**vHMML**

**Configuration and Development Guide**

Prepared on 12/23/2016 by  
  
Chad LaVigne, Consultant, sdg.com

Abridged edition by William Straub, Systems Librarian, HMML Feb. 2017

Contents

[Design Overview](#_Toc470299681)

[Software](#_Toc470299682)

[The vHMML Application](#_Toc470299683)

[IIIF Service](#_Toc470299684)

[Digilib Image Server](#_Toc470299685)

[Servers](#_Toc470299686)

[Infrastructure](#_Toc470299687)

[Development Environment](#_Toc470299688)

[Production Environment](#_Toc470299690)

[Server Details](#_Toc470299691)

[Development](#_Toc470299692)

[Production](#_Toc470299694)

[Build Server](#_Toc470299695)

[Build shell scripts](#_Toc470299696)

[Build & Deployment Process](#_Toc470299697)

[Building with Jenkins](#_Toc470299698)

[Deploying with Jenkins](#_Toc470299699)

[Moving to Production](#_Toc470299700)

[Development Environment Setup](#_Toc470299701)

[Install Development Software](#_Toc470299702)

[Checking Out Source Code](#_Toc470299703)

[Quick Start](#_Toc470299704)

[Detailed Instructions](#_Toc470299705)

[Local Database, Search & Image Server Setup](#_Toc470299706)

[Starting Docker](#_Toc470299707)

[Eclipse Server Setup](#_Toc470299708)

[Development](#_Toc470299709)

[Creating New Static Pages](#_Toc470299710)

[Adding New Pages to Existing Page Categories](#_Toc470299711)

[Create the page](#_Toc470299712)

[Add the Controller Method](#_Toc470299713)

[Creating Help Pages](#_Toc470299714)

[Create the Page & Controller Method](#_Toc470299715)

[Update the HelpLinkFilter](#_Toc470299716)

[Elastic Search](#_Toc470299717)

[Synonyms](#_Toc470299718)

[Database Reference](#_Toc470299719)

[Import/Export](#_Toc470299720)

[Data Export](#_Toc470299721)

[Data Import](#_Toc470299722)

[Connecting to MySQL with MS Access](#_Toc470299723)

[Appendix](#_Toc470299724)

[Useful Linux Commands](#_Toc470299725)

[Database Table Reference](#_Toc470299726)

[General Application Tables](#_Toc470299727)

[Lexicon Tables](#_Toc470299728)

[Reference Tables](#_Toc470299729)

[Reading Room Tables](#_Toc470299730)

[Useful Queries](#_Toc470299731)

[Show RR Objects in Collection without thumbnail selected](#_Toc470299732)

[Grant role to user by username](#_Toc470299733)

[Show roles for user](#_Toc470299734)

[Create country, city, repository, etc.](#_Toc470299735)

[Update country, city, repository, etc.](#_Toc470299736)

[Show Reading Room object by contributor](#_Toc470299737)

[Show Reading Room object for a piece of content](#_Toc470299738)

[Show Reading Room content by HMML Project Number](#_Toc470299739)

[Show Reference entry by id](#_Toc470299740)

[Show Lexicon entry by id](#_Toc470299741)

[Delete Reading Room data](#_Toc470299742)

# Design Overview

The vHMML application is a distributed system comprised of several applications running on multiple servers.

## Software

### The vHMML Application

This is a custom web application that is deployed to [Tomcat](https://tomcat.apache.org). It was developed using Java technologies such as [Spring](https://projects.spring.io/spring-framework/), [Hibernate](http://hibernate.org/orm/) and [Elastic Search](https://www.elastic.co/products/elasticsearch) along with various web technologies such as HTML, CSS, Javascript, [jQuery](https://jquery.com/) and [Twitter Bootstrap](http://getbootstrap.com/).

### IIIF Service

The IIIF Service is another custom web application that is responsible for generating IIIF compliant image meta data that can be consumed by IIIF compliant image viewers, such as [Mirador](http://projectmirador.org/). This is a fairly small application with the primary job of reading vHMML images and generating meta data from those images that contains information about the images such as width, height, etc. When this application is built, it also generates and executable jar file name iiif-util.jar that can be used to generate the image portion of a IIIF manifest. This is useful because generating the meta data for the images is very resource intensive so it isn’t reasonable to generate this data in real time.

### Digilib Image Server

#### Configuring Digilib

The vHMML application uses IIIF compliant image server software named [Digilib](http://digilib.sourceforge.net/) to generate images requested by the [Mirador](http://projectmirador.org/) viewer. Digilib is a Java web application that is packaged as a .war file and deployed to Tomcat on separate hardware to offload CPU intensive image processing work and allow for optimization. Digilib provides [several configuration options](http://digilib.sourceforge.net/digilib-config.html) that can be configured via a file named digilib-config.xml. This file is currently located in the web application in the WEB-INF directory. Ideally, this file should be moved to a location outside the application at some point to avoid loss of configuration changes when a new version of Digilib is deployed. The current configuration for Digilib can be view at http://<hostname >:<port>/digilib/server/dlConfig.jsp. Because the image servers can only be access via ssh, the easiest way to retrieve this configuration is by using ssh to log into the image server and accessing the config URL using curl on the command like, e.g. curl <http://localhost:8080/digilib/server/dlConfig.jsp>.

#### Optimization

If image retrieval is not performing adequately, one option to help optimize performance is to render pre-scaled images. Digilib provides the capability to deliver pre-scaled images out of the box. Basically, the Digilib basedir-list property can contain multiple directories that are searched from last to first when an image is requested. The first matching image that is found that is at least as large as the image size requested will be served. This can yield large performance gains if the original images are very large because a smaller, pre-scaled image can be read into memory instead of the large original. If an image is found that exactly matches the requested image size, then the image is served without any scaling, resulting in much better performance. More details can be found [here](http://digilib.sourceforge.net/image-directories.html).

#### Troubleshooting Image Issues

##### Images for Project X won’t show

1. Check the manifest
   1. SSH into the web server
   2. SSH from the web server to the image server
   3. Get the manifest from the image server: curl <http://localhost:8080/iiif-service/manifest/GARZ%2000005>
      1. Note that the HMML project number on the end of the URL must be URL encoded, this is why the space in “GARZ 00005” is replaced with %20 in the example above.
   4. If there is an error when you request the manifest, try to diagnose the stack trace. Go to the images directory for the object and make sure the data.json file containing image data for the manifest exists.
      1. If the meta data doesn’t exist, generate it.
      2. If the meta data does exist, get the data and verify it.
         1. First, paste the text into <http://jsonlint.com/>. If the JSON doesn’t pass validation, re-generate the meta data.
         2. If the JSON passes validation, open the file in a text editor and make sure there are no null elements by searching for the text “null”. At times the meta data generation creates corrupt data and null values are inserted. If you find a null value, regenerate the meta data.

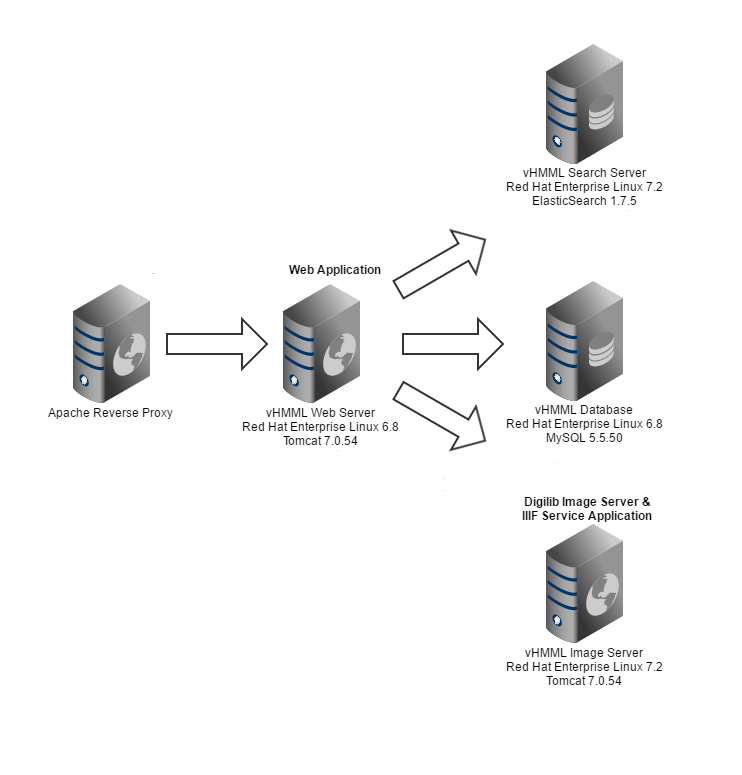
If the config request responds, then the problem is probably that the IIIF Service can’t read the manifest image files. Perhaps Tomcat has too many files open?

