Struts Framework Requirements Specification

1. Scope

1.1 Overview

We are going to move Direct(www.topcoder.com/direct) from flex to HTML based platform. This contest is a architecture contest that will mainly focus on defines the front end technical approach to meet the requirements of the project's release 1. Release 1 contains Launch Contest page(s). Clients can use this page to create/update studio (nonsoftware) and software contests. Existing back end services will be used.

1.1.1 Version

1.0

1.2 Logic Requirements

This component provides the framework for handling user requests and integrating them with the services. The framework will be built on top of Struts 2.0.

This component will provide all elements defined in the Struts Framework Class Diagram.

1.2.1 Authentication Interceptor

This interceptor will be responsible for checking if the user info is in session. It will look for session key “user_session”. If found, it allows the action to proceed. If not, it is sent to the login page.

The user session key and the login page will be configurable.

Please note that our current strategy is to use TopCoder’s existing authentication/authorization process that will ensure that the user is valid. The purpose of this interceptor is to ensure that there is no hot-linking directly to any action. So the simple test for the user session is basically testing whether the request comes from an existing session.

Also, there are some ambiguities as to how the TopCoder’s existing authentication/authorization process actually works. The assumption is that there will be some bean (possibly an implementation of jav.security.Principal) in session that can be checked. This will be further clarified.

1.2.2 Logging Interceptor

This interceptor will simply log the exception. Basicaly, when the action is done, it will intercept the result, and if on the stack, there is a AggregateDataModel on the stack and in it there will be a result under key “result”. If its value is an Exception, log its error message and stacktrace using the Logging Wrapper.

be responsible for checking if the user info is in session. It will look for session key “user_session”. If found, it allows the action to proceed. If not, it is sent to the login page.

1.2.3 AbstractAction and AggregateDataModel

To aid in the development of Actions, this component will provide an extension of the

ActionSupport that provides some infrastructure. The AbstractAction will provide access to the a model class and the action name of this action. The extensions of this class must still provide all processing in the execute and validate methods.

Expected extension of the execute method calls for it to instantiate the AggregateDataModel and populate it with response parameters specific to the action. Once that is complete, it will set the model to its namesake property, where it is then available to the value stack for JSON processing.

1.2.4 Transaction handling

Any transactions will be handled externally by the container.

1.2.5 Thread-safety

Since we are operating in a JSP container, threading is not an issue.

1.2.6 Configuration

All configuration will be done via method injection. Each item being injected via a setter will have a corresponding getter.

1.3 Required Algorithms

None

1.4 Example of the Software Usage

The framework will be part of actions that will process user requests via AJAX.

1.5 Future Component Direction

None

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

None

2.1.3 Environment Requirements

- Development language: Java1.5, J2EE 1.5
- Compile target: Java1.5, J2EE 1.5
- Application Server: JBoss 4.0.2
- Informix 11

2.1.4 Package Structure

com.topcoder.service.actions
com.topcoder.service.interceptors

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- User session key
- Login page name
- Logger name

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

- Struts 2.1.1
- Spring 3.0

3.2.2 TopCoder Software Component Dependencies:

- Logging Wrapper 2.0
  - Used for logging errors

3.2.3 Third Party Component, Library, or Product Dependencies:

Json-lib (http://json-lib.sourceforge.net/)

3.2.4 QA Environment:

- Java 1.5/J2EE 1.5
- JBoss 4.0.2
- Informix 11
- MySQL 5.1
- Struts 2.1.8.1
- Spring 3.0
- Javascript 1.8
- Mozilla Firefox 2.0/3.0
- IE 6.0/7.0
- Google Chrome
- Safari 3/4

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of the TopCoder UML Tool.