157
Shares

2.1 - 12 features every developer should know

90 ry 17, 2015 by Thorben Janssen – 3 Comments

36 29 36 3

10 are a regular reader of this blog, you know that I wrote several articles about features and cements introduced with JPA 2.1. One thing that was missing, was a general overview
3 all the changes. So here it is 😃
llowing paragraphs provide a description of the 12 features and enhancements introduced PA 2.1. And as a special bonus, I created a cheat sheet with a short description and onal code snippets for each change, which you can download for free.

Features and Enhancements in JPA 2.1

Named Stored Procedure Query

Sometimes it is easier or more efficient to use a stored procedure to perform the operations within the database. Before JPA 2.1 the only way to call a stored procedure was to use a native query. The newly introduced *@NamedStoredProcedureQuery* can now be used to annotate a query to call the stored procedure.

Stored Procedure Query

The Stored Procedure Query is an alternative way to implement a stored procedure call without using annotations. For this purpose the *EntityManager* was extended by the *createStoredProcedureQuery(String procedureName, Class… resultClasses)* method.

Attribute Converter

Attribute Converter provide a nice and easy way to define a custom mapping between your property on the entity and the database column. The only thing that is needed is a class that implements the *AttributeConverter* interface and is annotated with *@Converter*. You can find a more detailed introduction to Attribute Converter in JPA 2.1 – How to implement an Attribute Converter.
One of the most obvious ways to use an Attribute Converter is to implement a custom type mapping to persist a not supported data type or to change an existing default mapping, as it was done in JPA 2.1 Attribute Converter – The better way to persist enums. Or you could keep the type and change the stored value to implement some business requirements like encryption: How to use a JPA Attribute Converter to encrypt your data.

Constructor Result Mapping

The *@ConstructorResult* annotation is a handy addition to the already existing *@SqlResultSetMapping* and can be used to map the result of a query to a constructor call.

Programmatic Named Queries

Before JPA 2.1 the *@NamedQuery* annotation was the only way to define named queries. A programmatic creation was not supported. This was changed with JPA 2.1.
The *EntityManager* now provides the *addNamedQuery(String name, Query query)* method to do this.

Named Entity Graph

Lazy loading of relations between entities is a common pattern to load only the required information from the database and to improve the performance of the application. While this is a great feature as long as the related entities are not required, it creates additional load when the relations need to be initialized. There are multiple ways to initialize these lazy relations and using Named Entity Graphs is one of the better ones.
The annotations *@NamedEntityGraph*, *@NamedAttributeNode* and *@NamedSubGraph* allow us

to define a graph of entities that will be loaded from the database. You can find a more detailed description on how to do this in JPA 2.1 Entity Graph – Part 1: Named entity graphs.

Entity Graph

Entity Graphs are the second option introduced with JPA 2.1 to define a graph of entities that shall be loaded from the database and their usage is similar to Named Entity Graphs. The only difference is that Entity Graphs are defined via a Java API and not via annotations. Therefore the *EntityManager* was extended by the *createEntityGraph(Class rootType)* method. This is explained in more detail in JPA 2.1 Entity Graph – Part 2: Define lazy/eager loading at runtime.

JPQL Enhancements

There were several enhancements to the JPQL which can come in handy. You can now use the keyword *ON* to define additional join parameters, call database functions by using *FUNCTION* and downcast entities with *TREAT*.

Criteria API Bulk Operations

Up to JPA 2.1 the Criteria API did not provide any support for update or delete operations. The only options available were to perform the update on an entity or to write a native query to update multiple records at once. As described in Criteria Update/Delete – The easy way to implement bulk operations with JPA2.1, the Criteria API was extended with *CriteriaUpdate* and *CriteriaDelete* to also support bulk write operations.

Unsynchronized Persistence Context

Using a synchronized persistence context to propagate every change to the database is the default approach in JPA. If you need more control about the database propagation, you can now use the unsynchronized persistence context. Therefore you need to provide the synchronization mode to the injection with *@PersistenceContext(synchronization=SynchronizationType.UNSYNCHRONIZED).* You then need to call *EntityManager.joinTransaction()* manually to synchronize the changes.

Generating DB Schema

Up to JPA 2.1 you needed to use vendor specific configuration parameter to define the database setup in the *persistence.xml* file. Starting from version 2.1 there is also a standard way to do this. Therefore the specification defines the following long list of parameters:

- 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

CDI-Support in Entity Listener

The integration with CDI was improved with JPA 2.1. You can now use CDI to inject beans into *EntityListener*s and to implement the *@PreDestroy* and *@PostConstruct* methods.

To sum it up

From my point of view, JPA 2.1 – despite being just a minor release – introduced some great improvements to the specification. I especially like the entity graphs and the stored procedure queries but also the AttributeConverter can be quite handy. Which features do you like best? Sometimes it is difficult to remember all the implementation details, if you need them. This is where I personally like to have a cheat sheet that provides all the information I need, like this one 😉

g+ 90 f 29 36 in 3

Filed Under: JavaEE7, JPA, JPA2.1

ENJOYED THIS POST?

Comments

Chris Pratt says
February 20, 2015 at 7:31 pm

I anyone aware of a lightweight JPA 2.1 implementation that is also compatible with Java 8? I tried OpenJPA, but it doesn't appear to support Java 8.

Reply

Anonymous says
March 30, 2015 at 6:05 pm

EclipseLinks works fine with Java 8, though it did not support lambda in entities and stream on collections before 2.6.0.

Reply

Thorben Janssen says
March 31, 2015 at 7:43 pm

What do you mean with “compatible with Java 8″?
EclipseLink and Hibernate work fine with Java 8. But the new types like LocalDate are not supported by JPA because Java 8 was released after JPA 2.1. I am not aware of any JPA implementation that supports these types at the moment.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Comment

POST COMMENT

SEARCH

Search this website…

ABOUT ME

Hi,

my name is Thorben Janssen and I write about Java EE related topics.

I am a member of the CDI 2.0 (JSR 365) expert group and have more than 10 years of experience in Java EE development. Currently, I live and work as a senior developer and architect in Dresden, Germany.

FIND ME AT

Tweets Follow

Thorben Janssen @thjanssen123 9h
Comment Your Fucking Code! by @nipafx

blog.codefx.org/techniques/doc...
Show Summary

Baeldung @baeldung 14 Jul
Really strong week: baeldung.com/2015-week-revi... with pieces by @unclebobmartin, @jetbrains, @loggly, @springcentral, @awscloud and @aphyr
Retweeted by Thorben Janssen
Show Summary

Abhishek Gupta @abhi_tweeter 14 Jul

BLOG ARCHIVE

- July 2015 (1)
- June 2015 (5)
- May 2015 (3)
- April 2015 (9)
- March 2015 (7)
- February 2015 (5)
- January 2015 (1)
- December 2014 (6)
- November 2014 (2)
- October 2014 (4)
- September 2014 (5)
- August 2014 (3)
- July 2014 (3)
- June 2014 (5)
- April 2014 (2)
- March 2014 (2)
- January 2014 (1)

October 2013 (7)

September 2013 (4)