The Developer Panel

Diagnostic tools that can help you figure out what VIVO is doing.

Introduction

Are you developing code for VIVO? Are you doing some extreme customization? You might benefit from VIVO's set of built-in diagnostic tools.

These tools help to reveal how VIVO operates behind the scenes. In general, they are only used during development, because they may have serious negative effects on the performance of the VIVO application. However, they may also be used (carefully) in production, to diagnose a particularly difficult problem.

The diagnostic tools are enabled and controlled by settings within VIVO. These settings may be changed interactively, without restarting VIVO. The settings may also be read from a file, so they will be in effect as VIVO is starting up.

Terms

Developer Mode

When the diagnostic tools are enabled, VIVO is said to be running in "Developer Mode". This reflects the fact that all of the developer settings are ignored unless the tools as a whole are enabled.

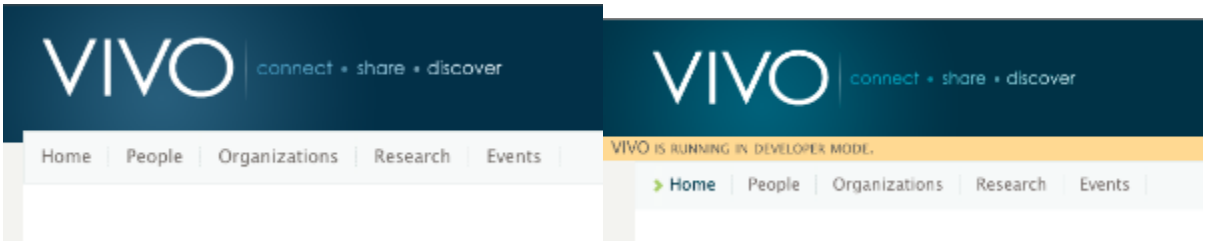
Developer Settings

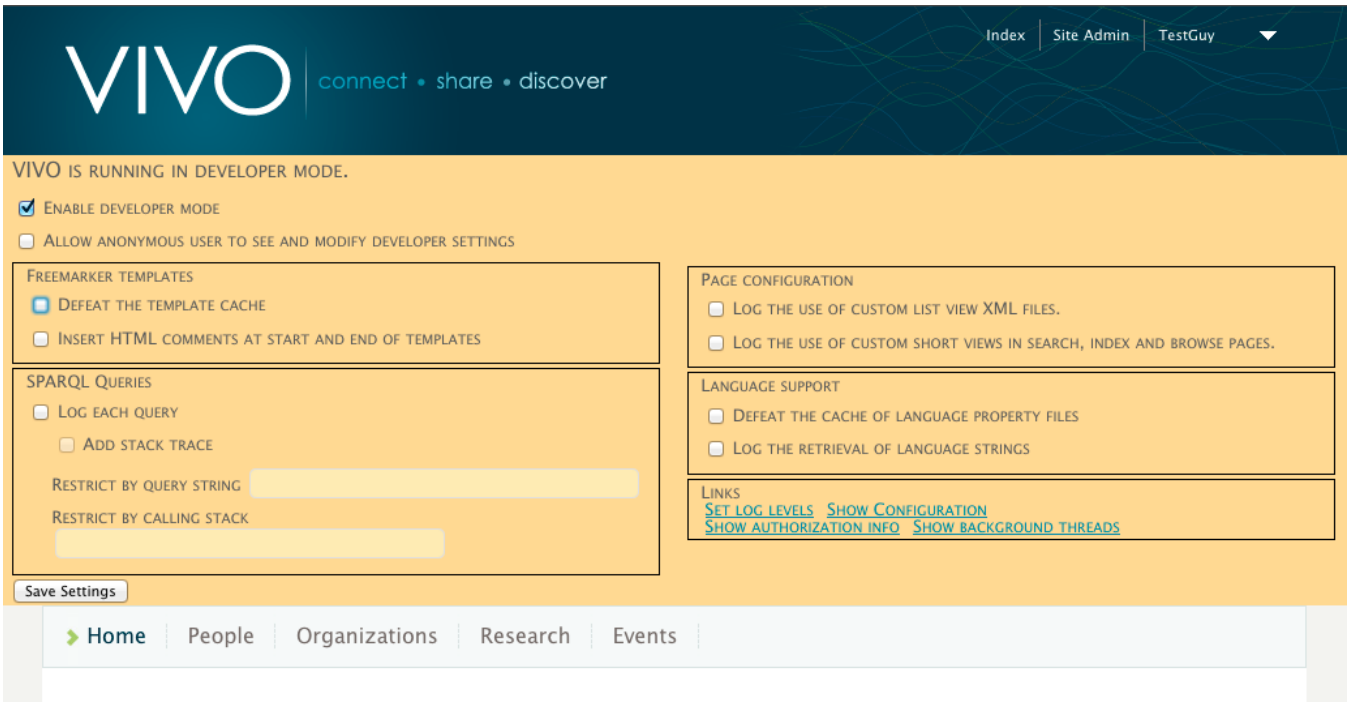
These are parameters that control the diagnostic tools. They may be set interactively, using the Developer Panel, or read from the `developer.settings` file at startup.

