[TopCoder]

Digital_Run_Entities 1.0 Component Specification

1. Design

This component defines entities to represent database information, providing also the O/R mapping. A DB diagram (DataModel.png) and DDL for creating the DB tables (DRSchema.ddl) are provided together with this specification.

1.1 Design Patterns

- Adapter Pattern is used by DigitalRunEntityIDGenerator to adapt IDGenerator to IdentifierGenerator

1.2 Industry Standards

- Hibernate

1.3 Required Algorithms

1.3.1 Compatibility and Testing

The component will be compatible with JBoss 4.2 GA and Informix Database 10.00.UC 5

with Hibernate 3.2.5 installed. The component test suite will include persistence tests that verify the correct operation of the component in the target environment.

1.4 Component Class Overview

BaseEntity
The BaseEntity entity.
It holds the command attributes id, creation date, modification date, and description for all the entities in this component.
It's mutable and not thread safe.

TrackType
The TrackType entity.
All the attributes of this entity are defined in its base entity.
It's mutable and not thread safe.

TrackContestType
The TrackContestType entity.
All the attributes of this entity are defined in its base entity.
It's mutable and not thread safe.

TrackContestResultCalculator
The TrackContestResultCalculator entity.
Plus the attributes defined in its base entity, it holds the attribute class name
It's mutable and not thread safe.

ProjectType
The ProjectType entity.
Plus the attributes defined in its base entity, it holds the attributes name, creation user, modification user.
It's mutable and not thread safe.

PointsCalculator
The PointsCalculator entity.
It holds the attribute class name
It's mutable and not thread safe.

DigitalRunPointsReferenceType
The DigitalRunPointsReferenceType entity.
All the attributes of this entity are defined in its base entity.
It's mutable and not thread safe.

DigitalRunPointsOperation
The DigitalRunPointsOperation entity.
All the attributes of this entity are defined in its base entity.
It's mutable and not thread safe.

DigitalRunPointsStatus
The DigitalRunPointsStatus entity.
All the attributes of this entity are defined in its base entity.
It's mutable and not thread safe.

DigitalRunPointsType
The DigitalRunPointsType entity.
All the attributes of this entity are defined in its base entity.
It's mutable and not thread safe.

TrackContest
The TrackContest entity.
Plus the attributes defined in its base entity, it holds the attributes track contest type, track contest result calculator, and track.
It's mutable and not thread safe.

Track
The Track entity.
Plus the attributes defined in its base entity, it holds the attributes track type, points calculator, track status, start date, end date, and project types, etc.
It's mutable and not thread safe.
The Track entity.
It holds the attribute track type, track status, start date, end date, project types, points calculator, digital run pointses, and track contests
It's mutable and not thread safe.

DigitalRunPoints
The DigitalRunPoints entity.
Plus the attributes defined in its base entity, it holds the attributes track, digital run points status, digital run points type, digital run points reference type, digital run points operation, user id, amount, application date, award date, reference id and track.
It's mutable and not thread safe.

TrackStatus
The TrackStatus entity.
All the attributes of this entity are defined in its base entity.
It's mutable and not thread safe.

DigitalRunEntityIDGenerator
DigitalRunEntityIDGenerator is used to generate the id for all the entities defined in this component(required by requirement 1.2.1)
This class implements the Configurable interface to allow application users to configure different id generator names for different entities.
Assume that we configure the id generator for DigitalRunPoints entity,
the id configuration element should be:<id
name="id"
column="dr_points_type_id"
type="long"><generator
class="com.topcoder.service.digitalrun.entity.idgenerator.DigitalRunEntityIDGenerator"><
param
name="id_generator_name">DigitalRunPoints_IDGenerator</param></generator></id>
This class is immutable and thread safe.

1.5 Component Exception Definitions

No custom exceptions are defined in this component. Also system exceptions (like IllegalArgumentException) are documented in the class diagram.

1.6 Thread Safety

JPA/Hibernate does not require entity classes to be thread safe, and those provided by this component indeed are not thread safe too as they are mutable classes.

2. Environment Requirements

2.1 Environment

- At minimum, Java 1.5 is required for compilation and executing test cases

2.2 TopCoder Software Components

- **ID Generator 3.0** DigitalRunEntityIDGenerator use IDGenerator to generate the long id.

