Struts Actions 3 Requirements Specification

1. Scope

1.1 Overview

As we progress in upgrading the Direct application, the next thing we are focusing on is the complete flow of launching a contest. This includes everything from entering the basic data for a contest to payment, spec review, provisioning, etc. We currently have a basic flow for launching a contest created in the service layer and being assembled for the html version of Direct. This contest will dig deeper into the process of launching a contest to address the requirements introduced and clarified by the requirements that are referenced in this contest.

This component provides the struts actions to get wiki link, get contest receipt, send contest receipt by email, view game plan, view project's contests, get active contest prize, increase prize of active contest, get active contest schedule, and extend active contest schedule.

1.1.1 Version

1.0

1.2 Logic Requirements

Each of the following actions will be designed as a Struts action with AJAX handling capability. It is up to the designer to make best use of Struts capabilities, including interceptors and base action classes. The only client of this application will be the JSPs, so there are no APIs or structural requirements. The designer is free to refactor any interceptor or action as long as requirements are met.

This component is responsible for several actions that are used to perform requests from the Frontend.

1.2.1 Struts Actions

Here we have some general information about the actions:

- All actions will extend the AbstractAction class from the struts framework component.
- Javabean properties (i.e. getXXX/setXXX) should be used for all action data. The input data will be provided as request parameters and will need to be mapped to these setters using JavaBean conventions.
- The design may rely on some data conversion to take place before the action, such as converting dates, but the designer must specify what those conversions must be.
- The execute method will place all result data in the AggregateDataModel then it will set it to the action to make it available in the ValueStack. Essentially, we expect return parameters to be sent back. Use simple "return" key for the model.

1.2.2 Validation

Validation here will comprise the user input as specified in the use cases provided in the ARS for each relevant action. Validation errors would be packaged in the ValidationErrors entity with each property that fails validation placed in the ValidationErrorRecord entity. ValidationErrorRecord contains a messages array field that allows multiple validation errors to be provided per field (for example, a field could be too long and contain incorrect format).

1.2.3 Get Wiki Link

This action will retrieve the contest data to render links for creating contest's wiki page.

Input:
contestId – the contest id.

studio – flag indicating it's a software or studio contest

Process
The ContestServiceFacade's getContest (for studio contest) or getSoftwareContestByProjectId (for software contest) should be called to get the studio or software contest.

Then create wiki page link as below:
http://www.topcoder.com/wiki/pages/createpage.action?spaceKey=docs&title=My%20Page%20Title&location=competitions

The link above will bring user to the create page action in the wiki with the title and space already set. We want to set the space to docs, the locations to competitions, and the title to the [contest type] - [contest name].

For studio contest, the [contest type] and [contest name] can be retrieved from the StudioContest's category and name attributes.

For software contest, the [contest type] and [contest name] can be retrieved from the SoftwareCompetition's assetDTO.name and category.name attributes.

Result:
Wiki page link

1.2.4 *Get Contest Receipt*

This action will retrieve the contest receipt for display.

Input:
contestId – the contest id.
studio – flag indicating it's a software or studio competition

Process:
The DirectServiceFacade's getContestReceiptData method will be called with the contestId and studio flag to get the contest receipt data.

Result:
The retrieved contest receipt data.

1.2.5 *Send Contest Receipt by Email*

This action will retrieve the contest receipt and send it by email.

Input:
contestId – the contest id.
studio – flag indicating it's a software or studio competition
additionalEmailAddresses – as explained below

Data Element	Description	Format	R?
Additional E-mail Address(es)	One or more target e-mail addresses for receipt.	String, max 4096 chars, can be empty. It has to contain valid e-mail addresses (like "name@some.domain").	N

The e-mail addresses will be separated (by "," or by ";" character).

Process:
The DirectServiceFacade's sendContestReceiptDataByEmail method will be called with the contestId, studio flag and additional email addresses to send the contest receipt.

Result:
The returned contest receipt data.

1.2.6 *View Game Plan*

The action will retrieve project's game plan.

Input:
projectId – the project id

Process
The DirectServiceFacade's getProjectGamePlan method is called to retrieve the game plan for display.

Result
Retrieved contest plans.

1.2.7 *View Project's Contest*

The action will retrieve project's contests and project summary data for display.

Input:
projectId – the project id

Process:
The ContestServiceFacade.getCommonProjectContestDataByPID method is called to retrieve the project's contests.

And the project summary data will be retrieved by calling the search projects method defined in the ContestServiceFacade (Refer to http://www.topcoder.com/wiki/display/docs/TC+Direct+Search – section 1.1.1.1 search projects).

Result:
The retrieved project's contests and project summary data.

1.2.8 *Get Active Contest Prizes*

The action will retrieve the contest's prizes (including milestone prizes if available) for display.

