# Facets

Facets allow the developer to produce partial code re-generation.

When the project is small—containing a few tables and views—it doesn't make much of a difference to generate all the DAOs of the whole project every time a configuration change is added.

However, when the project grows—specially when the configuration includes expensive DAOs—it can be much faster to re-generate only the specific section that changes instead of the full list of DAOs.

It's recommended that a full re-generation is done periodically, nevertheless (maybe once a day), to avoid cases where a new configuration change fell through the crack and was never regenerated by mistake. One member of the team can be in charge of this, or even a continuous integration server can be set up to do this every night.

## Regeneration Slowness

The need to generate only a subset of database objects is based on the need to keep the development speed up and agile. You want to keep you development speed high, don't you? According to our experience it turns out there are three main issues that slow down the HotRod re-generation.

**Great Number of DAOs**

One main unavoidable source of slowness, is that some databases are slow to provide database objects metadata. The JDBC metadata interface is an optional part of the JDBC drivers' implementation, and is somewhat disregarded by the database vendors. On some database the metadata extraction is fast and efficient, but in other ones is... well it leaves room for improvement.

If your project has massive number of tables, views, and selects, then the HotRod generation can become slow simply because the database takes minutes to respond instead of (milli)seconds.

**Expensive DAOs**

Another source of slowness, can be the complexity of the <select> tags defined in the configuration file. Some databases are slow to create and drop complex views. This is particularly important when your project has massive number of <select> statements. If this is the case, every entire code generation will refresh these SQL select statements by creating and dropping complex views on the database.

**Slow Database**

Sometimes you are just unlucky with the database server. Maybe it's overwhelmed. Maybe it's a small machine. Maybe the network connection is slow, specially if you need to connect to it over a VPN or network tunnel. Usually there's not much you can do about it.

In the end, it doesn't really matter why the code generation becomes slow. The important thing here is that generating small subsets of the persistence layer instead of its entirety becomes much more efficient. And this is exactly what facets do.

## Multiple Facets

When generation the persistence layer the developer can specify one or more facets to be generated. This way facets may be additionally subdivided to suit different development needs.

On the other hand, if no facet is specified, then HotRod generates the whole configuration file.

## Facets are Merged

Facets act more like a mark on the configuration tag than as a section of it. That is the same facet may show up multiple times in the configuration file or any of its included fragments. All the facet tags are combined into a single one that performs the partial code generation.

## Example

The facets selection is not part of the configuration file but it's specified when running HotRod. The configuration file defines the facets, and it's the HotRod execution that chooses the ones to be generated.

The excerpt below, obtained from the Ant target that executes HotRod shows how to decide which facets are to be generated:

<hotrod driverclass="..."

url="..."

username="..."

password="..."

catalog="..."

schema="..."

generator="..."

facets="employee, version-5.1"

configfile="..."

display="..." />

The list of selected facets is separated by commas. If no facet is specified—either by an absent facet attribute or because its value is blank or empty—then the whole configuration is included and generated.

The example shown below includes a configuration file that includes another fragment. The main configuration file is:

<?xml version="1.0"?>

<!DOCTYPE hotrod SYSTEM "hotrod.dtd">

<hotrod>

<generators>

<mybatis>

...

</mybatis>

</generators>

<facet name="employee">

<table name="employee" />

</facet>

<table name="branch" />

<fragment file="accounting/purchases.xml" />

<facet name="employee">

<view name="department" />

<select java-class-name="ExecutiveEmployee">

<![CDATA[

select \* from employee where is\_executive = 1

]]>

</select>

</facet>

</hotrod>

The included purchases.xml fragment file is:

<?xml version="1.0"?>

<!DOCTYPE hotrod-fragment SYSTEM "hotrod-fragment.dtd">

<hotrod-fragment>

<facet name="orders-management">

<table name="order" />

<select java-class-name="ExpensiveOrder">

<![CDATA[

select \* from order where price &gt; 1000.00

]]>

</select>

</facet>

<table name="gift\_card" />

<select java-class-name="MonthlyCoupon">

<![CDATA[

select \* from coupon where type = 'MONTHLY'

]]>

</select>

<facet name="employee">

<table name="employee\_order" />

</facet

</hotrod>

These two fragments define several tables and three SQL selects. There are a few things to notice here:

* There are two facets: employee, and order-management.
* One facet (employee) shows up multiple times in the same file (the main configuration file), the other one (order-management) shows up in a single file (purchases.xml).
* One facet (employee) shows up in multiple files. It shows up in both of them.
* Some tables and SQL selects do not belong to any facet: table branch, and select MonthlyCoupon. These are only generated when the whole configuration is selected, i.e. no facets are specified.

Essentially you can specify the facets in any combination needed by the development team. It's the HotRod execution that selects which ones to use.