# Servers

There vHMML application runs in three environments, Development, Test and Production. The Development environment is for ongoing development work, the Test environment is for testing of release candidate builds and the Production environment is the actual vhmml.org web site. The Development environment runs the entire vHMML eco system on a single server while Test and Production are comprised of several servers. The Test and Production infrastructure are outlined in the infrastructure diagrams below.

## Infrastructure

### Production Environment



## Server Details

As part of day-to-day operations and troubleshooting, it is helpful to keep a list of the servers handy so you can quickly log into these servers, monitor activity, move files, etc. Below is a reference for each environment containing server information such as server name, IP address and helpful file locations.

One option for this server set up is to run the Web, Search, Image and Database servers all on one machine. This is recommended only for a development or environment, but not production.

* + Start Tomcat Command: sudo /etc/init.d/elasticsearch start
  + Stop Tomcat Command: sudo /etc/init.d/elasticsearch stop
  + Logs
    - /usr/share/tomcat7/logs/vhmml-log.txt – This is the main log file that the vhmml web application logs to, configuration for it can be found in /usr/share/tomcat7/webapps/ROOT/WEB-INF/classes/\_\_\_\_.xml.
    - /usr/share/tomcat7/logs/digilib-log.txt – This is Digilib’s log file, configuration for it is in /usr/share/tomcat/webapps/digilib/WEB-INF/\_\_\_\_-config.xml
    - /usr/share/tomcat7/logs/iiif-log.txt – This is the log file for the IIIFService application, configuration for it is in /usr/share/tomcat/webapps/iiif-service/WEB-INF/classes/\_\_\_\_.xml
    - /usr/share/tomcat7/logs/catalina.out – By default, this is the log file where Tomcat puts all messages logged to the console, so all messages from Tomcat and from our applications will get logged in this file. In the vHMML application and IIIF Service application, only the root logger is configured to log to the console and this logger is set to WARN by default so that the log files don’t grow too quickly.

### Production

* Web Server
* Search Server
* Image Server
* Database
  + To connect to the database via the command line you can use the following commands (you will be prompted for a password when you hit enter):

|  |  |  |
| --- | --- | --- |
| Syntax | | Example |
| mysql –u<username> -p <schema> | mysql -asparagus-p broccoli | |
| mysql –h<hostname> -u<username> -p <schema> | mysql -jello -asparagus-p broccoli | |

### Build Server

The build server runs Jenkins for automated build and deployment. There is a build job and a deploy job for both the vHMML application and the IIIFService application.

### Build shell scripts

Build scripts are in /usr/local/bin

# Build & Deployment Process

[Jenkins](https://jenkins.io/) automation server is used to build and deploy the vHMML application. You can login in to Jenkins from your browserwe have four Jenkins jobs configured:

* vHMML Build
* vHMML Deploy
* IIIF Service Build
* IIIF Service Deploy

As the names imply, we have jobs to build the vHMML and IIIF Service applications and we also have jobs to deploy those applications to our Dev, Test and Production environments. The build jobs compile the applications and package them into web application archive (.war) files that can be deployed to a servlet container, e.g. Tomcat. The deploy jobs move the build artifacts (.war files) to the environment you specify when running the job. The vHMML build job is set to run whenever new code is checked into Subversion. This continuous integration is in place to provide immediate feedback about issues created by code changes, for example compile errors or test failures.

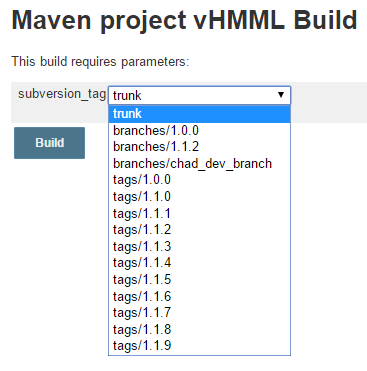
## Building with Jenkins

The following steps outline how to build the vHMML application. The steps for building the IIIF Service application are identical to these steps with the exception that the IIIF Service Build Jenkins job is run instead of the vHMML Build job.

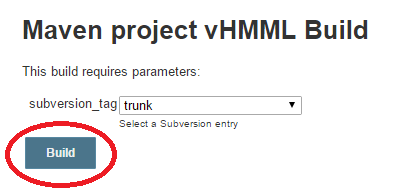
1. Log into Jenkins
2. Click on the build icon next to the vHMML Build job



1. Select the version you would like to build from the “subversion\_tag” dropdown list. Note that the “trunk” tag is the latest code from the repository.



1. Click the Build button to run the build.



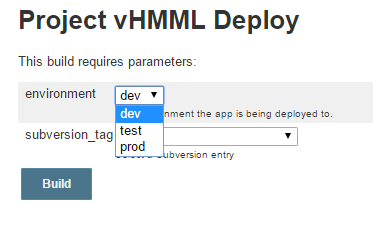
### Deploying with Jenkins

The following steps outline how to deploy the vHMML application. The steps for deploying the IIIF Service application are identical to these steps with the exception that the IIIF Service Deploy Jenkins job is run instead of the vHMML Deploy job.

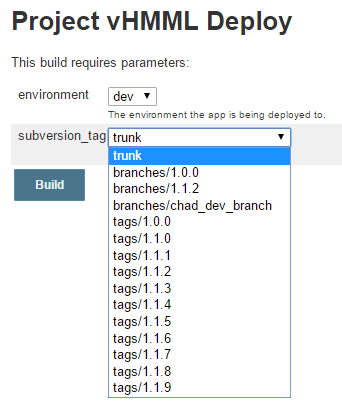
1. Log into Jenkins
2. Click on the build icon next to the vHMML Deploy job



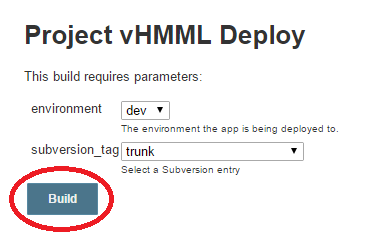
1. Select the environment you would like to deploy the application to from the “environment” dropdown list.



1. Select the version you would like to build from the “subversion\_tag” dropdown list. Note that the “trunk” tag is the latest code from the repository it is **IMPORTANT** that you do not deploy the trunk to the Production environment as this contains untested development code.



1. Click the Build button to run the deployment.



### Moving to Production

There are several additional steps required when moving a new version of vHMML to production.

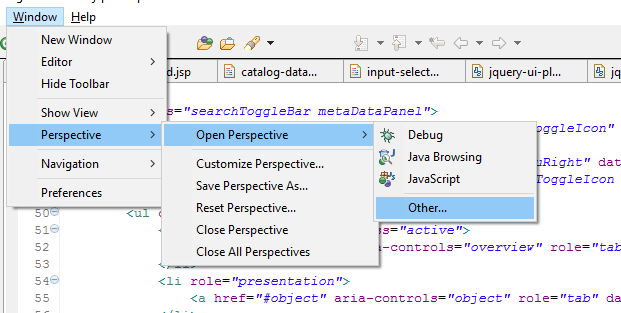
#### Tagging a build

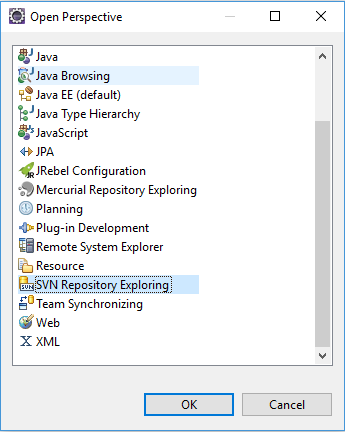
When planning a build for production, the vHMML code should be “tagged” in Subversion. Tagging is a simple process that applies a version number to the application code. This is important for a few reasons:

* We know exactly what code is deployed to production
* We can attach version information to issues providing traceability
* We can move the versioned artifact to different environments and know exactly what is deployed

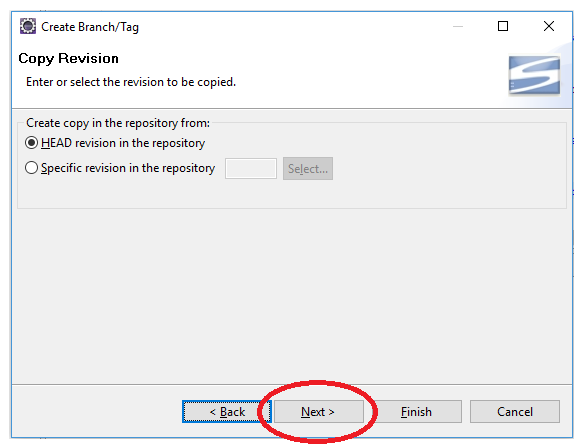
The following steps outline the process for tagging a build using the Subclipse plugin in Eclipse. For instructions on how to set up Subversion and connect to the repository see the [section on checking out the source code](#_Detailed_Instructions).

