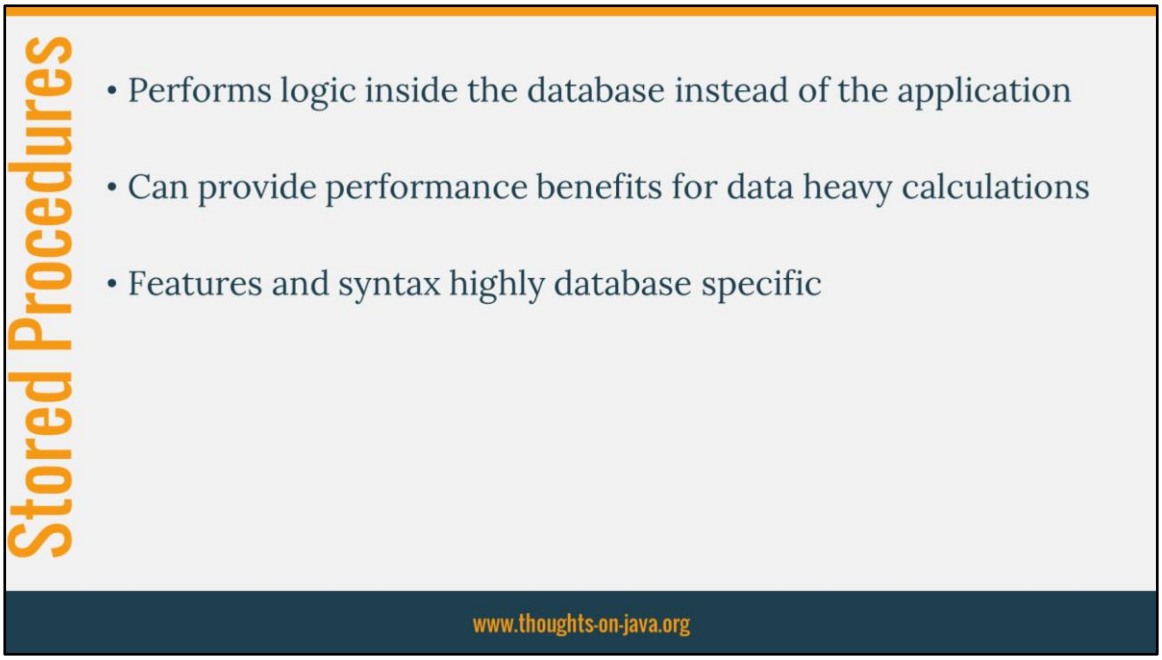
Stored Procedures

www.thoughts-on-java.org

Let each system do the work it is most capable of is always a good way get a good performance for you application. And the database can handle huge amounts of data way better than your Java business tier. One option to let the database perform some logic are functions which I explained in the previous module. In this module, I want to show you how to call stored procedures which can be used to implement more complex logic. But I will not get into details about how to implement a stored procedure in the database.

