[TopCoder]

Direct Service Facade 1.0 Component Specification

1. Design

As TopCoder progresses in upgrading the Direct application, the next thing TopCoder is 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. TopCoder currently has a basic flow for launching a contest created in the service layer and being assembled for the html version of Direct. This component digs deeper into the process of launching. It provides façade API for the struts actions.

This component defines DirectServiceFacade interface and provides its implementation.

1.1 Design Patterns

Strategy pattern – this component provides DirectServiceFacade interface and its implementation that can possibly be used in some external strategy context.

DAO/DTO pattern – DirectServiceFacade acts as a DAO for ContestReceiptData, ContestPrize, ContestSchedule, ContestPlan, Project, ProjectBudget and SpecReviewState DTOs.

Façade pattern – DirectServiceFacadeBean acts as a façade to ProjectManager, ContestServiceFacade, PhaseManager, ProjectDAO, UserService and ProjectLinkManager interfaces.

1.2 Industry Standards

EJB 3.0, IoC

1.3 Required Algorithms

1.3.1 Logging

This component must perform logging in all public methods of DirectServiceFacadeBean.

All information must be logged using log:Log attribute. If log attribute is null, then logging is not required to be performed.

In all mentioned methods method entrance with input arguments, method exit with return value and call duration time must be logged at DEBUG level. It's not required to log method exit when method throws an exception.

All errors (for all thrown exceptions) must be logged at ERROR level with exception message and stack trace.

1.3.2 Format of email templates used by DirectServiceFacadeBean

When sending email messages to the users, DirectServiceFacadeBean generates email message title and body from templates. These templates are provided via resource injection and can contain the following variables (variable names are self-explanatory):

Variables supported in sendContestReceiptByEmail():

```
%contestId%
%studio%            (true/false)
%projectName%
%contestName%
%contestFee%
%startDate%
%contestType%
%prizes%            (in format prize1, prize2, ...)
%milestoneDate%     (in format yyyy-MM-dd HH:mm)
%status%
%contestTotalCost%
%endDate%           (in format yyyy-MM-dd HH:mm)
```

```
%companyName%
%address%
%city%
%state%
%zipCode%
%country%
%phone%
%email%
%purchaseOrderNo%
%invoiceTerms%
%referenceNo%
```

Variables supported in updateProjectBudget():

```
%billingProjectName%
%oldBudget%
%newBudget%
%changedAmount%
```

1.4 Component Class Overview

ContestPlan

This class is a container for contest plan data. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no validation in the setters.

ContestPrize

This class is a container for contest prize data. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no validation in the setters.

ContestReceiptData

This class is a container for contest receipt data. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no validation in the setters.

ContestSchedule

This class is a container for contest schedule data. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no validation in the setters.

ContestScheduleExtension

This class is a container for contest schedule extension data. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no validation in the setters.

DirectServiceFacade [interface]

This interface represents a direct service facade. It provides methods for retrieving and sending by email contest receipts, retrieving and updating contest prize data, retrieving and updating contest schedule, reposting and creating new version contests, deleting contests, retrieving and updating project budget, retrieving and updating links between projects and retrieving the state of the spec review and project game plan.

DirectServiceFacadeBean

This class is an EJB that implements DirectServiceFacade business interface. This bean uses Logging Wrapper component to log exceptions and debug information and Email Engine component to send email messages. This class uses pluggable implementations of ContestServiceFacade, UserService, ProjectDAO, ProjectManager, ProjectLinkManager and PhaseManager interfaces to access data in persistence.

DirectServiceFacadeLocal [interface]

This interface represents the local interface for DirectServiceFacade session bean. It extends that interface and provides no additional methods.

DirectServiceFacadeRemote [interface]

This interface represents the remote interface for DirectServiceFacade session bean. It extends that interface and provides no additional methods.

ProjectBudget
This class is a container for project budget data. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no validation in the setters.

SpecReviewState
This class is a container for specification review state data. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no validation in the setters.

SpecReviewStatus [enum]
This is an enumeration for spec review statuses.

1.5 Component Exception Definitions

ContestNotFoundException
This exception is thrown by implementations of DirectServiceFacade when studio/software contest with the specified ID cannot be found in persistence.

DirectServiceFacadeConfigurationException
This exception is thrown by DirectServiceFacadeBean when error occurs while initializing this bean.

