Supporting different OpenMRS versions

In OpenMRS 1.10, 1.9.8, 1.8.5, 1.7.5 and later we introduced new ways to make it easier to develop modules supporting OpenMRS versions with incompatible API changes.

Let's consider changes in DrugOrder introduced in OpenMRS 1.10. In OpenMRS 1.9 `DrugOrder.getFrequency()` returns a string, whereas in OpenMRS 1.10 it returns a newly introduced `OrderFrequency` object. It is a backwards incompatible change, which you need to consider writing a module that supports OpenMRS 1.10 and 1.9.

One way to approach such incompatible changes is to use Java Reflection API to invoke methods found with `java.lang.Class.getMethod()`. It's considerably easy to do when all that changed is a method name, but gets quite complex when you have to deal with a newly introduced class.

The other approach that we suggest now is to use conditionally loaded beans or resources. We will explain how to use conditionally loaded beans first.

Loading beans conditionally

We will continue with the `DrugOrder` example. You start from creating a maven submodule for a version of OpenMRS that has an incompatible change in API. In that submodule you need to set openmrs-api version to 1.10 in order to be able to use the `OrderFrequency` class. Our suggested convention for such a submodule artifact id is yourmodule-api-1.10. Configure it to depend on yourmodule-api as well.

Next you have to refactor code that do any calls to `DrugOrder.getFrequency()` by extracting it to a separate class e.g. `DrugOrderCompatibility1_9` which you will annotate with

```
@OpenmrsProfile(openmrsVersion = "1.9.8 - 1.9.*")
```

`@OpenmrsProfile` instructs Spring to create a bean only when running OpenMRS in version from 1.9.8 (including) to 1.10 (excluding). Create an interface for that class e.g. `DrugOrderCompatibility` and use it with `@Autowired` in all places that used to call `DrugOrder.getFrequency()`.

Create a similar class e.g. `DrugOrderCompatibility1_10` implementing `DrugOrderCompatibility` in the 1.10 maven submodule. Annotate it with `@OpenmrsProfile(openmrsVersion = "1.10")`, which will instruct Spring to create such a bean only when running OpenMRS 1.10 and later.

That's it! From now on if you install a module on OpenMRS 1.9.8 Spring will inject `DrugOrderComaptibility1_9`, whereas on OpenMRS 1.10 it will inject `DrugOrderCompatibility1_10`.

Note that `@OpenmrsProfile` can be used even before OpenMRS 1.9.8, 1.8.5, 1.7.5 (before the feature was added). This annotation will be simply ignored in older versions of OpenMRS and Spring will not create beans for classes annotated with just `@OpenmrsProfile`. In order for Spring to load a bean on versions of OpenMRS, which do not support conditional loading of beans you just need to annotate a class with **both `@OpenmrsProfile` and `@Component`**. See here in htmlformentry's `DrugOrderTagHandlerSupport1_6`.

In this example on OpenMRS 1.6.5 the `@OpenmrsProfile` annotation will be completely ignored, but `@Component` will not so the bean will be created.
For any classes that need to be loaded conditionally you use **`@OpenmrsProfile` only**. See here in htmlformentry's `DrugOrderTagHandlerSupport1_10`.
In this example on OpenMRS 1.6.5 the `@OpenmrsProfile` annotation will be completely ignored and the bean will not be created.
Using this approach we made the htmlformentry module work on OpenMRS versions that don't have conditional loading of resources

For a full example see: https://tickets.openmrs.org/browse/HTML-515

Loading resources conditionally

Alternately or in addition to the above you can instruct the OpenMRS module loader to skip loading resources based on the OpenMRS version. After you create a maven submodule for a specific OpenMRS version, maven packages it in a jar file and puts in your omod in the lib directory. In config.xml in your module you can declare:

```
<conditionalResources>
        <conditionalResource>
                <path>/lib/yourmodule-api-1.10.*</path>
                <openmrsVersion>1.10</openmrsVersion>
        </conditionalResource>
</conditionalResources>
```

, which will instruct OpenMRS to load yourmodule-api-1.10.* only when OpenMRS version is 1.10 or above.

This is also a handy feature if your module uses a library that you ship with your module, but it is now included in OpenMRS core. You want to load such a library only on OpenMRS versions that don't provide it.

Conditional loading based on running modules

It is also possible to load beans/resources based on running modules. For instance below is the annotation to add to load a bean only if the metadatasharing and the metadatamapping modules in version 1.x are running:

```
@OpenmrsProfile(modules = {"metadatasharing:1.*", "metadatamapping:1.*"})
```

Similarly use

```
<conditionalResources>
        <conditionalResource>
                <path>/lib/yourmodule-api-1.10.*</path>
                <modules>
                        <module>
                                <moduleId>metadatasharing</moduleId>
                                <version>1.*</version>
                        </module>
                        <module>
                                <moduleId>metadatamapping</moduleId>
                                <version>1.*</version>
                        </module>
                </modules>
        </conditionalResource>
</conditionalResources>
```

to conditionally load a resource.

Optionally you can specify OpenMRS version.

The feature was implemented in https://tickets.openmrs.org/browse/TRUNK-3644

Conditional loading based on missing modules

As of OpenMRS 2.2.0 it is possible to control loading beans on the condition that modules are *missing*. This is how a bean is loaded under the condition that the module exti18n is missing:

```
@OpenmrsProfile(modules = {"!exti18n"})
```

Alternatively this can also be done via config.xml:

```
<conditionalResources>
        <conditionalResource>
                <path>/lib/yourmodule-api-*</path>
                <modules>
                        <module>
                                <moduleId>exti18n</moduleId>
                                <version>!</version>
                        </module>
                </modules>
        </conditionalResource>
</conditionalResources>
```

Note that in this case it is important to not complement the use of `@OpenmrsProfile` and `@Component` (or a bean definition in moduleApplicationContext.xml). This is because the bean must be filtered out well ahead of everything else, which would be defeated by Spring processing the `@Component` annotation.

The feature was implemented in https://tickets.openmrs.org/browse/TRUNK-5213