The Developer Panel

When VIVO is in developer mode, the Developer Panel appears on every page. This serves two purposes:

1. It enables you to change the Developer Settings without navigating away from your current page.
2. It provides a visual reminder that VIVO is in Developer Mode. If a production instance were accidentally configured to run in Developer Mode, it would be easily noticed.





Entering Developer Mode

`developer.properties` file

When VIVO starts up, it looks for a file in the home directory, named `developer.properties`. If the file is found, the settings are read from it. Any settings that are not found in the file keep their default values. If the file is not found, then all settings keep their default values until set interactively.

A typical developer.properties file

```
developer.enabled=true
developer.permitAnonymousControl=true
developer.defeatFreemarkerCache=true
```

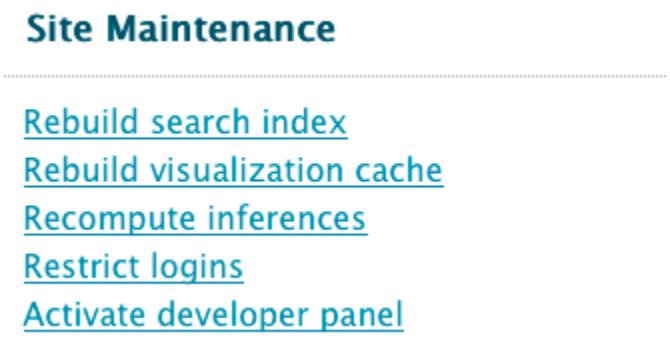
This example causes:

- Developer Mode is enabled immediately. The specified settings are in effect even while VIVO is starting.
- Users who are not logged in can manipulate the developer settings. Obviously, this should only be permitted on a development system.
- The Freemarker template cache is defeated. Each time a template is requested, it will be loaded from disk. This will cause VIVO to run more slowly, but it means that a developer can see the effects of a template change immediately, instead of waiting for the template to expire and be reloaded.

The VIVO distribution includes an `example.developer.properties` file. It contains descriptions of all of the settings, with examples. You can rename `example.developer.properties` to `developer.properties`, and uncomment the settings you want to use.

Interactively entering developer mode

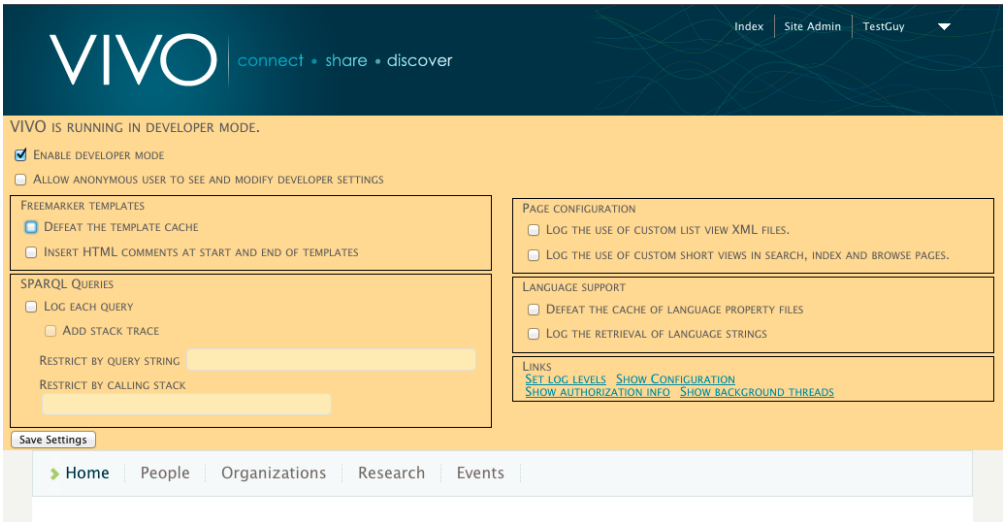
Log in as a system administrator (or root). Go to the `Site Administration` page. Click on `Activate developer panel`.



The Developer Panel will immediately appear below the page header. You may click on the panel to open it and change the settings. The Developer Panel will continue to appear in every page (except for some "back-end" pages for editing the ontology).

The settings

When open, the developer panel looks like this:



The following tables show the meaning of each setting in the developer panel, and how to specify it in the `developer.properties` file.

