Enhancing Freemarker templates with DataGetters

Overview

It is possible for a Freemarker template to display data that is not normally provided to it.

You can create an RDF file that describes a custom `DataGetter` object, and associates it with the desired template. Each time that template is used, the `DataGetter` will be executed, and the data will be stored in a variable, so the template can display it.

This does not require changes to the Java code. You create the RDF file in your VIVO distribution directory and modify the template in your theme.

An example

Let's assume that we need to display information about the most recent data ingest operation. We want to display the name of the Person who supervised the ingest. We would like to display this on every page.

As part of the ingest process, we can load statements like this into the data model:

```
<http://vivo.mydomain.edu/individual/n5242>
    <http://vivo.mydomain.edu/individual/isMostRecentUpdater>
    "true" .
<http://vivo.mydomain.edu/individual/n5242>
    <http://www.w3.org/2000/01/rdf-schema#label>
    "Baker, Able" .
```

We would like for VIVO to display the name of this individual on every page, so the footer will change from this:

to this:

Creating the DataGetter

VIVO allows you to define and use `DataGetter` objects in several contexts. `DataGetters` come in many flavors, but the most commonly used is the `SparqlQueryDataGetter`, which lets you define a SPARQL query, and store the results of that query for your Freemarker template to display.

By adding statements to your data model, you can define a `SparqlQueryDataGetter` object, and associate it with a Freemarker template. Here is the definition that is used in this example:

```
@prefix display: <http://vitro.mannlib.cornell.edu/ontologies/display/1.1#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<freemarker:footer.ftl> display:hasDataGetter display:updatedInfoDataGetter .

display:updatedInfoDataGetter
    a <java:edu.cornell.mannlib.vitro.webapp.utils.dataGetter.SparqlQueryDataGetter> ;
    display:saveToVar "updatedInfo" ;
    display:query """
    PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    PREFIX local: <http://vivo.mydomain.edu/individual/>

    SELECT (str(?rawLabel) AS ?updater)
    WHERE {
        ?uri local:isMostRecentUpdater ?o ;
             rdfs:label ?rawLabel .
    }
    LIMIT 1
     """ .
```

These statements can be added to your data model in any of several ways. For this example, I stored these lines in a file called `data_getter_for_example.n3` and placed it in the VIVO distribution directory under `rdf/display/everytime`. Files placed in this directory are loaded when VIVO starts, but are not persisted when VIVO stops. This allows you to edit or remove the file without leaving residual statements in your data model.

Notice that:

- The first statement says that the Freemarker template `footer.ftl` has a `DataGetter`, defined in subsequent lines.
- The definition of the `DataGetter` states:
    - the type of the data getter,
    - the SPARQL query that will be executed
    - the Freemarker variable that will hold the results.

The results of the SPARQL query are stored in a Freemarker variable, in this case `updatedInfo`. The variable will contain a Sequence of Hashes, where each Hash represents one line of the SPARQL result. Within each Hash, result values are specified as key/value pairs.

For more information on Sequences and Hashes, consult the Freemarker manual:

- Retrieving data from a Sequence
- Retrieving data from a Hash

Modifying the template

Here is the standard `footer.ftl` template:

footer.ftl

```
<#-- $This file is distributed under the terms of the license in /doc/license.txt$ -->
</div> <!-- #wrapper-content -->
<footer role="contentinfo">
    <p class="copyright">
        <#if copyright??>
            <small>&copy;${copyright.year?c}
            <#if copyright.url??>
                <a href="${copyright.url}" title="${i18n().menu_copyright}">${copyright.text}</a>
            <#else>
                ${copyright.text}
            </#if>
             | <a class="terms" href="${urls.termsOfUse}" title="${i18n().menu_termuse}">${i18n().menu_termuse}<
/a></small>
        </#if>
        ${i18n().menu_powered} <a class="powered-by-vivo" href="http://vivoweb.org" target="_blank"
title="${i18n().menu_powered} VIVO"><strong>VIVO</strong></a>
        <#if user.hasRevisionInfoAccess>
             | ${i18n().menu_version} <a href="${version.moreInfoUrl}" title="${i18n().menu_version}">${version.
label}</a>
        </#if>
    </p>
    <nav role="navigation">
        <ul id="footer-nav" role="list">
            <li role="listitem"><a href="${urls.about}" title="${i18n().menu_about}">${i18n().menu_about}</a><
/li>
            <#if urls.contact??>
                <li role="listitem"><a href="${urls.contact}" title="${i18n().menu_contactus}">${i18n().
menu_contactus}</a></li>
            </#if>
            <li role="listitem"><a href="http://www.vivoweb.org/support" target="blank" title="${i18n().
menu_support}">${i18n().menu_support}</a></li>
        </ul>
    </nav>
</footer>
<#include "scripts.ftl">
```

Insert these lines between lines 17 and 18:

```
        <#if (updatedInfo?first.updater)??>
            | Updated by ${updatedInfo?first.updater}
        </#if>
```

The SPARQL result is obtained and stored into the Freemarker variable `updatedInfo` each time the `footer.ftl` template is loaded for display. The name we want is in the first row of the SPARQL result, keyed to the name `updater`.

Summary

Enhancing Freemarker templates is one more way to use the VIVO `DataGetter` mechanism. When you associate a `DataGetter` with a Freemarker template, that `DataGetter` will be run each time the template is invoked. This is true whether the template is specified by the controller, or included in another template. You can modify the template to display the data from the `DataGetter`, but it is prudent to include an `<#if>` tag, so your template won't fail if the data is not found.