Cheat Sheet

New Features in JPA 2.1

www.thoughts-on-java.org

Hi,

I am Thorben, the author of thoughts-on-java.org, and I want to thank you for signing up and downloading the “What's new in JPA 2.1” cheat sheet!

This cheat sheet gives you an overview about the 12 new features and enhancements introduced in JPA 2.1:

- Attribute Converter,
- Named Stored Procedure Query,
- Stored Procedure Query,
- Criteria API Bulk Operations,
- Constructor Result Mapping,
- Unsynchronized Persistence Context,
- Named Entity Graph,
- Entity Graph,
- Generating DB Schema,
- Programmatic Named Queries,
- CDI-Support in Entity Listener and
- JPQL Enhancements.

Each feature comes with a short description and some code snippets to bring you all the information you need for your daily work.

If you like to get in touch, write me an email at thorben@thoughts-on-java.org or tweet me at @thjanssen123.

Take care,

Thorben

Attribute Converter

Define a custom mapping between an attribute of an entity and a database column.

Define the Converter with `autoApply=true` to use it for all entity attributes of this type:

```
@Converter(autoApply=true)
public class ColorConverter
                      implements AttributeConverter {
    ...
}
```

If the Converter was defined with `autoApply=false`, its usage has to be annotated on each attribute:

```
@Entity
public class RectangleEntity {
    @Column
    @Convert(converter = ColorConverter.class)
    private Color color;
    ...
}
```

Read more: http://bit.ly/1EmYzOy

Named Stored Procedure Query

`@NamedStoredProcedureQuery` can be used to define stored procedure calls via annotations.

```
@NamedStoredProcedureQuery(
    name = "queryName",
    resultClasses = MyResult.class,
    procedureName = "My_Procedure",
    parameters = {
        @StoredProcedureParameter(
            mode=IN,
            name="paramName",
            type=Integer.class)
    }
)
```

Stored Procedure Query

The `EntityManager` was extended by a method to call stored procedures.

```
EntityManager.createStoredProcedureQuery(
     String procedureName, Class... resultClasses)
```

Criteria API Bulk Operations

`CriteriaUpdate` and `CriteriaDelete` were added to the Criteria API to support bulk operations.

```
CriteriaUpdate<Order> update =
             cb.createCriteriaUpdate(Order.class);
CriteriaDelete<Order> delete =
             cb.createCriteriaDelete(Order.class);
```

Read more: http://bit.ly/1AwX55x

Constructor Result Mapping

The `@ConstrutorResult` annotation defines a named mapping from a query result to a constructor call.

```
@SqlResultSetMapping(
     name = "resultMapping",
     classes = {
          @ConstructorResult(
               targetClass =
               org.thoughts.on.java.MyResult.class,
               columns={
                    @ColumnResult(name="id"),
                    @ColumnResult(name="name")
               }
          )
     }
)
```

Read more: http://bit.ly/1NwQtaO

Unsynchronized Persistence Context

JPA supports programmatic synchronization of the persistence context with the transaction.

Get an unsynchronized persistence context:
```
@PersistenceContext(synchronization =
                    SynchronizationType.UNSYNCHRONIZED)
```

Tell the `EntityManager` to join the persistence context with the transaction:
```
EntityManager.joinTransaction()
```

Named Entity Graph

Use `@NamedEntityGraph` to define a graph of entities that shall be loaded from the database.

Define the graph:
```
@NamedEntityGraph(name = "graph.Order.items",
     attributeNodes = {
          @NamedAttributeNode("address"),
          @NamedAttributeNode(value = "items",
                                   subgraph = "items")
          },
     subgraphs = @NamedSubgraph(name = "items"
          attributeNodes =
                    @NamedAttributeNode("product"))
     )
```

Provide the graph as a hint to the `EntityManager`:
```
EntityGraph graph =
     this.em.getEntityGraph("graph.Order.items");
Map hints = new HashMap();
hints.put("javax.persistence.fetchgraph", graph);
this.em.find(Order.class, orderId, hints);
```

Read more: http://bit.ly/1DKhuVz

Entity Graph

A graph of entities that shall be loaded from the database can be defined via a Java API.

Define the graph:

```
EntityGraph<Order> graph =
    this.em.createEntityGraph(Order.class);
graph.addAttributeNodes("address");
Subgraph<OrderItem> itemGraph =
    graph.addSubgraph("items");
itemGraph.addAttributeNodes("product");
```

Provide the graph as a hint to the `EntityManager`:

```
Map hints = new HashMap();
hints.put("javax.persistence.fetchgraph", graph);
this.em.find(Order.class, orderId, hints);
```

Read more: http://bit.ly/1Ag6NdR

Programmatic Named Queries

The `EntityManager` now provides a method to define named queries at runtime.

```
EntityManager.addNamedQuery(String name,
                                               Query query);
```

CDI-Support in Entity Listener

If used inside a Java EE container, Entity Listener can use CDI to inject beans and implement `@PreDestroy` and `@PostConstruct` methods

New Features in JPA 2.1

Generating DB Schema

There is a long list of new parameters in the `persistence.xml` to define the database setup.

```
javax.persistence.schema-generation.database.action
javax.persistence.schema-generation.scripts.action
javax.persistence.schema-generation.create-source
javax.persistence.schema-generation.drop-source
javax.persistence.schema-generation.create-database-
schemas
javax.persistence.schema-generation.scripts.create-
target
javax.persistence.schema-generation.scripts.drop-
target
javax.persistence.database-product-name
javax.persistence.database-major-version
javax.persistence.database-minor-version
javax.persistence.schema-generation.create-script-
source
javax.persistence.schema-generation.drop-script-
source
javax.persistence.schema-generation.connection
javax.persistence.sql-load-script-source
```

JPQL Enhancements

There were several enhancements to JPQL. You can now:

- define additional join parameters with `ON`,
- call database functions with `FUNCTION` and
- downcast entities with `TREAT`.