1. Navigate to the SVN Repository Exploring perspective in Eclipse, Window->Perspective ->Open Perspective -> Other. From the Open Perspective dialog, select SVN Repository Exploring.

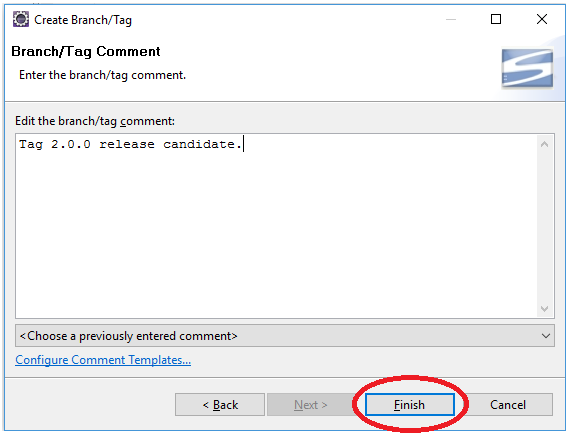




1. On the SVN Repositories tab, expand the vhmml\_vhmmlsdg repository, right click on the “trunk” folder and select “Branch/Tag…”.
2. Enter a location for the tag in the “Copy to URL” field and click the Next button. The location should go under the tags directory. Please note that the default location will be the trunk folder, be sure to change it to tags/<your\_tag\_number>. For example, if you’re tagging version 2.0.0 then you would enter [https: your\_repo/reponame/vhmml/tags/2.0.0](https://vhmml.unfuddle.com/svn/vhmml_vhmmlsdg/vhmml/tags/2.0.0).
3. On the Copy Revision dialog click the Next button, accepting the default option to copy the HEAD revision.



1. Enter a comment on the Branch/Tag comment dialog to indicate that you’re tagging a version and click the Finish button.



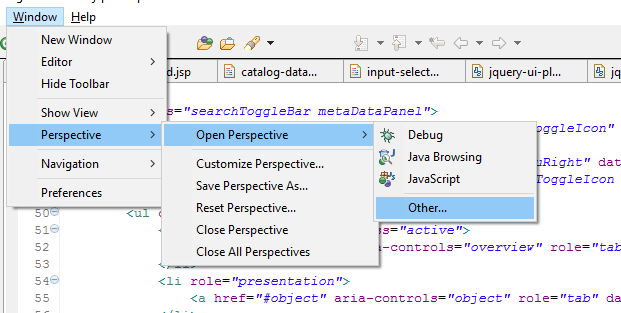
#### Branching

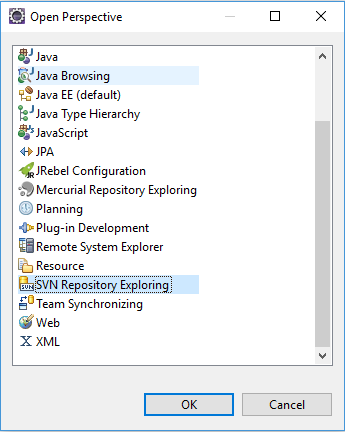
After a tagged version of the application has passed regression testing and you’re ready to move the code to production, a branch copy of the release candidate code should be made. Like a tag, a branch is just another copy of the application code but it serves a different purpose. The branch is created to support the production version of the application. It’s a copy of the tagged version that can be checked out by developers, allowing them to test and fix issues against the same code that is running in production. This provides a mechanism for the development team to implement bug fixes for the production code in isolation from on-going development. If an emergency fix is needed in production, the basic steps for implementing and deploying a hot fix are:

1. Check out the branch (it is helpful to create a new Eclipse workspace for the branch)
2. Implement and test the fix
3. Check the fix into the branch AND the trunk
4. Create a new tag from the branch
5. Build and deploy the new tag [using the Jenkins build](#_Building_with_Jenkins)

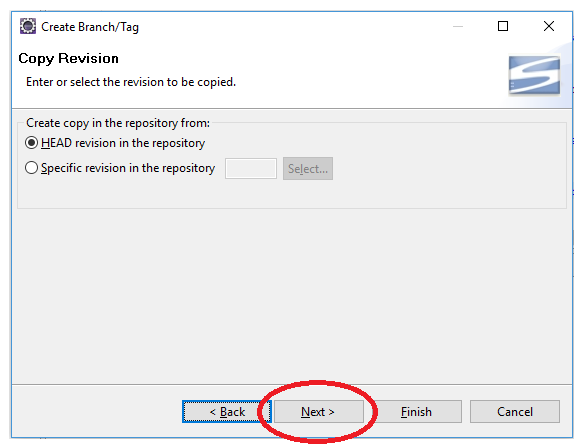
The following steps outline the process for creating a branch using the Subclipse plugin in Eclipse. For instructions on how to set up Subversion and connect to the repository see the [section on checking out the source code](#_Detailed_Instructions).

1. Navigate to the SVN Repository Exploring perspective in Eclipse, Window->Perspective ->Open Perspective -> Other. From the Open Perspective dialog, select SVN Repository Exploring.

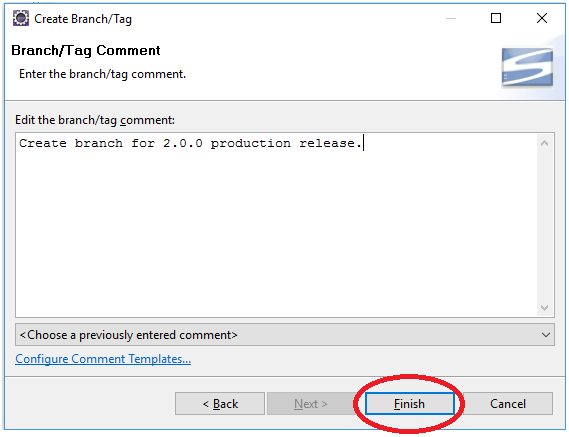




1. On the SVN Repositories tab, expand the repository. Open the “tags” folder and right click on the release candidate tag you are creating a branch from and select “Branch/Tag…”.
2. Enter a location for the branch in the “Copy to URL” field and click the Next button. The location should go under the branches directory. Please note that the default location will be the tags folder, be sure to change it to branches/<your\_tag\_number>. For example, if you’re creating a branch from the version 2.0.0 then you would enter https: yourrepo/svn/reponame/vhmml/branches/2.0.0.
3. On the Copy Revision dialog click the Next button, accepting the default option to copy the HEAD revision.



1. Enter a comment on the Branch/Tag comment dialog to indicate that you’re creating a branch and click the Finish button.



#### Database backup

Before promoting a build to production it is very important to back up the production database so we can revert to the previous version of the application in the case of an emergency issue. Our process for backing up the data is to use the mysqldump command provided by MySQL to export the data to a file that can be imported later if necessary. For instructions on how to perform a data export, please see the [Import/Export](#_Import/Export) section of the Data Overview in this document.

# Development Environment Setup

This documentation assumes you are developing with Eclipse on a Windows workstation using JDK version 1.7. Newer versions of Java, Eclipse and Tomcat may work, but they haven’t been tested and would likely require configuration that may differ from what is documented here.

