment: <i>The Java Runtime Environment is a collection of libraries and other components provided by Sun Microsystems that allows a computer system to run applets and applications written in the Java programming language.</i>
JSF	Java Server Faces: <i>JSF is a Java-based web application framework.</i>
JSP	Java Server Pages: <i>JavaServer Pages is a server side scripting language developed by Sun Microsystems that is used by Java developers to dynamically generate web pages.</i>
MVC	Model View Controller: <i>A paradigm, stating that the data (model) should be separated from the user interface (view) and the processing (controller).</i>
POM	Project Object Model: <i>The POM is an XML file, which holds the entire configuration for Maven projects.</i>
URL	Uniform Resource Locator: <i>A URL is the unique address for a file that is accessible on the Internet.</i>
W3C	The World Wide Web Consortium: <i>The W3C defines the specifications and guidelines for internet standards.</i>
WAF	Web Application Framework: <i>A WAF provides various functionalities required to build a web application.</i>
WAR	Web ARchive: <i>A WAR file is a JAR used to deploy a collection of resources needed for a Java web application.</i>
XHTML	eXtensible Hypter Text Markup Language: <i>XHTML is a text-based markup language written in XML for the creation of web pages. It has been developed by the W3C as the successor of HTML.</i>
XML	eXtensible Markup Language: <i>XML is a general-purpose markup language for creating special-purpose markup languages, and is used to describe different kinds of data.</i>

Bibliography

- Apache. 2007. "Apache JMeter.". Available at: <http://jakarta.apache.org/jmeter/>.
- Eclipse. 2007. "Eclipse.org home.". Available at: <http://www.eclipse.org>.
- Hermanns, Rainer, Ted Husted and Don Brown. 2006. "Validation Annotation.". Available at: <http://struts.apache.org/2.0.6/docs/validation-annotation.html>.
- JBoss. 2007a. "Apache Reference: mod_proxy.". Available at: <http://labs.jboss.com/portal/jbossas/download/>.
- JBoss. 2007b. "Securing JBoss." *Jboss Wiki* . Available at: <http://wiki.jboss.org/wiki/Wiki.jsp?page=SecureJBoss>.
- Laurie, Ben and Chuck Murcko. 1996. "Apache Reference: mod_proxy.". Available at: http://www.apacheref.com/ref/mod_proxy.html.
- Lightbody, Patrick. 2007. "WebWork (Struts 2) In Action.". Available at: <http://www.infoq.com/presentations/struts-2-webwork-pat-lightbody>.
- Lightbody, Patrick, Rene Gielen, Philip Luppens, Don Brown, Ted Husted and Musachy Barroso. 2007. "Validation.". Available at: <http://struts.apache.org/2.0.6/docs/validation.html>.
- Lightbody, Patrick and Ted Husted. 2006. "Client Side Validation.". Available at: <http://struts.apache.org/2.0.6/docs/client-side-validation.html>.
- Massol, Vincent, Jason va Zyl, Brett Porter, John Casey and Carlos Sanchez. 2006. *Better Builds with Maven*. Mergere Inc.
- McCuaig, Pann. 2003. "Debian Packaging System.". Available at: <http://www.chuug.org/talks/20030325/>.
- Struts. 2007a. "Apache Struts 2 Documentation.". Available at: <http://struts.apache.org/2.0.6/docs/home.html>.

Struts. 2007b. "MailReader Demonstration Application.". Available at:
<http://www.planetstruts.org/struts2-mailreader/>.

Struts. 2007c. "Struts 2.0.6 Distributions.". Available at:
<http://struts.apache.org/download.cgi#struts206>.

Tigris. 2007. "Subclipse Eclipse Plugin.". Available at: <http://subclipse.tigris.org/>.

UCBerkeley. 2006. "Connecting Apache 2.0.## to JBoss-Tomcat via mod_jk2.". Available at: http://sis36.berkeley.edu/projects/streek/howto/apache-mod_jk2-win.html.