2.3 Third Party Components

- Hibernate Entity Manager 3.2
- Jboss 4.2 with JPA support

3. Installation and Configuration

3.1 Package Name

- **com.topcoder.service.digitalrun.entity**
- **com.topcoder.service.digitalrun.entity.idgenerator**

3.2 Configuration Parameters

- None.

3.3 Dependencies Configuration

- The hibernate configuration has to be updated with proper database credentials. Also the IDGenerator requires the configuration of DBConnectionFactory to be updated with the proper database credentials.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

The component provides entities and configuration for use with the Java Persistence API. It is used in accordance with JPA, by obtaining an EntityManager and employing its services to create / load / store / delete entities.

4.3 Demo

4.3.1 The usage of classes as java beans

```
// create the DigitalRunPointsType instance
DigitalRunPointsType digitalRunPointsType = new
DigitalRunPointsType();
digitalRunPointsType.setDescription("points type");

// create the DigitalRunPointsStatus instance
DigitalRunPointsStatus digitalRunPointsStatus = new
DigitalRunPointsStatus();
digitalRunPointsStatus.setDescription("points statuses");

// create the DigitalRunPointsReferenceType instance
DigitalRunPointsReferenceType digitalRunPointsReferenceType = new
DigitalRunPointsReferenceType();
digitalRunPointsReferenceType.setDescription("points reference
type");

// create the DigitalRunPointsOperation instance
DigitalRunPointsOperation digitalRunPointsOperation = new
DigitalRunPointsOperation();
digitalRunPointsOperation.setDescription("points operation");

// create the DigitalRunPoints instance
DigitalRunPoints digitalRunPoints = new DigitalRunPoints();

// create track
Track track = new Track();
track.setDescription("description");
track.setStartDate(new Date());
track.setEndDate(new Date());
track.setCreationDate(new Date());
track.setModificationDate(new Date());

// set track
// assume that tracks entities were created
digitalRunPoints.setTrack(track);

// set digital run points status

digitalRunPoints.setDigitalRunPointsStatus(digitalRunPointsStatus
);

// set digital run points type
```

```
digitalRunPoints.setDigitalRunPointsType(digitalRunPointsType);

// set digital run points reference type

digitalRunPoints.setDigitalRunPointsReferenceType(digitalRunPoint
sReferenceType);

// set digital run points operation

digitalRunPoints.setDigitalRunPointsOperation(digitalRunPointsOpe
ration);

Date applicationDate = new Date();
Date awardDate = new Date();

// for the following userId, amount, applicationDate,
// awardDate and referenceId
// we assume that they were retrieved somewhere.
// set user id
digitalRunPoints.setUserId(101);

// set amount
digitalRunPoints.setAmount(101.20);

// set application date
digitalRunPoints.setApplicationDate(applicationDate);

// set award date
digitalRunPoints.setAwardDate(awardDate);

// set reference id
digitalRunPoints.setReferenceId(40);

// set potential
digitalRunPoints.setPotential(true);

// the usage of other entites are similiar
```

4.3.2 *The usage of classes as entities via a transaction-scoped JTA EntityManager*

```
// get the EntityManager
Ejb3Configuration cfg = new Ejb3Configuration();
EntityManagerFactory emf =
cfg.configure("hibernate.cfg.xml").buildEntityManagerFactory();
manager = emf.createEntityManager();

// get the EntityTransaction
EntityTransaction et = manager.getTransaction();
try {
      // begin the transaction
      et.begin();

      TrackType entity = new TrackType();
      entity.setDescription("description");

      // Persist the entity
```

```
            manager.persist(entity);

            // refresh to get the time stamps to the entity
            manager.refresh(entity);

            // read
            TrackType peristed = manager.find(TrackType.class,
            entity.getId());

            // update the entity
            peristed.setDescription("newdisc");

            manager.merge(peristed);

            // delete the entity
            manager.remove(peristed);
      } catch (PersistenceException e) {
            // if any errors occurs, rollback the transaction
            et.rollback();
            throw e;
      } finally {
            // finally close the EntityManager
            manager.close();
      }

    // NOTE: Manipulations on the other entities are quite
      // similar, so they are not shown.
```

5. **Future Enhancements**
    - None.