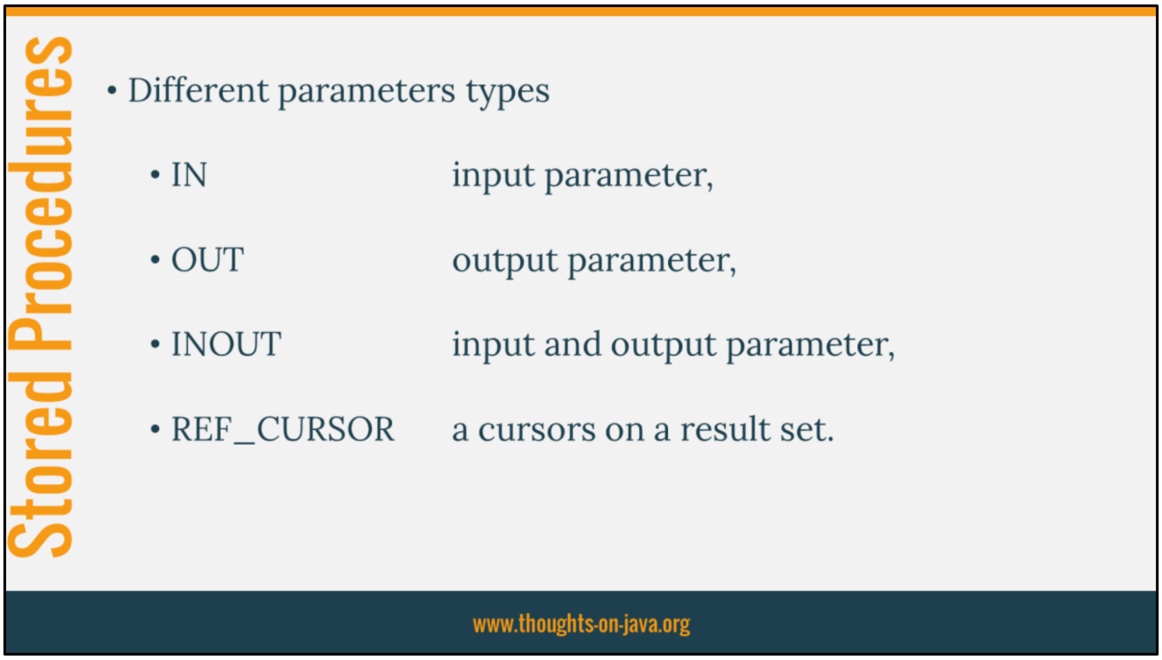


Stored Procedures

- Performs logic inside the database instead of the application
- Can provide performance benefits for data heavy calculations
- Features and syntax highly database specific

www.thoughts-on-java.org

The huge benefit of using stored procedures is that you don't have to transfer any data from the database to the logic that uses it. This can provide huge performance improvements for very data heavy operations. But please, don't move too much data into the database. As you know, there are a lot of good reasons why we implement the business tier in Java and not in the database. Using stored procedures should be the exception.
And please make sure that you don't have to support multiple different database systems because stored procedures are highly database specific.