## Install Development Software

1. Download and install JDK 1.7, this JDK can be downloaded from the [archival downloads section on the Java web site](http://www.oracle.com/technetwork/java/javase/archive-139210.html). Because this is an archived version of Java, you need to create an account on the Oracle web site to download JDK 1.7. After downloading the installer simply run it, accepting all the default options.
2. Download and Install [Eclipse](https://www.eclipse.org/downloads/). At the time of this writing, Eclipse Mars was the current version. After downloading Eclipse, installation is just a matter of extracting the downloaded zip file into a directory of your choosing.
3. Navigate to the directory where you have extracted Eclipse, find the file called eclipse.exe and double click that file to open Eclipse. You may want to make a shortcut to this file to make it easy to access in the future.
4. When you open Eclipse you are prompted to selected a workspace. A workspace is Eclipse’s name for the directory where you projects are stored. The default us usually something like C:\Users\yourname\workspace. You can accept the default or choose a different location if you would like to store the files elsewhere, then click the OK button to open Eclipse.
5. Install a database client for connecting to the database such as [MySQL Workbench](https://dev.mysql.com/downloads/workbench/5.2.html).
6. You will also want to download and install software to extract zip files such as [7zip](http://www.7-zip.org/download.html), an scp client such as [WinSCP](https://winscp.net/eng/download.php) to transfer files between different and an ssh client such as [PuTTY](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) for logging into the various server environments.
7. When Eclipse opens for the first time you are presented with a Welcome screen. Click the Workbench button to close this screen and go to the workbench where you can begin setting up the vhmml project by [checking out the source code](#_Checking_Out_Source_1) as explained in the next section.



## Checking Out Source Code

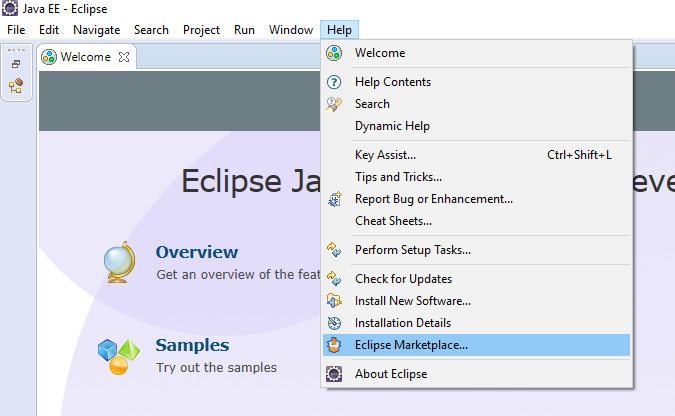
The vHMML application is stored in a hosted Subversion repository. Before you start, make sure you have credentials to authenticate to this repository.

### Quick Start

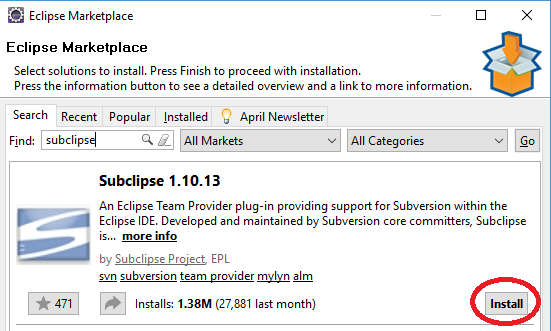
Open your repo from the URL for your Subversion repository.

### Detailed Instructions

1. Download and install a Subversion client. [SlikSVN](https://sliksvn.com/download/) or [TortoiseSVN](https://tortoisesvn.net/downloads.html) are good options. You will most likely interact with the code repository using the Subclipse plugin for Eclipse, but a Subversion client is required by that plugin. Simply download the installer and run it, accepting defaults configuration values during the installation.
2. Install the Subclipse Eclipse plugin.
   1. In Eclipse, go to Help -> Eclipse Marketplace…

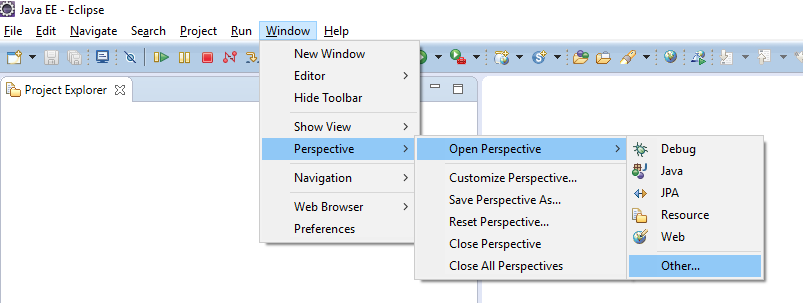


* 1. Type “subclipse” in the Find search field and press enter

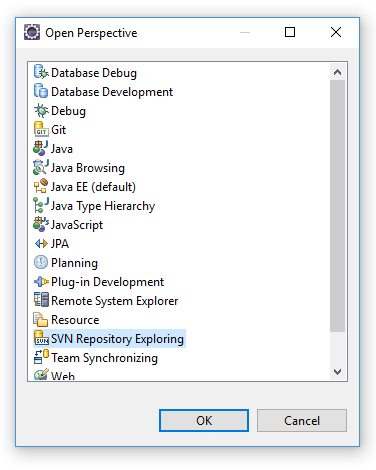


* 1. Subclipse should be the first option, click the Install button next to Subclipse
  2. You will be presented with several prompts during the install asking you to do things like accept the license agreement and agree to installing “unsigned content”. Agree to these prompts and follow the Subclipse install instructions, accepting default configuration values.
  3. Restart Eclipse when prompted at the end of the Subclipse install.
  4. Upon restarting Eclipse you will be prompted to “Report usage of Subclipse to Subclipse team”. If you do not wish to do this, be sure to uncheck the checkbox before clicking ok.

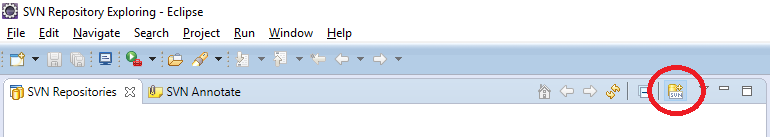
1. Connect to Subversion Repository
   1. In Eclipse, go to Window -> Perspective -> Open Perspective -> Other



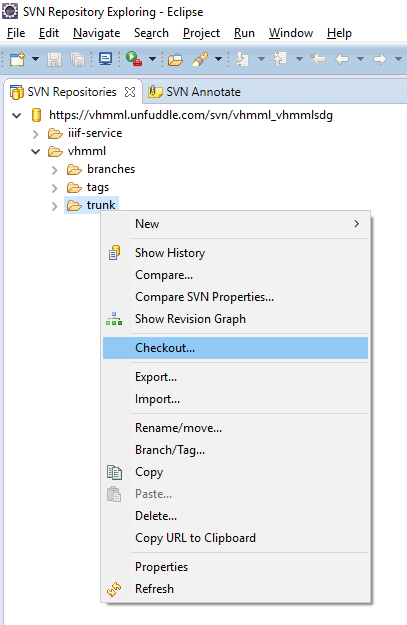
* 1. In the dialog that opens, select SVN Repository Exploring and click OK.



* 1. When the Repository Exploring perspective opens, click the “Add SVN Repository” icon.



* 1. In the URL field on the Add SVN Repository dialog, enter your SVN repo URL and click Finish.
  2. Provide your repository credentials when prompted, clicking the Save Password checkbox so you won’t be prompted repeatedly.
  3. Expand the “vhmml” project, right click on the “trunk” folder and select “Checkout…”



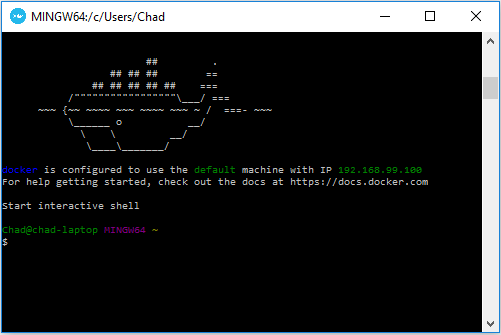
* 1. Click finish in the “Checkout from SVN” dialog, accepting default configuration values. The checkout will take quite a while depending on your connection due to the very large number of images included in the School portion of the application.

1. When you have finished checking out the vHMML project, repeat the previous steps to check out the iiif-service project.

## Local Database, Search & Image Server Setup

In order to run vHMML locally on your workstation, you will need a local database, image server and search server. To make setup and configuration of these servers easier we have created [Docker](https://www.docker.com/) images that will download and configure the required software and run the servers locally in Docker containers. The following section outlines the steps required to get these Docker containers up and running.