DirectServiceFacadeException
This exception is thrown by implementations of DirectServiceFacade when some other error occurred. Also this exception is used as a base class for other specific custom exceptions.

EmailSendingException
This exception is thrown by implementations of DirectServiceFacade when some error occurs while sending an email message.

ProjectNotFoundException
This exception is thrown by implementations of DirectServiceFacade when project with the specified ID cannot be found in persistence.

1.6 Thread Safety

This component is thread safe.

Implementations of DirectServiceFacade, DirectServiceFacadeLocal and DirectServiceFacadeRemote must be thread safe when entities passed to them are used in thread safe manner by the caller.

DirectServiceFacadeBean is mutable and not thread safe. But it is always used in thread safe manner in EJB container because its state doesn't change after initialization. Instances of ContestServiceFacade, UserService and ProjectDAO used by this class are thread safe. Instances of ProjectManager, ProjectLinkManager and PhaseManager are not thread safe, thus all call of their methods are synchronized in this class. This bean assumes that transactions are managed by the container.

ContestReceiptData, ContestPrize, ContestSchedule, ContestPlan, ContestScheduleExtension, ProjectBudget and SpecReviewState are mutable and not thread safe entities.

SpecReviewStatus is immutable and thread safe enum.

DirectServiceFacadeBean uses container managed transactions. All methods of this class that can change data in the persistence have @TransactionAttribute(TransactionAttributeType.REQUIRED) annotation.

2. Environment Requirements

2.1 Environment

Development language: Java1.5, J2EE 1.5

Compile target: Java1.5, J2EE 1.5
QA Environment: JBoss 4.0.2, Informix 11

2.2 TopCoder Software Components

Contest Service Façade 1.1 – defines ContestServiceFacade and entities used in this component.

User Service 1.0 – defines UserService and entities used in this component.

Project Service 1.2 – defines ProjectService and entities used in this component.

Studio Service 1.3 – defines ContestData and some other entities used in this component.

Catalog Entities 1.2 – defines Status entity used in this component.

Client Project Entities DAO 1.1 – defines ProjectDAO and entities used in this component.

Project Management 1.0.1 – defines ProjectLinkManager and entities used in this component.

Phase Management 1.0.4 – defines PhaseManagement interface used in this component.

Project Phases 2.0.1 – defines contest phases entities used in this component.

Project Service 1.1 – defines competition entities used in this component.

Project Services 1.1 – defines FullProjectData entity used in this component.

Base Exception 2.0 – is used by custom exceptions defined in this component.

Logging Wrapper 2.0 – is used for logging errors and debug information.

Document Generator 3.0 – is used for generating text of email messages from templates.

Email Engine 3.1 – is used for sending email messages.

NOTE: The default location for TopCoder Software component jars is../lib/tcs/COMPONENT_NAME/COMPONENT_VERSION relative to the component installation. Setting the tcs_libdir property in topcoder_global.properties will overwrite this default location.

2.3 Third Party Components

None

3. Installation and Configuration

3.1 Package Name

com.topcoder.service.facade.direct
com.topcoder.service.facade.direct.ejb

3.2 Configuration Parameters

3.2.1 EJB configuration of DirectServiceFacadeBean

DirectServiceFacadeBean is configured with the following EJBs via container injection:

EJB name	Description	Type
ejb/ContestServiceFacade	The contest service facade used by this class.	ContestServiceFacade
ejb/UserService	The user service used by this class.	UserService
ejb/ProjectDAO	The project DAO used by this class.	ProjectDAO

Also DirectServiceFacadeBean uses the following resources injected by EJB container:

Resource name	Description	Values

loggerName	The name of the Logger Wrapper logger to be used by this class for logging errors and debug information. When not specified, logging is not performed.	String. Not empty. Optional.
projectManagerClassName	The full class name of project manager to be used by this class.	String. Not empty. Required.
projectManagerNamespace	The namespace of project manager to be used by this class.	String. Optional.
projectLinkManagerClassName	The full class name of project link manager to be used by this class.	String. Not empty. Required.
projectLinkManagerNamespace	The namespace of project link manager to be used by this class.	String. Optional.
phaseManagerClassName	The full class name of phase manager to be used by this class.	String. Not empty. Required.
phaseManagerNamespace	The namespace of phase manager to be used by this class.	String. Optional.
receiptEmailTitleTemplateText	The template text of receipt email title. See section 1.3.2 for the list of the supported template variable.	String. Required.
receiptEmailBodyTemplateText	The template text of receipt email body. See section 1.3.2 for the list of the supported template variable.	String. Required.
budgetUpdateEmailTitleTemplateText	The template text of budget update notification email title. See section 1.3.2 for the list of the supported template variable.	String. Required.
budgetUpdateEmailBodyTemplateText	The template text of budget update notification email body. See section 1.3.2 for the list of the supported template variable.	String. Required.
emailSender	The address of the email sender.	String. Not empty. Required.

3.3 Dependencies Configuration

Please see docs of all the dependency components to configure them properly.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute ‘ant test’ within the directory that the distribution was extracted to.

4.2 Required steps to use the component

Please see the demo.

4.3 Demo

4.3.1 API usage

```
// Get direct service facade
Context context = new InitialContext();
DirectServiceFacade directServiceFacade =
```

```
    (DirectServiceFacadeRemote) context.lookup("DirectServiceFacadeBean/remote");

TCSubject tcSubject = ...

// Retrieve the contest receipt data for studio contest with ID=1001
ContestReceiptData contestReceiptData =
    directServiceFacade.getContestReceiptData(tcSubject, 1001, true);

// Send contest receipt by email for this contest
String[] additionalEmailAddrs = new String[] { "user@mail.com" };
ContestReceiptData contestReceiptData =
  directServiceFacade.sendContestReceiptByEmail(tcSubject, 1001, true, additionalEmailAddrs);

// Retrieve the contest prize for software contest with ID=2001
ContestPrize contestPrize = directServiceFacade.getContestPrize(tcSubject, 2001, false);

// Update first place prize for this contest
contestPrize.getContestPrizes()[0] += 100;
directServiceFacade.updateActiveContestPrize(tcSubject, contestPrize);

// Retrieve the contest schedule
ContestSchedule contestSchedule =
    directServiceFacade.getContestSchedule(tcSubject, 2001, false);

// Extend registration phase by 24 hours
ContestScheduleExtension extension = new ContestScheduleExtension();
extension.setExtendRegistrationBy(24);
directServiceFacade.extendActiveContestSchedule(tcSubject, extension);

// Repost software contest with ID=2001
long newId = directServiceFacade.repostSoftwareContest(tcSubject, 2001);

// Create new version for design contest with ID=2002
newId = directServiceFacade.createNewVersionForDesignDevContest(tcSubject, 2002);

// Delete software contest with ID=2002
directServiceFacade.deleteContest(tcSubject, 2002, false);

// Retrieve the project game plan for project with ID=1
List<ContestPlan> projectGamePlan = directServiceFacade.getProjectGamePlan(tcSubject, 1);

// Retrieve the parent projects for project with ID=2001
List<Project> parentProjects = directServiceFacade.getParentProjects(tcSubject, 2001);

// Retrieve the child projects for project with ID=2001
List<Project> childProjects = directServiceFacade.getChildProjects(tcSubject, 2001);

// Update project links (remove all links)
long[] parentProjectIds = new long[0];
long[] childProjectIds = new long[0];
directServiceFacade.updateProjectLinks(tcSubject, 2001, parentProjectIds, childProjectIds);

// Retrieve the project budget for billing project with ID=1
double projectBudget = directServiceFacade.getProjectBudget(tcSubject, 1);

// Update project budget (increase by 1000)
ProjectBudget projectBudget = directServiceFacade.updateProjectBudget(tcSubject, 1, 1000);

// Retrieve the spec review state for contest with ID=2001
SpecReviewState specReviewState = directServiceFacade.getSpecReviewState(tcSubject, 2001);
```

4.3.2 *Sample EJB configuration (ejb-jar.xml)*

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation=
    "http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd"
         version="3.0">
    <description>Direct Service Facade EJB</description>
    <display-name>Direct Service Facade EJB</display-name>
    <enterprise-beans>
```