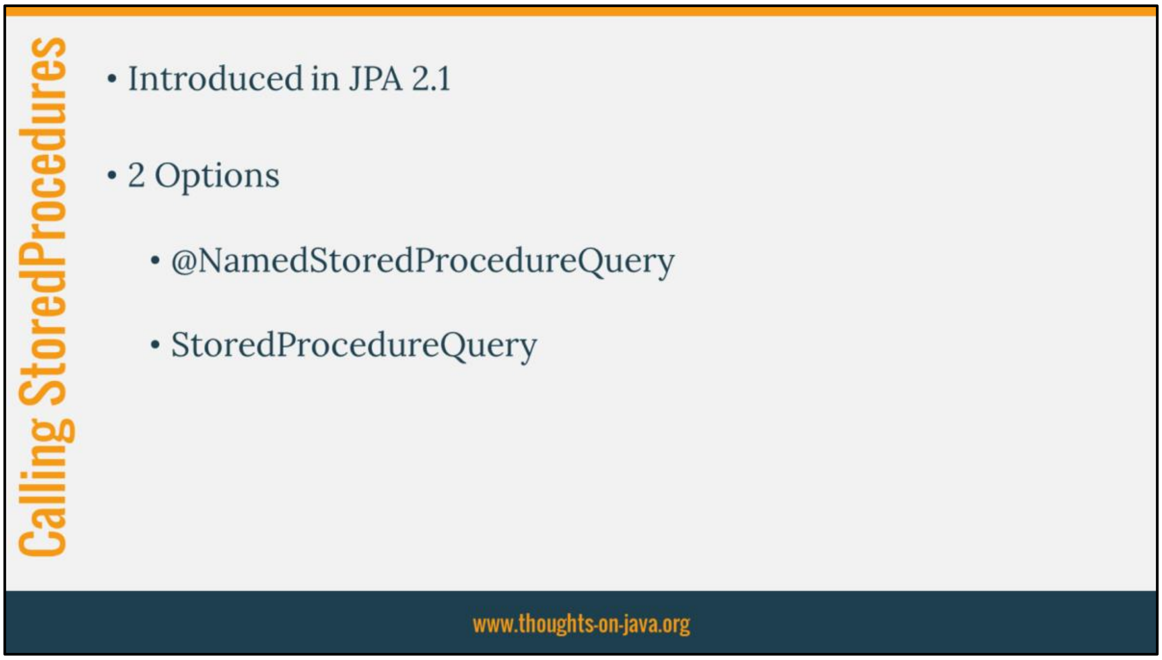
Stored Procedures

- Different parameters types

• IN	input parameter,
• OUT	output parameter,
• INOUT	input and output parameter,
• REF_CURSOR	a cursors on a result set.

www.thoughts-on-java.org

You can use 4 different types of parameters. The IN and OUT parameters are input and output parameters as you know from Java. The INOUT is a combination of these two and the REF_CURSOR is a cursor on a result set which you can for example use to return the result of a query.

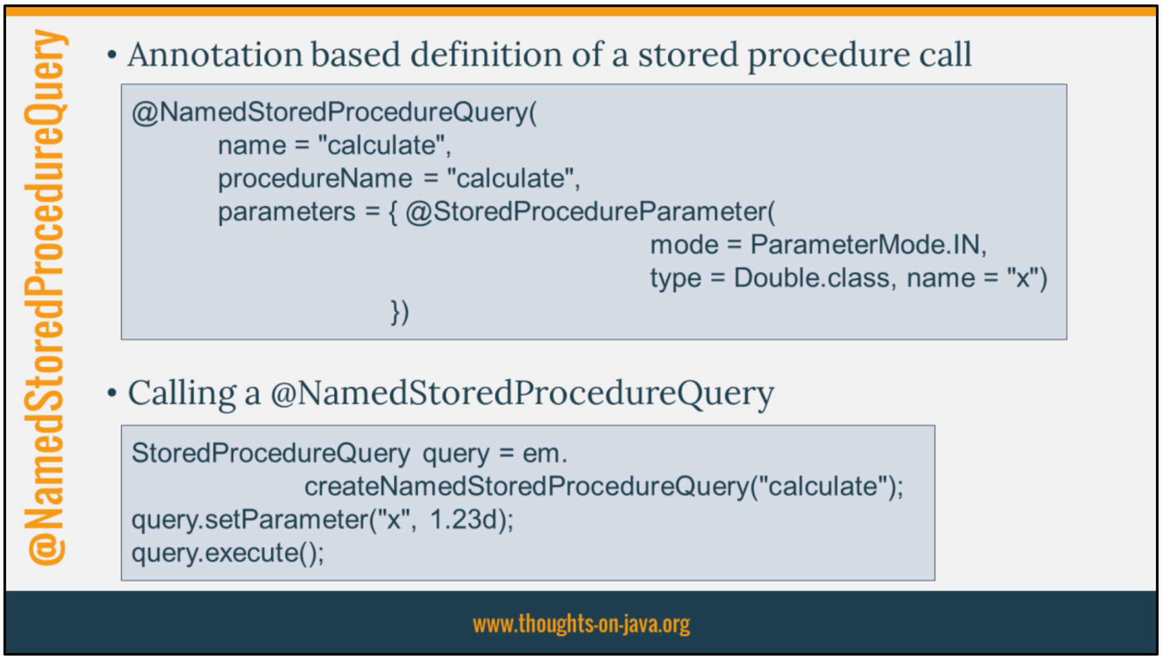


Calling StoredProcedures

- Introduced in JPA 2.1
- 2 Options
  - @NamedStoredProcedureQuery
  - StoredProcedureQuery

www.thoughts-on-java.org

Real support for stored procedures was added in JPA 2.1 with the @NamedStoredProcedureQuery and the StoredProcedureQuery. Prior to that, you had to use native queries to call a stored procedure.