1. If you are on Windows, before installing Docker, if you have Oracle VirtualBox installed it needs to be uninstalled. Docker runs on Oracle VirtualBox on Windows and is very particular about the version of VirtualBox it uses (which it will install during the Docker installation).
2. Download and install [Docker](https://www.docker.com/). Note that the installation is operating system dependent. Specifically, Windows versions prior to Windows 10 require [Docker Toolbox](https://www.docker.com/products/docker-toolbox), not the regular Docker download.
3. Run the downloaded Docker installer, accepting default options as you go and granting permission to install software/modify files when you are requested to do so. On Windows, be sure to right click on the installer and select “**Run as Administrator**”, this is **IMPORTANT**, if you don’t run it as administrator you will have to uninstall Docker and re-install it because it will fail to start. When prompted to allow the program to make changes to the machine, select “Yes”.
4. When the Docker installer is finished, if you are on Linux or Mac, you can skip to step 6. On Windows, you will be taken to the Docker installation folder where there are two files. Right click on the one named “Docker Quickstart Terminal” and click “**Run as Administrator**”. When prompted to allow the program to make changes to the machine, select “Yes”.
5. The Docker Quickstart Terminal will configure the underlying virtual machine that the Docker containers run on and start it. When the “Docker Quickstart Terminal” is done starting up the VirtualBox VM, you will see a Docker startup screen that looks like this:



1. The lightweight Virtual Box VM created by Docker doesn’t allocate enough memory to do large ElasticSearch operations, such as a full re-index. To increase the amount of memory allocated, follow the steps below:
   1. Open a PowerShell window **as Adminstrator** and navigate to the directory where VirtualBox is installed. This is most likely somewhere like C:\Program Files\Oracle\VirtualBox.
   2. In order to allocate more memory to the Docker virtual machine we have to stop it. To stop the Docker virtual machine with this command:

docker-machine stop

* 1. Execute the following command to allocate more memory to the virtual machine:

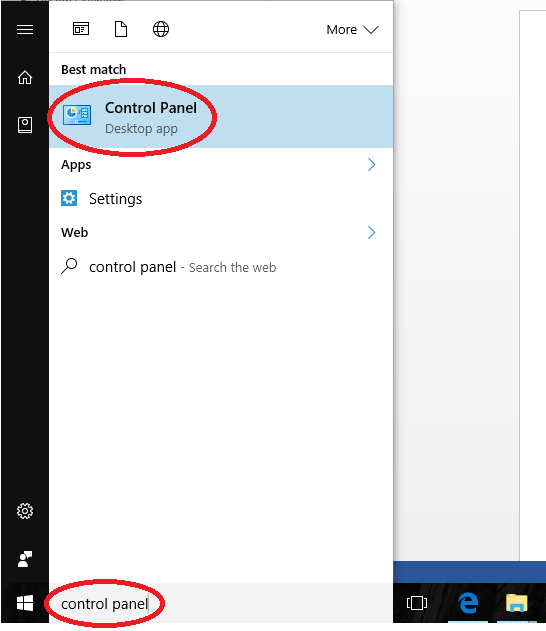
./VBoxManage modifyvm default --memory 4096

* 1. Start the docker virtual machine with this command:

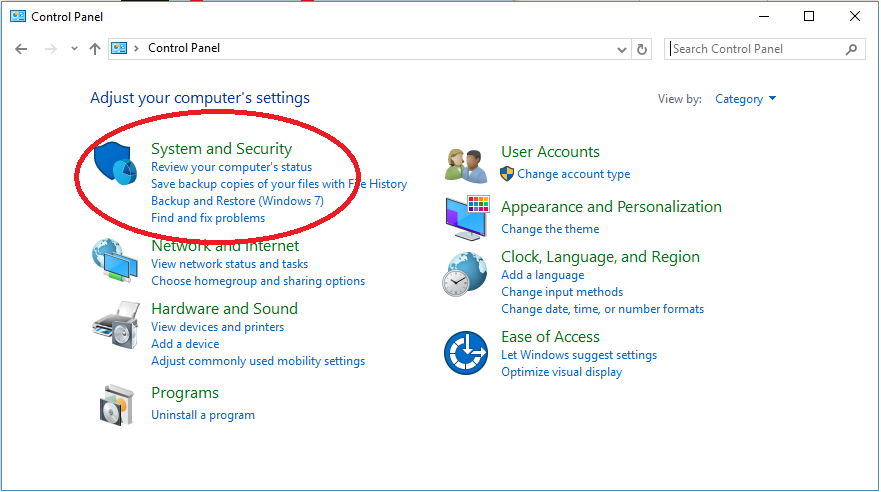
docker-machine start

Please note that the previous command is setting the allocated memory to 4GB. If your machine doesn’t have enough RAM to support 4GB, 2GB should suffice. Also, at the time of this writing, the name of the virtual machine created by Docker is “default”, as indicated in the command above. The name of the machine is displayed when it’s started via the Docker Quickstart Terminal program as can be seen in the screen shot in step 4. If the virtual machine created for you was given a different name, you will need to use that name in the memory allocation command.

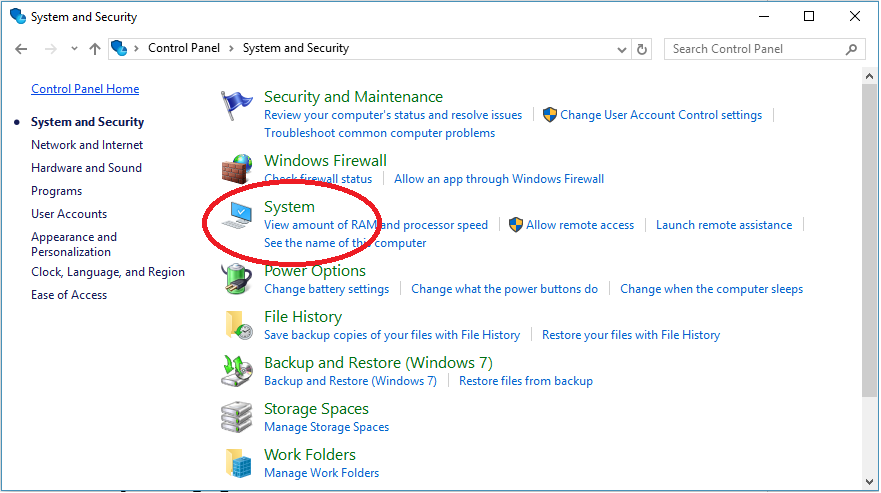
1. This virtual machine needs to be started in order to use Docker. When you first install Docker and run the “Docker Quickstart Terminal”, the machine is started for you. However, if you restart your computer you will need to start the virtual machine. To do this, open a PowerShell window as Administrator and run docker-machine start, just like you did to restart the machine after allocating more memory.
2. After the Docker Quickstart Terminal” is has completed you will need to set up several following environment variables.
   1. To do this on Windows open the Start menu and type “Control Panel” in the search box, the click on Control Panel.



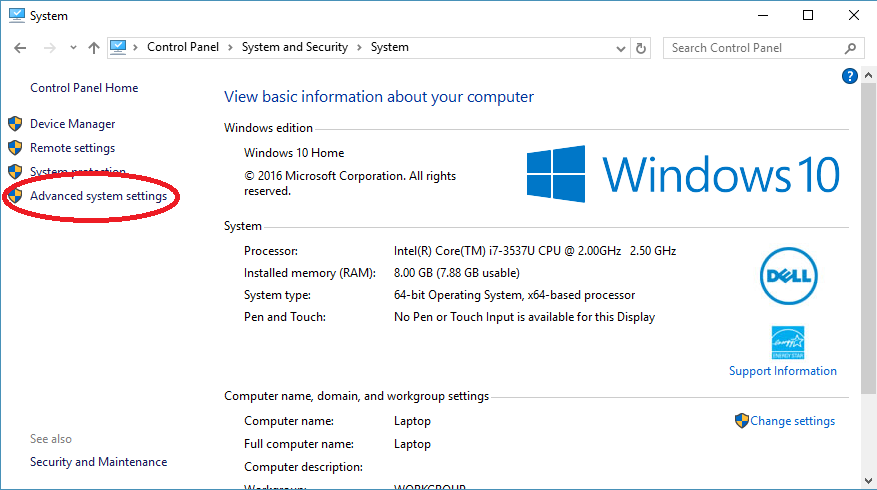
* 1. From the Control Panel, click on System and Security.