General settings

In the panel	Enable developer mode
In the file	`developer.enabled`
Effect	Causes the developer panel to be displayed on each VIVO page. Enables all of the other developer settings. If this is `false`, other settings will retain their values, but will not take effect.

In the panel	Allow anonymous user to see and modify developer settings
In the file	`developer.permitAnonymousControl`
Effect	If `true`, any VIVO user may change the developer settings. If `false`, only a system administrator (or root) may change the settings.

Freemarker settings

In the panel	Defeat the template cache
In the file	`developer.defeatFreemarkerCache`
Effect	If `true`, each Freemarker template is loaded from disk each time it is used. If `false`, a template change may be on disk for up to one minute before it is loaded.

In the panel	Insert HTML comments and start and end of templates
In the file	`developer.insertFreemarkerDelimiters`

Effect	If `true`, you may view the HTML source for a VIVO page to see which Freemarker templates were used to create each portion of the page. For example:
	`...`
	`<!-- FM_BEGIN view-search-default.ftl -->`
	`<a href="/vivo/display/n2252" title="individual name">Oswald, Jeremiah</a>`
	`<span class="display-title">Faculty Member</span>`
	`<p class="snippet"></p><!-- FM_END view-search-default.ftl -->`
	`...`

SPARQL Query settings

In the panel	LOG each query
In the file	`developer.loggingRDFService.enable`
Effect	Write an entry to the log for each SPARQL query, assuming that `INFO`-level logging is enabled for the `RDFServiceLogger`. Each entry includes • The time spent executing the query • The name of the method on `RDFService` that received the query • The format of the result stream from `RDFService` • The text of the query. The remaining settings in this area can be used to restrict which queries are logged, or to include more information for each query.

In the panel	Add stack trace
In the file	`developer.loggingRDFService.stackTrace`
Effect	Each log entry will include a stack trace. The trace is abridged so it starts after the `ApplicationFilterChain`, omits any Jena classes, and ends at the `RDFService`.

In the panel	Restrict by query string
In the file	`developer.loggingRDFService.queryRestriction`
Effect	Set this to a regular expression. A query will be logged only if the text of the query matches the regular expression, in whole or in part.

In the panel	Restrict by calling stack
In the file	`developer.loggingRDFService.stackRestriction`
Effect	Set this to a regular expression. A query will be logged only if the abridged calling stack matches the regular expression, in whole or in part.

Page configuration settings

In the panel	Log the use of custom list view XML files.
In the file	`developer.pageContents.logCustomListView`
Effect	Write an entry to the log each time a property is displayed using a list view other than the default lists view.

In the panel	Log the use of custom short views in search, index and browse pages.
In the file	`developer.pageContents.logCustomShortView`
Effect	Write an entry to the log each time a search result is displayed using a short view other than the default view for that context.

Language support settings

In the panel	Defeat the cache of language property files
In the file	`developer.i18n.defeatCache`
Effect	If `true`, the language property files are re-loaded each time they are called for. If `false`, the language property files are loaded only once, when VIVO starts up.

In the panel	Log the retrieval of language strings
In the file	`developer.i18n.logStringRequests.`
Effect	Write an entry to the log each time a language-specific string is retrieved from one of the language property files.

The links

The developer panel also contains several links to special VIVO pages that may be helpful to developers.

Link text	Set log levels
URL	`/admin/log4j.jsp`
The page	Displays the logging levels of every Java class in VIVO, providing that it has an active `Log`. You must be logged in as a system administrator (or root) to use this page. Find the class you are interested in, set the logging level, then scroll to the bottom of the page to `Submit changes to logging levels`.

Link text	Show Configuration
URL	`/admin/showConfiguration`
The page	Displays a list of the `build.properties` and `runtime.properties`. Displays a list of the System properties in the Java virtual machine.

Link text	Show authorization info
URL	`/admin/showAuth`
The page	Displays information about the user who is currently logged in, the identifiers associated with that user, and the permissions they have been granted. Display information about the configured `Policy` objects, and related objects.

Link text	Show background threads
URL	`/admin/showThreads`

The page	Displays information about the active background threads. These threads may be rebuilding the search index, re-inferencing the knowledge base, or rebuilding the Class Cache.

What if the developer panel doesn't appear?

If you are using a custom theme, and you created it from a VIVO release prior to 1.6, it's likely that your theme doesn't display the developer panel.

Confirm that the template `[vivo]/webapp/themes/[your_theme]/templates/menu.ftl` contains an `include` directive like this one:

Excerpt from menu.ftl

```
<#-- $This file is distributed under the terms of the license in /doc/license.txt$ -->

</header>

<#include "developer.ftl">

<nav role="navigation">

...
```