[TopCoder]

Software Documentation : Java Custom DR Points Manager

This page last changed on Jun 21, 2008 by pulky.

1. Scope

1.1 Overview

This component provides operations on digital run points like add new points, get points, update points, search points; CRUD (Create, Read, Update, Delete) operations on points types; CRUD (Create, Read, Update, Delete) operations on points statuses; CRUD operations on points operations; CRUD operations on points reference types.
Component runs as a stateless EJB. It uses Hibernate JPA implementation to work with persistence. It is used by Digital Run Service component.
DDL, entities and mappings are provided by Digital Run Entities 1.0 component.
Manager interface is provided together with this specification. (DR Points Manager Interface Diagram.gif)

1.1.1 Version

1.0

1.2 Logic Requirements

Please be sure to consider section "3.3 - Design Constraints" for your design.

1.2.1 Provide operations on the points

- Add points.
- Update points.
- Remove points.
- Get points by id.
- Search points using filter

See interface diagram for method's API

1.2.2 Provide CRUD operations on the points types

- Add points type
- Update points type
- Remove points type
- Get points type using id
- Get all points types

See interface diagram for method's API

1.2.3 Provide CRUD operations on the points status

- Add points status
- Update points status
- Remove points status
- Get points status using id
- Get all points status

See interface diagram for method's API

1.2.4 Provide CRUD operations on the points operations

- Add points operation
- Update points operation
- Remove points operation
- Get points operation using id

- Get all points operations

See interface diagram for method's API

1.2.5 Provide CRUD operations on the points reference types

- Add points reference type.
- Update points reference type
- Remove points reference type
- Get points reference type using id
- Get all points reference types

See interface diagram for method's API

1.2.6 Database access

Hibernate is used to provide access to database in EJB. Note that Hibernate 3.2 is completely compatible with JPA so it should be used as a standard JPA provider.

1.2.7 EJB description

Stateless bean should have both remote and local interface. All methods should use REQUIRED transaction attribute.

1.2.8 Logging

All defined operations should be logged. The logging mechanism should be pluggable, e.g., it should be an option to disable logging.
Logging strategy:

- Entrance and exit of methods should be logged at the INFO level
- Exception should be logged at the ERROR level

1.2.9 Searching

The Search Points Using Filter operation should use the Search Builder component.
Currently Search Builder doesn't fully support hibernate but this will be taken care of in another competition. (Enhancement for Search Builder component). This enhancement will include a new SearchStrategy for hibernate that will be used by this component.

The following Filters need to be supported:

Search by	Required operations
Points id	Equals / In
Track id	Equals / In
Points status id	Equals / In
Points type id	Equals / In
Points reference id + reference id	Equals+Equals / Equals+In
Points description	Equals / Like
user id	Equals / In
amount	Equals / Greater or equal than / Lower or equal than

points application date	Equals / Greater or equal than / Lower or equal than
points award date	Equals / Greater or equal than / Lower or equal than
is potential	Equals

Helper classes/methods for building searches should be provided.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

This component will be used by Digital run Service and possibly by some other services.

1.5 Future Component Direction

Add auditory to CRUD operations. Auditor 2.0 and Auditor Hibernate Plug-in 1.0 will be used for this purpose.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

See included interface diagram. (DR Points Manager Interface Diagram.gif)

2.1.3 Environment Requirements

- Development language: Java 6.0
- Compile target: Java 6.0

2.1.4 Package Structure

com.topcoder.service.digitalrun.points

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

Logging should be pluggable

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

- EJB 3.0

- JPA 1.0
- Hibernate 3.2 and higher

3.2.2 TopCoder Software Component Dependencies:

- Digital Run Entities 1.0
- Base Exception 2.0
- Logging Wrapper 2.0
- Search Builder 1.3.2+ (an enhanced Search Builder - with hibernate compatibility - is being prepared)

**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:

- Informix Database 10.00.UC 5
- JBoss 4.2 GA
- Java Persistence API 1.0 (JPA)
- Hibernate 3.2.5

3.2.4 QA Environment:

- RedHat Linux 9
- JBoss 4.2 GA
- Java 1.5
- Informix 10.00.UC 5

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.

Design must adhere to the provided interface. The manager should not work with the persistence layer directly. DAO classes should be provided for each entity.

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 TC UML Tool.