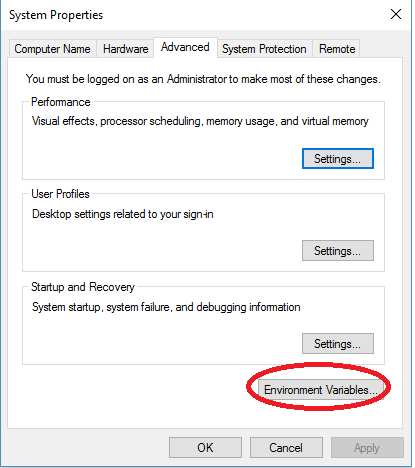
* 1. On the System and Security screen, click on System.



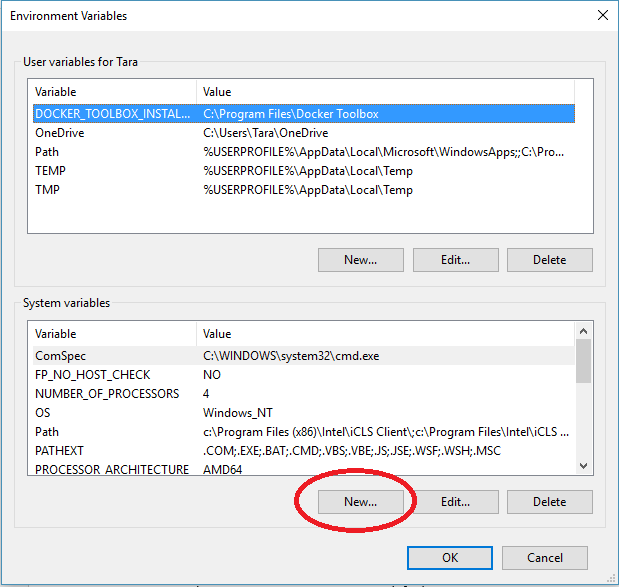
* 1. On the System screen, click on Advanced System Settings.



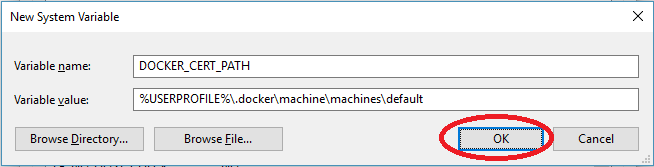
* 1. On the System Properties dialog, click the Environment Variables button.



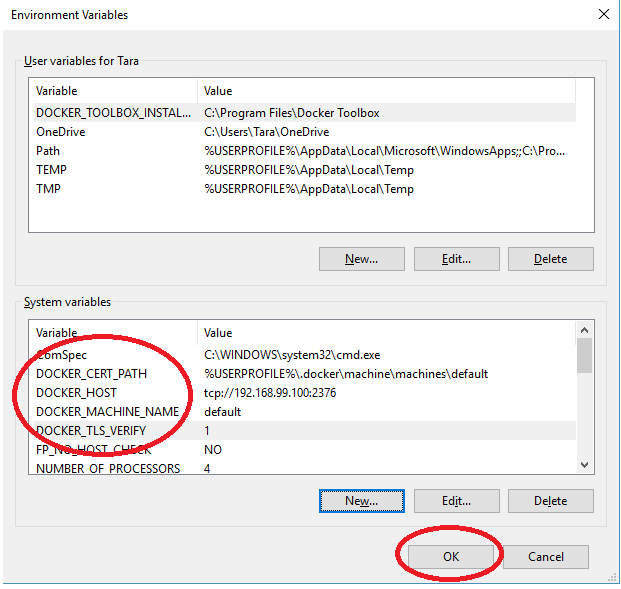
* 1. On the Environment Variables dialog, under the System variables heading, click the New… button.



* 1. In the New System Variable dialog, create a variable with a variable name of DOCKER\_CERT\_PATH and a variable value of %USERPROFILE%\.docker\machine\machines\default and click the OK button.



* 1. Create three more system environment variables with the following names and values:
     1. DOCKER\_HOST=tcp://192.168.99.100:2376
     2. DOCKER\_MACHINE\_NAME=default
     3. DOCKER\_TLS\_VERIFY=1
  2. Make sure your variables look like the screen shot below and then click the OK button.



* 1. Open a PowerShell window as Administrator and restart the Docker machine using these two commands:

docker-machine stop

docker-machine start

1. Checkout the local-server-setup project. For instructions on how to checkout a project see the section on [checking out the source code](#_Checking_Out_Source_1).
2. Open a PowerShell as administrator (or command prompt on Linux/Mac) and navigate to the location where you checked out the local-server-setup project, this will be in your workspace directory. For example C:\dev\workspaces\hmml\local-server-setup.
3. To bring up your local database, search and image server run the following:

docker-compose up

This will take a while the first time because it has to download all of the necessary software.

1. Once the server has started you can access the database using client database tools such as [MySQL Workbench](https://dev.mysql.com/downloads/workbench/5.2.html) using the following connection properties:
   1. Host name:
      1. On Windows: 192.168.99.100
      2. On Linux/Mac: localhost
   2. Port: 3306
   3. Username: jello
   4. Password: \*\*\*\*\*\*\*\*\*
   5. Default schema: broccoli
2. ElasticSearch can be accessed via the Marvel plugin which is automatically installed at:
   1. On Windows: <http://192.168.99.100:9200/_plugin/marvel/sense/index.html>
   2. On Linux/Mac: <http://localhost:9200/_plugin/marvel/sense/index.html>
3. The image server ([Digilib](http://digilib.sourceforge.net/image-directories.html)) is running on Tomcat at:
   1. On Windows: <http://192.168.99.100:8080>
   2. On Linux/Mac: <http://localhost:8080>
   3. The image server only has the images for Folio and GARZ 00005 in Reading Room

### Starting Docker

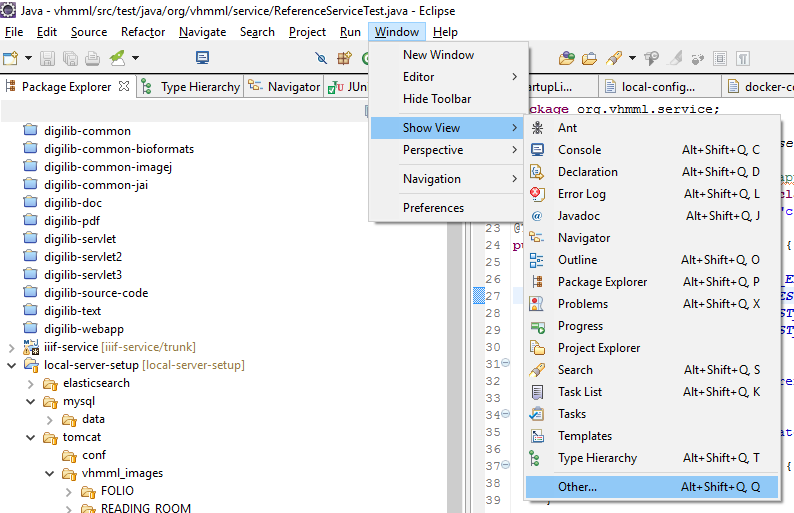
Once your Docker environment is set up, you will have to start the Docker virtual machine so that the database, search and image servers are available for vhmml to talk to. Please note that you don’t have the start Docker the very first time you install and configure it because it’s started as part of the set up process. However, you will have to start Docker whenever you restart your computer. This is a simple process that requires you to run two commands in a command window.

1. Open a command prompt (as administrator on Windows) and navigate to the hmml-dev docker directory
2. docker-machine start
3. docker-compose up

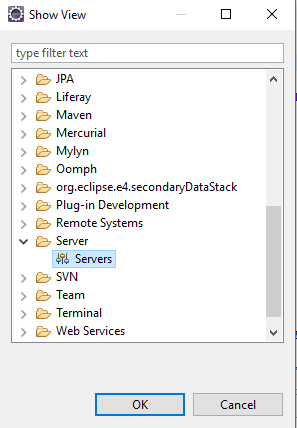
## Eclipse Server Setup

Now that you have a local database, search and image server, the next step is to set up a local Tomcat server in Eclipse where you can run and debug the vHMML and IIIF Service applications. At the time of this writing, Eclipse doesn’t support running an application on a remote server, so you need to download Tomcat and run it locally rather than just pointing to a Tomcat. The following steps explain how to do this.

