# FreeMarker Library

# Table of Contents

This module (`isis-module-freemarker`) provides a wrapper around the Freemarker templating engine.

# How to configure/use

## Classpath

Update your classpath by adding this dependency in your dom project's `pom.xml`:

```
<dependency>
    <groupId>org.isisaddons.module.freemarker</groupId>
    <artifactId>isis-module-freemarker-dom</artifactId>
    <version>1.16.1</version>
</dependency>
```

Check for later releases by searching Maven Central Repo.

For instructions on how to use the latest `-SNAPSHOT`, see the contributors guide.

## Bootstrapping

In the `AppManifest`, update its `getModules()` method, eg:

```
@Override
public List<Class<?>> getModules() {
    return Arrays.asList(
            ...
            org.isisaddons.module.freemarker.dom.FreeMarkerModule.class,
            ...
    );
}
```

## Configuration Properties

Optionally, enable/disable JODA support using a configuration property:

```
isis.services.addons.freemarker.jodaSupport=true
```

If not specified, then JODA support is ***enabled***.

# API

In order to understand the API of the service provided by this module (`FreeMarkService`), it's necessary to understand a little of the API/SPI exposed by FreeMarker itself.

## FreeMarker's Design

Freemarker's design is that templates are identified by a template name and a version. Freemarker's `TemplateLoader` SPI is used to lookup the template text and to check whether that template text is stale; Freemarker automatically caches any template text and only reads in new text if a "lastModified" property has changed. The sequence is:

- calls `TemplateLoader#findTemplateSource(templateName)`

  to look up an object that can act as the source of the template text (can return a reader). For example, this templateName might just be a file name.

- then calls `TemplateLoader#getLastModified(templateSource)`

  to determine the time (ie versoin) of this particular source. For example, if the template source is a file, then this would be the lastModified timestamp on this file.

- if necessary, then calls `TemplateLoader#getReader(templateSource, encoding)`.

  Freemarker automatically caches template text, so it will only make this call if the templateSource returns a newer lastModified timestamp than cached.

When FreeMarker passes in the templateName to `#findTemplateSource(.)` it also appends a locale, eg "_en_GB". The idea, obviously, is that locale-specific versions of templates can be returned.

## FreeMarkerService

This module provides a single service, called `FreeMarkerService`. It has the following API:

```
@DomainService(nature = NatureOfService.DOMAIN)
public class FreeMarkerService {
    public String render(
            String templateName,                       ①
            long version,
            String templateChars,                      ②
            Object dataModel)                          ③
        throws IOException, TemplateException;
}
```

① the name/version of the template, corresponding directly to FreeMarker's "templateName".

② the `templateChars` is the actual template text itself

③ the `dataModel` is the object whose values are interpolated by Freemarker into the template. This can either be a strongly typed DTO or just a simple `Map`.

This method takes parameters that (from FreeMarker's point of view) represent both the input to finding a template and also the output (the text of that template):

- the `templateName` broadly corresponds directly to FreeMarker's "templateName".

  To support multiple versions of a template over time, just create a composite name. Similarly, if multiple versions of a template are needed for different app tenancies, combine that into the composite name also.

  For example, to support two versions of an "InvoiceTemplate" for France and for Italy, each published on 1 Dec 2015, one could pass in a template of "Invoice:/FRA:2015-12-01" or "Invoice:/ITA:2015-12-01".

- the `templateChars` is the template text that is used to return a StringReader if and when FreeMarker calls `#getReader(…)`.

Internally FreeMarker caches the template characters; it uses the `templateName` and the `version` as a way to determine whether its internal cache is invalid. As a small wrinkle, it also performs the caching on a per-locale basis; the `templateName` used internally will have a locale appended to it (eg "_en_GB").

All this caching is irrelevant to the `FreeMarkerService`, because it passes in the template characters irrespective; these are simply made available on a thread-local. The responsibility for caching therefore moves outside of FreeMarker, and to the calling application. Thus, the intended usage is that the template characters will be stored in an entity (let's call it `DocumentTemplate`, say) which could be identified by `documentType` and `atPath`, say, and which also is versioned. The `documentType` and `atPath` can simply be joined together to create the logical template name.

FreeMarker also supports the notion of versioned templates (the `TemplateSourc#getLastModified()` API), however there's clearly a subtlety going on somewhere because in an earlier design of this service (which took in a `version` parameter for the template) it didn't seem to work. Since in most cases the templateName is likely to be a composite anyway (for application tenancy), the decision is simply to also include the version number as well in this "template name".

# Example Usage

From the unit tests:

```
// given
 Map<String, String> properties = ImmutableMap.of("user", "John Doe");

// when
String merged = service.render("WelcomeUserTemplate:/GBR:2015-12-01:", "<h1>Welcome
${user}!</h1>",  properties);

// then
assertThat(merged, is("<h1>Welcome John Doe!</h1>"));
```

## Known issues

None known at this time.

# Dependencies

Maven can report modules dependencies using:

```
mvn dependency:list -o -pl modules/lib/freemarker/impl -D excludeTransitive=true
```

which, excluding Apache Isis itself, returns these compile/runtime dependencies:

```
org.javassist:javassist:jar:3.19.0-GA
org.freemarker:freemarker:jar:2.3.25-incubating
```

For further details on 3rd-party dependencies, see:

- Apache Freemarker
- Javassist