Direct Service Façade 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 façade API for the struts actions.

1.2 Logic Requirements

The architecture diagram is provided.
This design is responsible for implementing the classes/interfaces in com.topcoder.service.facade.direct package.

The DirectServiceFacade will be deployed as web service either both local and remote EJB interface.

1.2.1 Get Contest Receipt Data

The getContestReceiptData method will get the contest receipt data for the studio or software competition.

The returned ContestReceiptData contains the following properties:

Data Element	Description
Project Name	The name of the project for this contest
Contest Name	The name of the contest. It will appear to competitors in a list of active contests.
Contest Fee	The fee for the contest
Start Date	The date, when the contest will start
Contest Type	The type of the contest (including the sub-type for the studio)
Prizes	The prize for the contest competitors
Milestone date	The assigned date for the milestone
Status	The contest status. For example: Scheduled, Draft, Active.
Contest Total Cost	The total amount of contest cost. It will be a sum of Contest Fee and Prizes values.
End date	The date when the contest will finish
Company Name	The name of the user's company
Address	The user address where to send bills
City	The city of the user
State	The state of the user (if the user is from US)
Zip code	The ZIP code of the user
Country	The country of the user
Phone	The phone number of the user
E-mail	The e-mail address of the user
Purchase Order #	The confirmation number of the order
Invoice Terms	The description of the invoice terms
Reference	The confirmation number of the order

Data Element	Description
Number	

The project data can be collected from the studio or software contest returned by ContestServiceFacade's getContest or getSoftwareContestByProjectId method.

The user contact data can be collected from the UserService (from User Service component).

1.2.2 *Send Contest Receipt by Email*

The sendContestReceiptByEmail method will send the receipt email to the following users:

- Contest activating user's e-mail address (ContestReceiptData.email)
- The users having permissions for the activated contest.
- The specified email addresses

It should call the getContestReceiptData method to get the ContestReceiptData first, and the returned ContestReceiptData will be used to populate the email content.

The email template needs to be configurable, and it's expected to contain all the fields defined in the ContestReceiptData.

1.2.3 *Get Project Game Plan*

The getProjectGamePlan will get the project's game plan.

It will call ContestServiceFacade.getCommonProjectContestDataByPID method by the project id to get all the contests belonging to the project. And then collect corresponding data from those contests to return.

1.2.4 *Update Active Contest Prize*

The updateActiveContestPrize method will update the active contest's prize.

The ContestPrize entity contains the following attributes:

Data Element	Description	Validation
contestId	the contest id.	Required
studio	indicating the contest is studio contest or software competition	Required
contestPrizes	the contest prizes for the winning submissions. the contestPrizes[X] means the prize for the submission winning the (X+1)th place	Must be non-empty. The element must be positive number, or 0. The 2nd prize must be at least 20% of the first prize if it's present.
milestonePrizes	the milestone prizes for the winning submissions. the milestonePrizes[X] means the prize for the submission winning the (X+1)th place	Can be null or empty. The element must be positive number, or 0.
equalMilestonePrize	indicating all the prizes in the milestonePrizes must be equal.	Required

The ContestServiceFacade's updateSoftwareContest & updateContest can be used to update prizes data to software or studio contest.

1.2.5 *Get Contest Prize*

The getContestPrize method will get the contest prize for the given contest (studio or software

competition). The corresponding ContestPrize entity described above will be returned.

The ContestServiceFacade's getContest or getSoftwareContestByProjectId can be used to retrieve the prize data from the returned studio or software contest.

1.2.6 *Extend Active Contest Schedule*

The extendActiveContestSchedule method will update the active contest's schedule.

The ContestScheduleExtension entity contains the following attributes:

Data Element	Description	Validation
contestId	the contest id.	Required
studio	indicating the contest is studio contest or software competition	Required
Extend Registration by (hours)	The extension duration for the registration phase	Positive integer, optional.
Extend Milestone by (hours)	The extension duration for the milestone phase	Positive integer, optional.
Extend Submission by (hours)	The extension duration for the submission phase	Positive integer, optional.

The ContestServiceFacade's updateSoftwareContest & updateContest can be used to update schedule data to software or studio contest.

Note that if specific phase is extended, all the phases' scheduled start date and end date should be recalculated accordingly. And the registration phase should be reopened after it's extended.

1.2.7 *Get Contest Schedule*

The getContestSchedule method will retrieve the contest schedule for the given contest.

The ContestSchedule entity contains the following attributes:

Data Element	Description
contestId	the contest id.
studio	indicating the contest is studio contest or software competition
Registration deadline	The end of the registration phase for the contest
Registration Duration (hours)	The duration of the registration phase
Milestone deadline	The end of the milestone phase for the contest. The milestone deadline & duration are only applicable if the contest contains multiple rounds.
Milestone Duration (hours)	The duration of the milestone phase
Submission deadline	The end of the submission phase for the contest
Submission Duration (hours)	The duration of the submission phase

The ContestServiceFacade's getContest or getSoftwareContestByProjectId can be used to retrieve the schedule data from the returned studio or software contest.

1.2.8 *Repost Contest*

The user can repost previously cancelled, failed or completed contest. The copy of the contest will be created and placed to the Draft state.