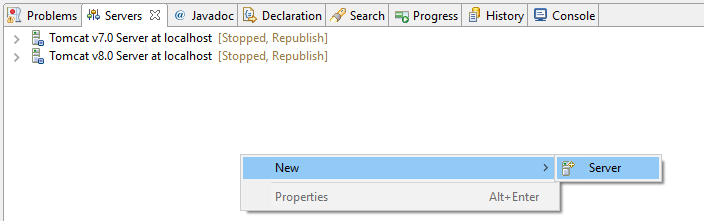
1. Download and install Tomcat. The vHMML application currently runs on Tomcat 7, downloads can be found [here](https://tomcat.apache.org/download-70.cgi). Look for the Binary distributions, download the zip file format and extract the zip file to a location of your choice.
2. After you have extracted the Tomcat zip file, you will need to put the MySQL database driver in Tomcat’s lib directory. The mysql-connector-java.jar can be downloaded [here](https://dev.mysql.com/downloads/connector/j/). The download is a compressed archive file (.tar.gz) that needs to be opened in a tool that can extract compressed files such as [7zip](http://www.7-zip.org/download.html). The only file you need from the compressed archive is the mysql-connector-java.jar file. Extract the mysql-connector-java.jar file from the downloaded zip file and copy it into the <TOMCAT\_INSTALL\_DIRECTORY>/lib, for example C:\dev\tomcat\apache-tomcat-7.0.73\lib.
3. In Eclipse, open the Show View dialog by selecting Window-> Show View -> Other…



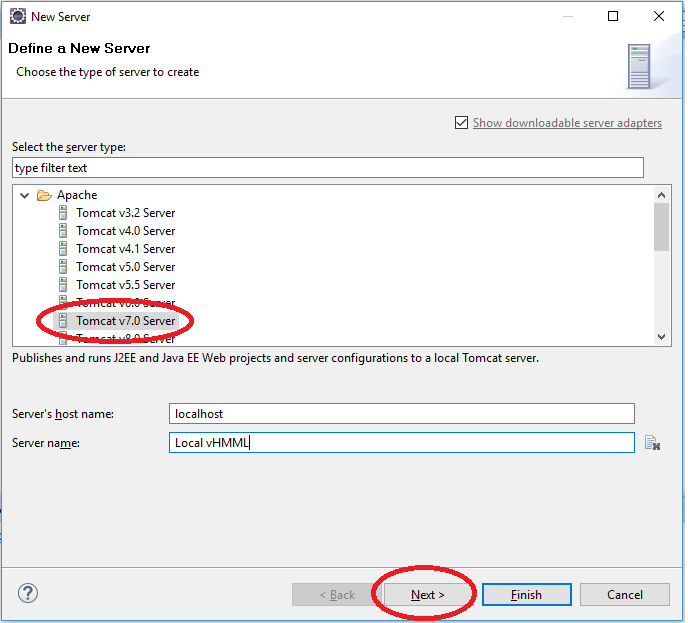
1. From the Show View dialog, select Server->Servers and click OK to open the Servers tab.



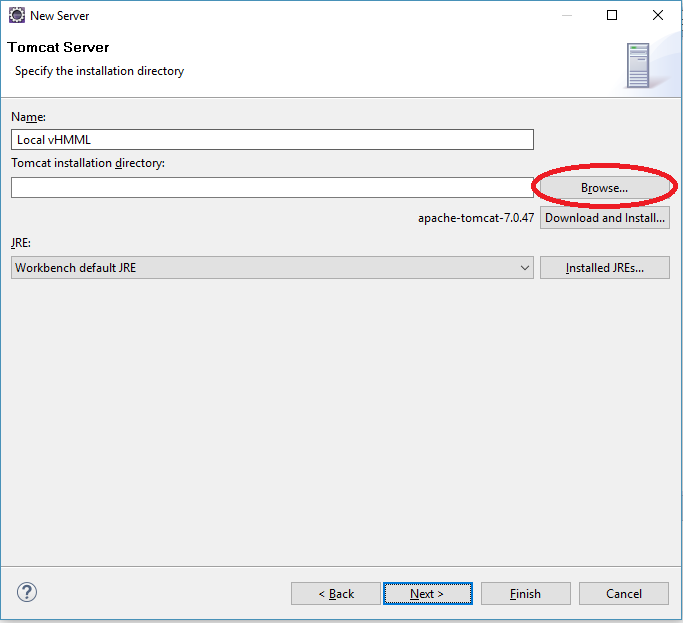
1. Right click anywhere in the Servers Tab, and select New->Server.



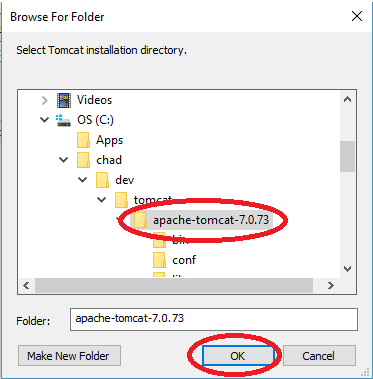
1. From the New Server dialog, open the Apache folder and select Tomcat v7.0 Server. Leave the Server’s host name field set to localhost. Optionally you may enter a Server name other than the default if you would like. When you are finished, click the Next button.



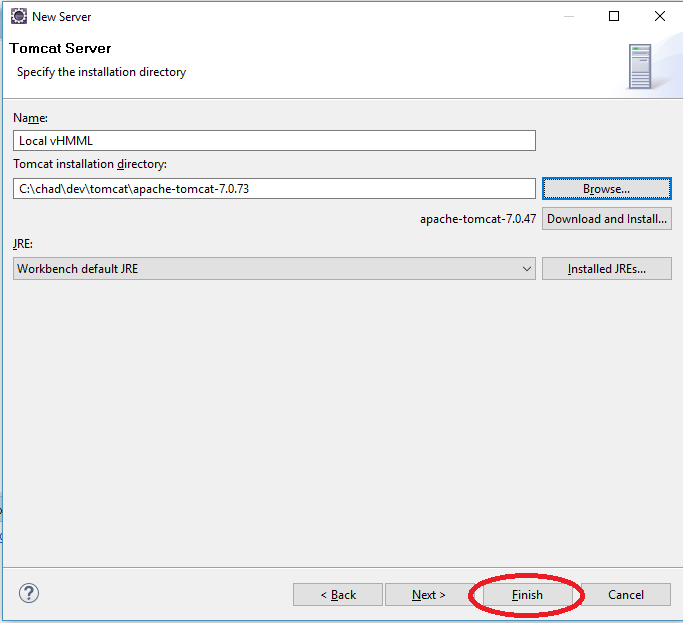
1. On the next screen, click the Browse button to open a browse dialog that will let you select the location of your Tomcat install.



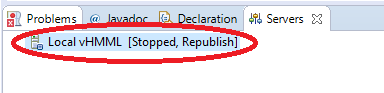
1. This will bring up a browse dialog where you can find and select the location of your Tomcat installation. Find the location where you extracted your downloaded Tomcat zip file and select the Tomcat folder and click the OK button.



