Persistence Framework

Overview

Documents the persistence framework.

Details

Persistence is handled by the JPA framework with a default Hibernate implementation.

The mapping tool includes a JPA-enabled implementation of the Domain Model objects that use annotations to define the connections, cascading, and other relationships between the various objects. For persistence interdependence, we have defined four categories of objects

- Terminology
- Mapping
- Workflow (including Feedback)
- Reporting

Where there are connections between these worlds (say a MapRecord refers to a Concept) the reference is by identifier field rather than by direct connection. In a sense, these are layers built on top of one another that do not require tight coupling. The downside of this is that it introduces potential referential integrity problems. The upside is that it allows flexible connections to exist and be resolved at a later time.

Some principles:

- Eager fetching is rarely used
- Cascading is also rarely used, only for when objects are very tightly bound (like Concepts and Descriptions or MapRecords and MapEntries).
- Objects are always doubly-linked (e.g. MapRecord accesses MapEntries which each know their MapRecord).
- The mapped superclass join strategy is always used - there are no cases of multiple object types within the same table.

Configuration

The JPA configuration is in the standard config.properties file.

Property	Default Value	
hibernate.dialect	org.hibernate.dialect. MySQLDialect	The default is MySQL though simply changing this should support other hibernate-supported environments.
javax.persistence.jdbc.driver	com.mysql.jdbc.Driver	JDBC driver, this requires the MySQL connector to be in the classpath
javax.persistence.jdbc.url	jdbc:mysql://127.0.0.1:3306 /mappingservicedb	Default connection URL to a "mappingservicedb" database
javax.persistence.jdbc.user	n/a	MySQL user
javax.persistence.jdbc. password	n/a	MySQL user's password
hibernate.show_sql	false	Useful debug setting, change to "true" to see all queries executed by JPA layer.
hibernate.format_sql	true	Formats SQL when showing queries
hibernate. use_sql_comments	true	Add comments when showing SQL to explain what is happening.
hibernate.jdbc.batch_size	500	Batch size for bulk operations.
hibernate.jdbc. default_batch_fetch_size	500	Batch size for fetch operations.

Annotations Used

Annotation	Explanation

@Column	Define columns, column names, and specifications about size and nullability.
@ElementCollection, @CollectionTable	Used to define collections of non JPA objects (like a set of String).
@Entity	Used to define JPA-tracked objects.
@Enumerated	Used to indicate fields that have enumerated values.
@id @GeneratedValue	Used to managed identifier fields and ID strategy.
@ManyToOne, @OneToOne, @OneToMany	Used to define object relationships between different @Entity annotated classes.
@MappedSuperclass	Used for abstract superclasses to indicate their fields should be included in persistence of concrete subclasses.
@Table	Used to indicate table names for objects. Default underscore-based naming convention is used.
@Temporal	Used for date fields.
@Transient	Used to avoid persistence of fields, these are typically used for DTO fields as we reuse the objects for data transfer.
@UniqueConstraint	Used to index columns that would otherwise not have indexes. Not actually used for uniqueness.

References/Links

- http://en.wikipedia.org/wiki/Java_Persistence_API