Customizing

There are some things Marmotta allows to customize.

Style

Default styles

Marmotta comes with thee styles:

1. 'blue' (available at 'core/public/style/rblue/', default)
2. 'white' (available at 'core/public/style/white/')
3. 'red' (available at 'core/public/style/red/')

So you can easily switch from any just adding the following configuration:

```
kiwi.pages.style_path = core/public/style/red/
```

Custom styles

Besides the default styles, you could customize the style of the admin interface. For that, you may need you set some concrete configurations:

```
kiwi.pages.project = custom
kiwi.pages.startup = path/to/custom/welcome.html
kiwi.pages.style_path = path/to/custom/style/
kiwi.pages.project.custom.logo = path/to/custom/logo.png
kiwi.pages.project.custom.footer = My custom project, developed with <a href="http://marmotta.incubator.apache.
org">Apache Marmotta</a>
```

Where:

- `kiwi.pages.startup` sets the custom page to use as welcome page
- `kiwi.pages.project` assert that Marmotta will use a custom style
- `kiwi.pages.style_path` indicates the base path were look for the normativle css files (see for example the custom style that LMF uses)
- `kiwi.pages.project.custom.logo` is the relative url to the custom logo
- `kiwi.pages.project.custom.footer` allows to write a custom footer

Templates

Since version 3.2, the templates Marmotta internally uses are being copied to the home directory on deployment (`/path/to/marmotta/templates`):

- `admin.ftl` is the template used to build the admin user interface
- `rdfhtml.ftl` renders RDF resources as HTML
- `404.ftl` provides the error page when a requested resources is not found

That would allow advanced users to customize the user interface by directly hacking the freemarker templates. For instance, to provide a customized HTML view of the RDF resources. But of course it requires to be very careful doing it for not breaking the templating.

If you would need to automatically deploy the templates on startup, you can add an event handler like this to your custom application:

```
public void systemInitialised(@Observes SystemStartupEvent event) {
    String template = TemplatingService.ERROR_TPL;
    File templateDir = new File(configurationService.getHome(), TemplatingService.PATH);
    final File tpl = new File(templateDir, template);
    if (!tpl.exists()) {
        final InputStream in = this.getClass().getResourceAsStream("/" + template);
        if (in != null) {
            try {
                log.debug("Copying custom template {}", template);
                FileUtils.copyInputStreamToFile(in, tpl);
            } catch (IOException e) {
                log.error("Error copying custom template '{}': {}", template, e.getMessage());
            }
        } else {
            log.error("Custom template {} not found, so ignoring", template);
        }
    }
}
```

Modules

You can also build your own applications based on Marmotta, both adding custom modules and/or having your own webapp launcher.

Build your custom module

There is a Maven archetype for a Marmotta Module:

```
mvn archetype:generate \
   -DarchetypeGroupId=org.apache.marmotta \
   -DarchetypeArtifactId=marmotta-archetype-module
```

This will generate the following structure:

```
.
|-- pom.xml
`-- src
    `-- main
        |-- java
        `-- resources
            |-- kiwi-module.properties
            |-- META-INF
            |   `-- beans.xml
            `-- web
                `-- admin
                    |-- about.html
                    |-- configuration.html
                    `-- img
                        `-- clock_small.png
```

Adding it to a custom webapp launcher will give the module its own space in the admin ui.

Build your custom webapp launcher

There is a Maven archetype for a Marmotta Webapp:

```
mvn archetype:generate \
   -DarchetypeGroupId=org.apache.marmotta \
   -DarchetypeArtifactId=marmotta-archetype-webapp
```

After that you will have a new Maven project with a structure like:

```
.
|-- pom.xml
`-- src
    |-- main
    |   |-- resources
    |   |   |-- default-config.properties
    |   |   |-- logback.xml
    |   |   `-- META-INF
    |   |       |-- beans.xml
    |   |       `-- persistence.xml
    |   `-- webapp
    |       |-- index.jsp
    |       `-- WEB-INF
    |           `-- web.xml
    `-- test
        `-- resources
            |-- data
            |-- META-INF
            |   |-- beans.xml
            |-- test-config.properties
            `-- WEB-INF
                `-- test-web.xml
```

Then, moving to the folder that Maven created, you can start it just by running one of the following command:

```
mvn tomcat7:run
```

By default it will start your new webapp at http://localhost:8080

Of course you can customize whatever you need, as soon as you do not break the general setup (servlets and filters).