```
        <session>
            <ejb-name>DirectServiceFacadeBean</ejb-name>
            <remote>
              com.topcoder.service.facade.direct.ejb.DirectServiceFacadeRemote
            </remote>
            <local>
              com.topcoder.service.facade.direct.ejb.DirectServiceFacadeLocal
            </local>
            <ejb-class>
              com.topcoder.service.facade.direct.ejb.DirectServiceFacadeBean
            </ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
            <env-entry>
                <env-entry-name>loggerName</env-entry-name>
                <env-entry-type>java.lang.String</env-entry-type>
                <env-entry-value>direct_service_facade_log</env-entry-value>
            </env-entry>
            <env-entry>
                <env-entry-name>projectManagerClassName</env-entry-name>
                <env-entry-type>java.lang.String</env-entry-type>
                <env-entry-value>
                  com.topcoder.management.project.ProjectManagerImpl
                </env-entry-value>
            </env-entry>
            <env-entry>
                <env-entry-name>projectLinkManagerClassName</env-entry-name>
                <env-entry-type>java.lang.String</env-entry-type>
                <env-entry-value>
                  com.topcoder.management.project.link.ProjectLinkManagerImpl
                </env-entry-value>
            </env-entry>
            <env-entry>
                <env-entry-name>phaseManagerClassName</env-entry-name>
                <env-entry-type>java.lang.String</env-entry-type>
                <env-entry-value>
                  com.topcoder.management.phase.DefaultPhaseManager
                </env-entry-value>
            </env-entry>
            <env-entry>
                <env-entry-name>receiptEmailTitleTemplateText</env-entry-name>
                <env-entry-type>java.lang.String</env-entry-type>
                <env-entry-value>Receipt for contest "%contestName%</env-entry-value>
            </env-entry>
            <env-entry>
                <env-entry-name>receiptEmailBodyTemplateText</env-entry-name>
                <env-entry-type>java.lang.String</env-entry-type>
                <env-entry-value>Project Name: %projectName%
Contest Fee: %contestFee%
Start Date: %startDate%
Prizes: %prizes%
Total Cost: %contestTotalCost%
See details here: http://topcoder.com/tc?module=ContestDetails&id=%contestId%
                </env-entry-value>
            </env-entry>
            <env-entry>
                <env-entry-name>budgetUpdateEmailTitleTemplateText</env-entry-name>
                <env-entry-type>java.lang.String</env-entry-type>
                <env-entry-value>Budget Updated</env-entry-value>
            </env-entry>
            <env-entry>
                <env-entry-name>budgetUpdateEmailBodyTemplateText</env-entry-name>
                <env-entry-type>java.lang.String</env-entry-type>
                <env-entry-value>Project Name: %billingProjectName%
Old Budget Amount: %oldBudget%
New Budget Amount: %newBudget%
</env-entry-value>
            </env-entry>
            <env-entry>
                <env-entry-name>emailSender</env-entry-name>
                <env-entry-type>java.lang.String</env-entry-type>
```

```
                <env-entry-value>service@topcoder.com</env-entry-value>
            </env-entry>
            <ejb-ref>
                <ejb-ref-name>ejb/ContestServiceFacade</ejb-ref-name>
                <ejb-ref-type>Session</ejb-ref-type>
                <remote>
                  com.topcoder.service.facade.contest.ejb.ContestServiceFacadeRemote
                </remote>
            </ejb-ref>
            <ejb-ref>
                <ejb-ref-name>ejb/UserService</ejb-ref-name>
                <ejb-ref-type>Session</ejb-ref-type>
                <remote>com.topcoder.service.user.UserServiceRemote</remote>
            </ejb-ref>
            <ejb-ref>
                <ejb-ref-name>ejb/ProjectDAO</ejb-ref-name>
                <ejb-ref-type>Session</ejb-ref-type>
                <remote>com.topcoder.clients.dao.ejb3.ProjectDAORemote</remote>
            </ejb-ref>
        </session>
    </enterprise-beans>
</ejb-jar>
```

4.3.3 *Sample receipt email message*

```
--------------------------------------------------------------------------------
From: service@topcoder.com
To: user@server.com
Bcc: user@mail.com
Subject: Receipt for contest "Sample Contest%
--------------------------------------------------------------------------------
Project Name: Sample Project
Contest Fee: 240
Start Date: 2010-20-05 09:00:00
Prizes: 800, 400
Total Cost: 1440
See details here: http://topcoder.com/tc?module=ContestDetails&id=1001
--------------------------------------------------------------------------------
```

5. Future Enhancements

None