[TopCoder]

Project Services 1.0 Requirements Specification

1. Scope

1.1 Overview

The Project Services Component implements some of the business rules for combining Projects, Resources, Phases and Teams together.
The main interface of this component is fine grained enough to provide a useful API to client applications, but coarse grained enough to offer transactional atomic services and allow the presentation layer to minimize the calls to this layer.

1.2 Logic Requirements

1.2.1 Interfaces

This component must define and implement the interfaces depicted in the Component Interface Diagram inside the com.topcoder.project.service package. The diagram will be provided.

1.2.2 Find Active Projects

This service returns all the currently active projects along with all known associated information. The information is gathered from the following resources:

- Project Management
- Phase Management
- Project Phases
- Resource Management
- User Project Data Store
- Team Management

This service assembles the FullProjectData DTO for each project.

1.2.3 Find Projects

This service returns all the projects matching a given filter along with all known associated information. The information is assembled the same way as with the Find Active Projects service. This service allows the user to use the Filters created with the ProjectFilterUtility to restrict the results.

1.2.4 Find Active Project Headers

This service returns a summary of each of the currently active projects. The summary comprises only the data managed by the Project Management component.

1.2.5 Find Project Headers

This service returns a summary of each of the projects matching a given filter. The summary comprises only the data managed by the Project Management component.
This service allows the user to use the Filters created with the ProjectFilterUtility to restrict the results.

1.2.6 Retrieve Full Project Data

This service returns a FullProjectData object with all known associated information for a given project Id. Null in case the project doesn't exist.

1.2.7 FullProjectData class

This class is a DTO assembled to transfer all the data regarding a single project. It must include getters and setters for elements modeled as attributes. The instance variables must be private. At least an empty constructor must be provided. It also extends com.topcoder.project.phases.Project class.

1.2.8 Authentication

No authentication should be performed in this component. Authentication will be done in an upper layer.

1.2.9 Return values of Array type

For all methods/services returning arrays, in case there are no objects to fill the array it must return an empty one, never a null.

1.2.10 Access to the data model

This component uses existing lower level Entity Management components to execute the requests whenever necessary. It's required not to use the Data Access Objects directly since low level validations are performed in Entity Management layer.
The Entity Management layer comprises the following components:

- **Team Management**: Under development. See component interface diagram.
- **Resource Management**: Generic component. Available in the catalog.
- **Project Phases**: Generic component. Available in the catalog.
- **Project Management**: Generic component. Available in the catalog.
- **User Project Data Store**: Custom component. Provided

The actual implementation of each of these components must be pluggable.

1.2.11 Exception Management

None of the services throw checked exceptions. If a problem occurs or the component is misused, a runtime exception should be thrown.

1.2.12 Logging

This component must produce the appropriate logging information for tracing / debugging / production support.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

This component will be used to generate listings of projects, the phase each project is in, and the resources involved in them.

1.5 Future Component Direction

New services will be added to support more generic filters.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

2.1.2.1 Interfaces defined

This component must define and implement the interfaces depicted in the Component Interface Diagram inside the com.topcoder.project.service package.

2.1.2.2 Fields in CID interfaces

Some interfaces in the CID have member variables. The meaning of those members is that the actual Java interface will have getters and setters for those fields.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4

2.1.4 Package Structure,

com.topcoder.project.service – Main package

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The actual implementation to be used of the following components must be configurable (if applicable).
  Team Management
  Resource Management
  Project Phases
  Project Management

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

Configuration Manager
Team Management
Resource Management
Project Phases
Project Management
User Project Data Store
Logging Wrapper

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

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.

3.3.1 EJB compliant

This component will be used from EJB objects, thus it must comply with EJB development

restrictions (http://java.sun.com/blueprints/qanda/ejb_tier/restrictions.html).

3.3.2 Component Scalability

The component needs to be scalable. Running multiple instances in the same JVM or in multiple JVM's concurrently should not cause any problem.

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.