Java Custom Cockpit Project Management Persistence

1. Scope

1.1 Overview

This component provides an implementation of the ProjectPersistence interface from the Project Management component. The implementation will adapt the ContestManagerBean from the Studio Contest Manager component to provide the necessary functionality. This component will allow the Project Management component to work with studio contests.

1.1.1 Version

1.0

1.2 Logic Requirements

1.2.1 Persistence Implementation

The persistence implementation will utilize the ContestManager interface, from the Studio Contest Manager component, which will be deployed in an EJB 3.0 container. The constructors for the persistence implementation should expect either a ContestManagerRemote bean being directly provided, or a name that can be used with JNDI to retrieve the bean.

1.2.2 Adapter pattern

This component will adapt the entities from the "Contest and Submission Entities" component to work with the entities from Project Management. The mapping should look something like this:

Project	**Contest**
Project.id	Contest.contestId
Project.name	Contest.name

Project.creationUser	Contest.createdUser
Project.projectStatus	Contest.status
Project.projectCategory	Contest.contestChannel

All other Contest properties should be added to the "properties" Map in the Project entity created. This mapping will be necessary both when creating the Project entity in the "get" methods. It will also be used when converting the Project entity back to a Contest entity in the "create" and "update" methods

ProjectStatus	**ContestStatus**
ProjectStatus.id	ContestStatus.contestStatusId
ProjectStatus.name	ContestStatus.name
ProjectStatus.description	ContestStatus.description

ProjectCategory	**ContestChannel**
ProjectCategory.id	ContestChannel.contestChannelId
ProjectCategory.name	ContestChannel.name
ProjectCategory.description	ContestChannel.description
ProjectCategory.projectType	ContestChannel.fileType

ProjectType	**StudioFileType**
ProjectType.id	StudioFileType.studioFileType
ProjectType.name	StudioFileType.extension
ProjectType.description	StudioFileType.description

1.2.3 Method mapping

The following table outlines the mapping between method calls to the persistence and method calls to the ContestManager interface:

Persistence	Contestmanager
createProject	createContest
updateProject	updateContest
getProject	getContest
getProjects	getContest
getAllProjectTypes	getAllStudioFileTypes
getAllProjectCategories	getAllContestCategories
getAllProjectStatuses	getAllContestStatuses
getAllProjectPropertyTypes	N/A, just implement as a method to return an empty list.

1.3 Required Algorithms

The adaptation algorithm used is integral to this component, and should be described in detail to ensure that it is implemented correctly.

1.4 Example of the Software Usage

The Auto Pilot component will be used to handle phases of contests in the Client Cockpit application. It uses the Project Management component, and this component will provide the bridge between Project Management and the Client Cockpit.

1.5 Future Component Direction

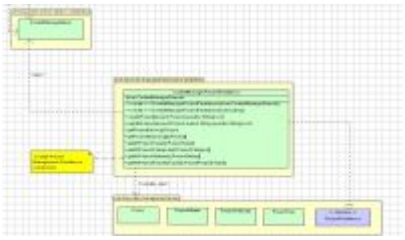
None.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces



2.1.3 Environment Requirements

- Development language: Java1.5
- Compile target: Java1.5 and Java1.6

2.1.4 Package Structure

com.topcoder.management.project.persistence

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The connection to the Contestmanager

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

JNDI

3.2.2 TopCoder Software Component Dependencies:

- Studio Contest Manager 1.0
- Contest and Submission Entities 1.0
- Project 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:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003
- Informix 10.0

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

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 Poseidon.