@NamedStoredProcedureQuery

- Annotation based definition of a stored procedure call

```
@NamedStoredProcedureQuery(
        name = "calculate",
        procedureName = "calculate",
        parameters = { @StoredProcedureParameter(
                                        mode = ParameterMode.IN,
                                        type = Double.class, name = "x")
                })
```

- Calling a @NamedStoredProcedureQuery

```
StoredProcedureQuery query = em.
            createNamedStoredProcedureQuery("calculate");
query.setParameter("x", 1.23d);
query.execute();
```

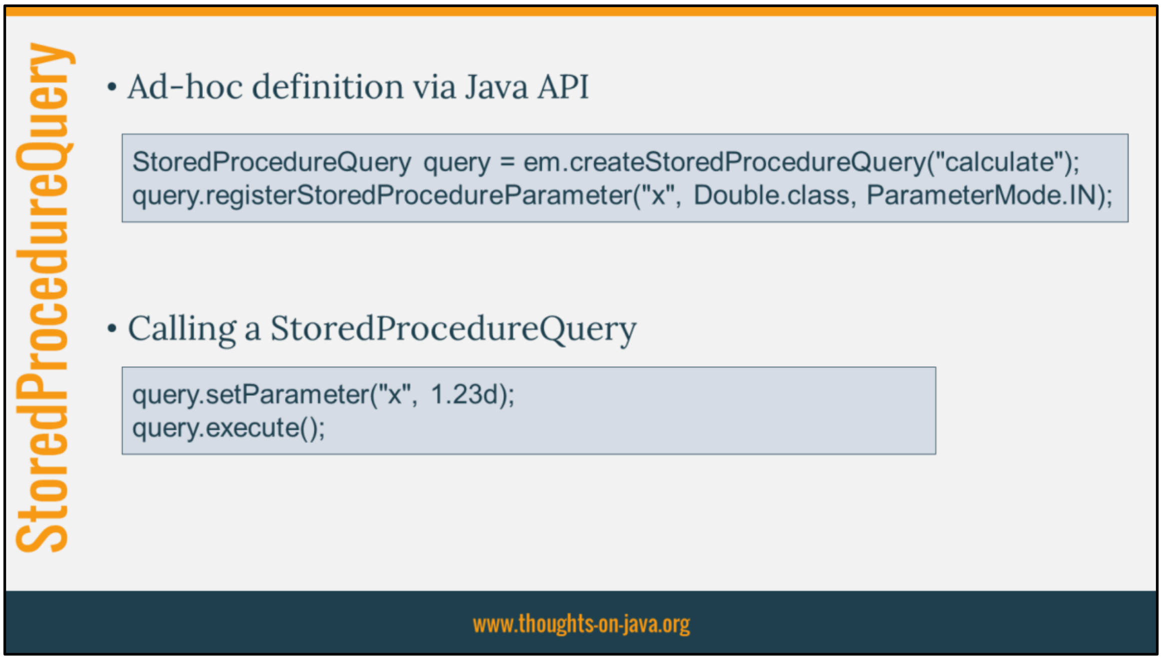
www.thoughts-on-java.org

The @NamedStoredProcedureQuery annotation allows you to define the stored procedure call via annotations and reference it by name, if you want to call it. As you can see in the first code snippet, the definition of a *@NamedStoredProcedureQuery* is quite simple. You just need to define the name of the query, the name of the stored procedure in the database and the parameters of the stored procedure. The name of the query is required by the *EntityManager* to create the *@NamedStoredProcedureQuery*. So better choose something that's easy to understand and remember. In this case, the @NamedStoredProcedureQuery is called calculate which is also the name of the stored procedure in the database and it gets only 1 input parameter called x.
The second code snippet shows how you can call the @NamedStoredProcedureQuery. You just need to provide the name of it to the createNamedStoredProcedureQuery method and set the values of the input parameters. As soon as you have set all input parameters, you can call the execute method to execute the stored procedure call.

OK, let's try it out and see what happens.