The repostSoftwareContest is for this purpose, and it will return the id of the reposted contest. It will do the followings:

- Make a full clone copy of the current contest and persist it as a repost for the current project.
- Modify the starting, ending and (optionally) milestone dates of the reposted contest by using the same durations as in the original contest, but setting the start date for the tomorrow 9 AM.
- Set the “Draft” status to the reposted contest.
- Finally return the id of the newly created contest.

Its implementation is quite similar to the existing reOpenSoftwareContest method in the ContestServiceFacade.

1.2.9 *Create New Component Version*

The user can create new version of the Software Design/Development component, which was already completed. The copy of the contest will be created and placed to the Draft state.

The createNewVersionForDesignDevContest method will do the followings:

- Make a full clone copy of the current contest and persist it as a new version contest for the current project.
- Automatically increase the component version (like changing to v. 1.1 from v.1.0).
- Modify the starting, ending and (optionally) milestone dates of the new version contest by using the same durations as in the original contest, but setting the start date for the tomorrow 9 AM.
- Set the “Draft” status to the new version contest.
- Finally return the id of the newly created contest.

Its implementation is quite similar to the existing createNewVersionForDesignDevContest method in the ContestServiceFacade.

1.2.10 *Delete Contest*

The user can remove any of his/her contests. First, the user can break active contest and it will move to the “cancelled” state. Second, it will remove draft/cancelled/failed contest from the project.

For software contest, if the contest is active, its status needs to be changed to ‘9 - Cancelled, Client Request’; if the contest is inactive, its status needs to be changed to ‘3 – Deleted’. This can be achieved by updating SoftwareCompetition.projectHeader.projectStatus to persistence.

For studio contest, if the contest is active, its status needs to be changed to ‘16 – Cancelled’; if the contest is inactive, its status needs to changed to ‘14 – Abandoned’. This can be achieved by updating StudioCompetition.contestData.statusId and StudioCompetition.contestData.detailedStatusId to persistence.

1.2.11 *Get Parent Projects*

The getParentProjects method will retrieve the parent projects the given project depends on.

It will call ProjectLinkManager.getSourceProjectLinks to get the projects the given project

depends on.

ProjectLinkManager is defined in the Project Management component.

1.2.12 *Get Child Projects*

The getChildProjects method will retrieve the child projects depending on the given project.

It will call ProjectLinkManager.getDestProjectLinks to get the projects depending on the given project.

1.2.13 *Update Project Links*

The updateProjectLinks method will update the parent projects and child projects for the given project.

It will call ProjectLinkManager.updateProjectLinks to update the project links.

1.2.14 *Get Project Budget*

The getProjectBudget method will get the billing project's budget.

It will use the updated ProjectDAO (from the Client Project Entities DAO component) to get the project's budget.

1.2.15 *Update Project Budget*

The updateProjectBudget method will update the billing project's budget.

It will use the updated ProjectDAO (from the Client Project Entities DAO component) to update the project's budget.
It should also send an email about budget change information (billing project name, old budget, new budget and the changed amount) to the user.

1.2.16 *Get Specification Review State*

The getSpecReviewState method will get the specification review state.

The SpecReviewState attributes are explained below:

Data Element	Description
Deadline	The date/time when the specification review will expire
Is Delayed	The flag defining whether the specification review is delayed or not
Delay hours	The amount of hours for review delay
status	The status of the spec review. It will contain the following values: ● Nobody registered – assigned when the specification review competition just posted and no user registered to it yet, ● Review started – assigned when the TC Member registered for the specification review competition, ● Review scorecard provided – assigned when the TC Member finished reviewing of the contest specification, ● Final Fix Review – assigned when the user updated specification according to the TC Member review and send the final fix, ● Final Fix rejected – assigned if the TC Member rejected the final fix from the user, ● Final Fix accepted – assigned if the TC Member agreed with the user's final fix, ● Review completed – assigned when the user finished the specification review competition.

Data Element	Description

The getProject method in ProjectManager (from Project Management component) can be called to retrieve the project data. The returned project's projectSpec.projectSpecId attribute represents the specification project id.

The getPhases method in PhaseManager (from Phase Management component) can be called by the specification project id to get specification project phases.

Then we can collect the data from the returned phases.

1.2.17 Transaction handling

The transaction is managed by the container.

1.2.18 Thread-safety

The implementation needs to be thread-safe.

1.2.19 Configuration

The configuration needs to be done via resource injection.

1.2.20 Logging

The method entry and exit should be logged with DEBUG level in DirectServiceFacade implementation, and the exceptions should be logged with ERROR level in DirectServiceFacade implementation.

1.2.21 Exception

The methods in DirectServiceFacade will throw the DirectServiceFacadeException, and designers are free to provide child exceptions to it.

1.3 Required Algorithms

None

1.4 Example of the Software Usage

It provides service methods for TopCoder Launch Contest application.

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.facade.direct

3. Software Requirements

3.1 Administration Requirements

3.1.1 *What elements of the application need to be configurable?*

- Service Facades
- Email Templates

3.2 Technical Constraints

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

- EJB 3.0

3.2.2 *TopCoder Software Component Dependencies:*

- Contest Service Façade 1.0
- Project Management 1.0
- Client Project Entities DAO 1.0
- Email Engine 1.0
- Document Generator 1.0
- User Service 1.0
- Logging Wrapper 2.0
- Base Exception 2.0
- Security Manager 1.1
- Phase Management 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

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.