Software Documentation : Web Service Wrapper for Resource Management

This page last changed on Apr 20, 2008 by ghostar.

1. Scope

Web Service Wrapper for Resource Management Specification

1.1 Overview

This component provides an implementation of the RegistrationPersistence interface from the Registration Framework component. The implementation will use a web service defined in this component that wraps a ResourceManager instance from the Resource Management component. This way when a user registers using the new Registration Framework architecture, and this component is added as a registration persistence mechanism in the framework, the Resource Management component will also be alerted of the registration. The web service will be deployed in a JAX-WS environment.

1.1.1 Version

1.0

1.2 Logic Requirements

1.2.1 Adapter pattern

This component will adapt the entities from the Registration Framework component to work with the entities from Resource Management. The mapping will look something like this:

Resource Management	Registration Framework
Resource.ResourceRole.Name	ContestRole.Role.Name
Resource.ResourceRole.Id	ContestRole.Id
Resource.Project	Contest.Id

NOTE: The other values of the User entity, including the Handle, Email, and IsSuspended properties must be added into the Resource properties map.

1.2.2 Method mapping

The following table outlines the mapping between method calls to the RegistrationPersistence interface and method calls to the ResourceManager service.

ResourceManagerService	RegistrationPersistence
updateResource	register
removeResource	unregister

A general overview of the implementation of a register(contest, user, role) call in the persistence interface will look something like this. The difference in unregister would be that ResourceManagerService.removeResource is called instead of updateResource.

1. Create a new Resource instance
2. Set the project ID using the Contest ID
3. Call getAllResourceRoles from the ResourceManagerService
4. Loop through all the roles, finding the one that matches the name of the Role.Name property of the ContestRole provided
5. Set the Resource.ResourceRole property using the ResourceRole found.
6. Update the registration by calling ResourceManagerService.updateResource(...)

Note that if no matching ResourceRole is found, an exception will be thrown.

1.2.3 Webservice

This component must also provide a wrapper to the Resource Management component, exposing all ResourceManager functionality as webservices. This component must also provide a client class that will allow for easy remote access to the service methods.

1.2.4 EJB Wrapper

This component must provide an EJB wrapper between the web service and the provided Resource Manager that the calls delegate to.

1.2.5 EJB Security

All service methods will have a security role named "User" applied. The methods will use the caller principal name to apply the "operator" string of update calls to the Resource Manager.

1.2.5 EJB Transactions

Proper EJB transactions must be defined and used to ensure that the service can handle multiple, simultaneous calls successfuly

1.2.6 ResourceManager delegation

This component must define a service interface and implementation that delegates to a contained ResourceManager implementation.

1.2.7 Webservice Client

This component must define a client class that mimics the API of the service, to make it easy for a user to access the webservices remotely.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

This component will be used to provide a bridge between the Registration Framework and the Resource Manager component.

1.5 Future Component Direction

None.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Design must adhere to the API of the Registration Framework component.

2.1.3 Environment Requirements

- Development language: Java1.5
- Compile target: Java1.5 and Java1.6
- JAX-WS https://jax-ws.dev.java.net/

2.1.4 Package Structure

com.topcoder.registration.persistence

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The URL to the web service to use

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

- Registration Framework 1.0
- Resource Management 1.0.2
- Configuration API 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:

- Windows XP
- Windows 2003
- JBoss 4

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 TCUML.