Demo

www.thoughts-on-java.org


StoredProcedureQuery

- Ad-hoc definition via Java API

```
StoredProcedureQuery query = em.createStoredProcedureQuery("calculate");
query.registerStoredProcedureParameter("x", Double.class, ParameterMode.IN);
```

- Calling a StoredProcedureQuery

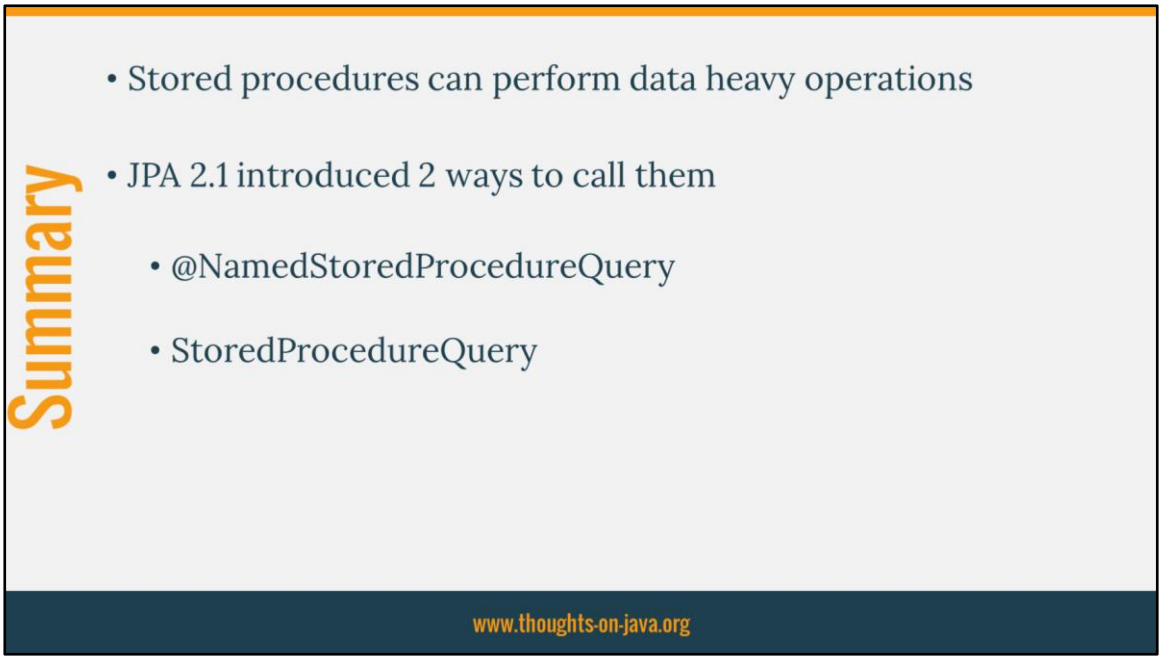
```
query.setParameter("x", 1.23d);
query.execute();
```

www.thoughts-on-java.org

The StoredProcedureQuery is very similar to the @NamedStoredProcedureQuery. The main difference is, that it uses a Java API instead of annotations. This allows you to create ad-hoc definitions for stored procedure calls.
As you can see in the first code snippet, the definition of a StoredProcedureQuery uses the same concepts as the annotation based definition of a @NamedStoredProcedureQuery. You have to provide the name of the stored procedure to the createStoredProcedureQuery method of the EntityManager and register the parameters of the stored procedure afterwards.
The second code snippet shows how you can call a StoredProcedureQuery. You use the StoredProcedureQuery which you previously created, set the values for the input parameters and call the execute method.

Demo

www.thoughts-on-java.org



Summary

- Stored procedures can perform data heavy operations
- JPA 2.1 introduced 2 ways to call them
  - @NamedStoredProcedureQuery
  - StoredProcedureQuery

www.thoughts-on-java.org

As I said in the beginning of this module, performing very data heavy operations with a stored procedure can be faster than transferring all the required information into the business tier and process them there. Since JPA 2.1, you can use a @NamedStoredProcedureQuery and a StoredProcedureQuery to call a stored procedure in the database. The @NamedStoredProcedureQuery defines the stored procedure call via annotations and you can then reference its name to create the query. The StoredProcedureQuery uses a Java API to define the stored procedure call and can be used to a query ad-hoc.