Input:
contestId – the contest id.
studio – flag indicating it's a software or studio competition

Process:
The DirectServiceFacade.getContestPrize method to get the contest prizes by the contest id and studio flag

Result:
Retrieved contest prize

1.2.9 *Increase Prize of Active Contest*

The user can increase the prizes for the already active contest and (optionally) for the milestone.

Input:
contestId – the contest id.
studio – flag indicating it's a software or studio competition
contest prizes – see 1.2.9.1
milestone prizes – see 1.2.9.2

1.2.9.1 Contest Prizes

For Studio contest:

Data Element	Description	Format	R?
X Place	The prize amount for the submission winning place X (where X is from 1 to 9). Please note, the 2nd prize must be at least 20% of the first prize. All nine prize fields can be increased, even if some of them were not used (assigned to $0).	Positive number, or 0. It will be money value. By default it is displayed in whole dollars, but will also display cents if cents are non-zero. It will have "$" prefix.	the 1st and 2nd place are required. Other – optional

For Software contests:

Data Element	Description	Format	R?
1st Place	The top prize for the contest winner	Positive number, or 0. It will be money value. By default it is displayed in whole dollars, but will also display cents if cents are non-zero. It will have "$" prefix.	Y
2nd Place	The second place prize for the competitor. Please note, the 2nd prize must be at least 20% of the first prize	Positive number, or 0. It will be money value. By default it is displayed in whole dollars, but will also display cents if cents are non-zero. It will have "$" prefix.	Y

The user can add more money to the contest prize fields. The fields will be editable, but values, lesser than previous amounts will not accepted.

1.2.9.2 Milestone Prizes

The equal milestone prize (one option):

Data Element	Description	Format	R?
Pay $	The payment amount for top submissions to be paid in the milestone	Positive number, or 0. It will be money value. By default it is displayed in whole dollars, but will also display cents if cents are non-zero. It will have "$" prefix.	Y
for each submission up to X submissions	Where X is a number of the top submissions to be paid in the milestone. Please note, this value can be also increased.	Single item selection list, each item is integer. Values are 2, 5, 10 are allowed.	Y

Up-to 3 different prizes for the milestone placements (another option):

Data Element	Description	Format	R?
Pay $ for place <X>	The payment amount for top submissions to be paid in the milestone. The X will be from 1 to 3. Please note, X value can be also increased up-to 3.	Positive number, or 0. It will be money value. By default it is displayed in whole dollars, but will also display cents if cents are non-zero. It will have "$" prefix.	Y

The user can add more money to the milestone prize fields. The fields will be editable, but values, lesser than previous amounts will not accepted.

Process:
The DirectServiceFacade.updateActiveContestPrize method to update the active contest's prizes, the contest prizes and milestone prizes will both be updated.

Result:
None

1.2.10 *Get Active Contest Schedule*

The action will get active contest's schedule for display.

Input:
contestId – the contest id.
studio – flag indicating it's a software or studio competition

Process:
The DirectServiceFacade.getContestSchedule method is called to get the contest schedule by the contest id and studio flag.

Result
Retrieved contest schedule

1.2.11 *Extend Active Contest Schedule*

The user can add more time to the registration, milestone and submission phases of the active competition.

Input:
contestId – the contest id.
studio – flag indicating it's a software or studio competition
extensions – see below

Data Element	Description	Format	R?
Extend Registration for (days)	The extension duration for the registration phase	Positive integer, in days by default.	N
Extend Milestone for (days)	The extension duration for the milestone phase	Positive integer, in days by default.	N
Extend	The extension duration for the	Positive integer, in days by	N

Data Element	Description	Format	R?
Submission for (days)	submission phase	default.	

Please note, the user can enter the extension in hours, by adding “h” postfix to the number (like “12h”).

Process:
The DirectServiceFacade.extendActiveContestSchedule method will be called to extend the active contest’s schedule.

Result:
None.

1.2.12 *Struts action mapping*

The designer does not need to provide a mapping file for struts actions. All struts and spring files will be provided by the assembly.

1.2.13 *Services*

The actions will directly use façade classes. These will be injected.

1.2.14 *Logging and Error Handling*

The actions will not perform logging and any error handing. If any errors occur, they actions will simply put the Exceptions into the model as a “result”. There will an interceptor that will process the logging of this.

1.2.15 *Transaction handling*

Any transactions will be handled externally by the container at the level of the façade. Nothing needs to be done in this component.

1.2.16 *Thread-safety*

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

1.2.17 *Configuration*

All configurations 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 actions will handle contest launching requests.

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

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
- Service Facades

3.2 Technical Constraints

3.2.1 *Are there particular frameworks or standards that are required?*

- Struts 2.1.1
- Spring 3.0
- EJB 3.0

3.2.2 *TopCoder Software Component Dependencies:*

- Struts Framework 1.0
- Contest Service Façade 1.0
- Direct Service Façade 1.0

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 *Third Party Component, Library, or Product Dependencies:*

None

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.