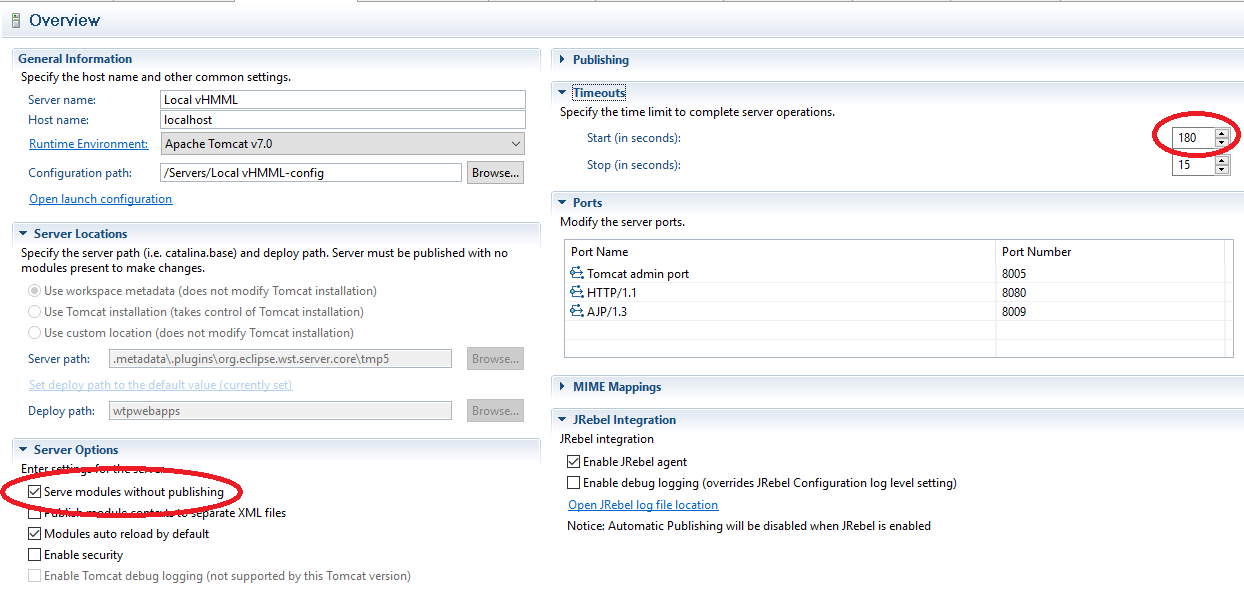
1. Click the Finish button on the New Server dialog.



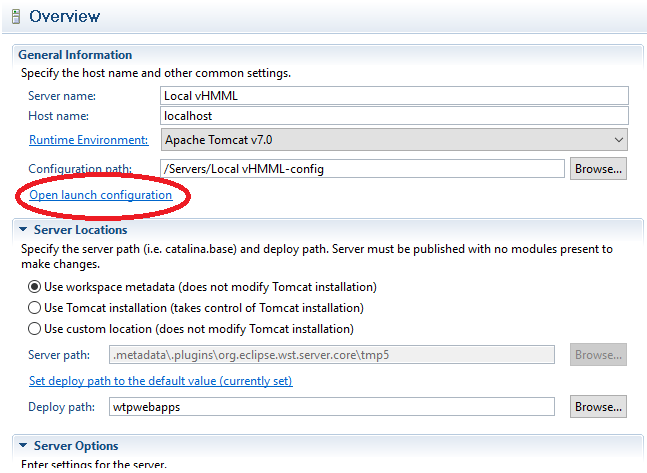
1. On the servers tab, double click on your new server to open the server configuration screen.



1. On the Server configuration screen, check the “Serve modules without publishing” checkbox and set the “Start (in seconds)” timeout to 180 seconds.

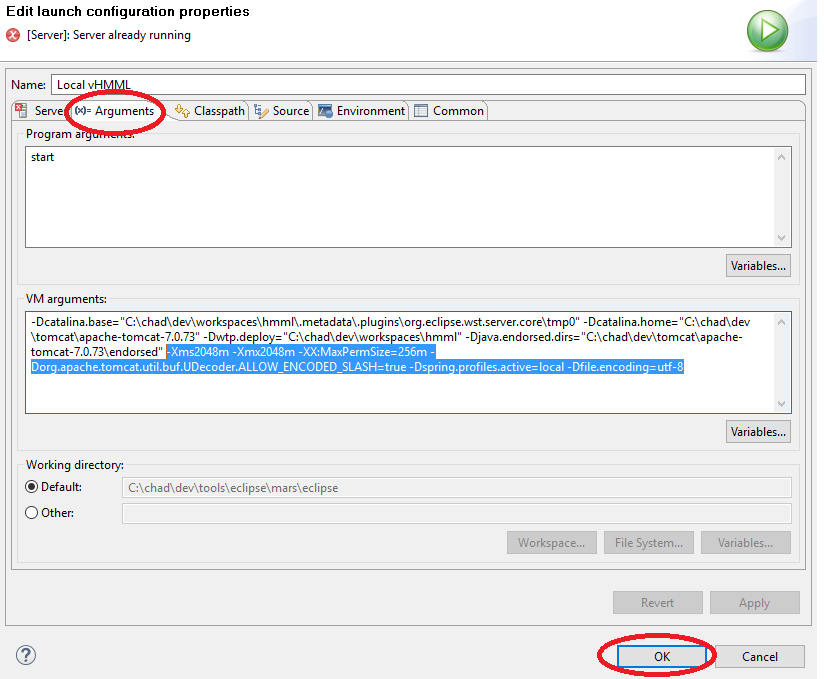


1. Click the “Open launch configuration” link to open the Edit Launch Configuration dialog.

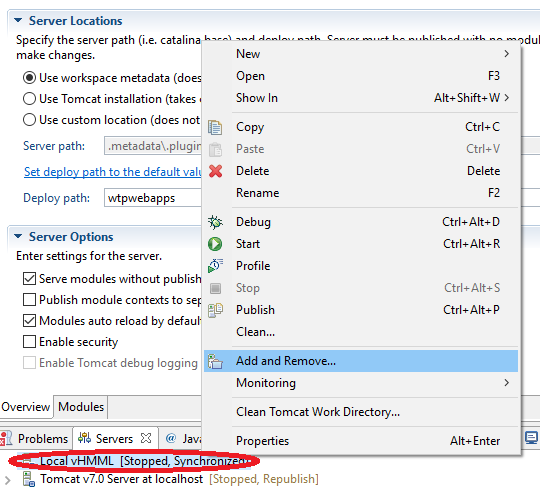


1. On the Edit Launch Configuration dialog, select the Arguments tab, and enter the following VM arguments at the end of the VM arguments that are already present, then click the OK button.

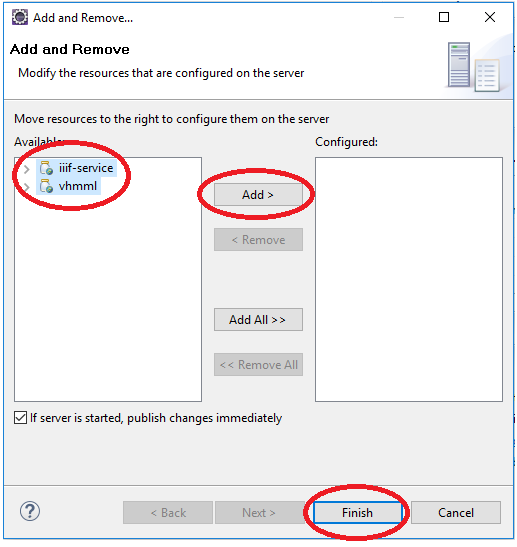
-Xms2048m -Xmx2048m -XX:MaxPermSize=256m -Dorg.apache.tomcat.util.buf.UDecoder.ALLOW\_ENCODED\_SLASH=true -Dspring.profiles.active=local -Dfile.encoding=utf-8



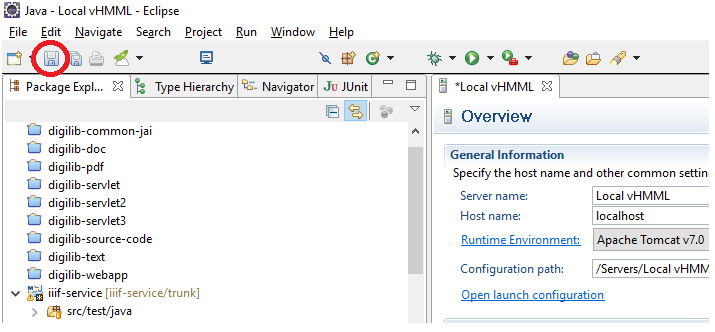
1. On the Servers tab, right click on the server and select “Add and Remove…”



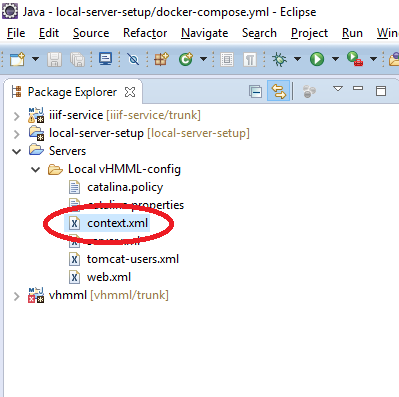
1. On the Add and Remove dialog, select the iiif-service and vhmml applications and use the Add button to add them to the server, then click the Finish button



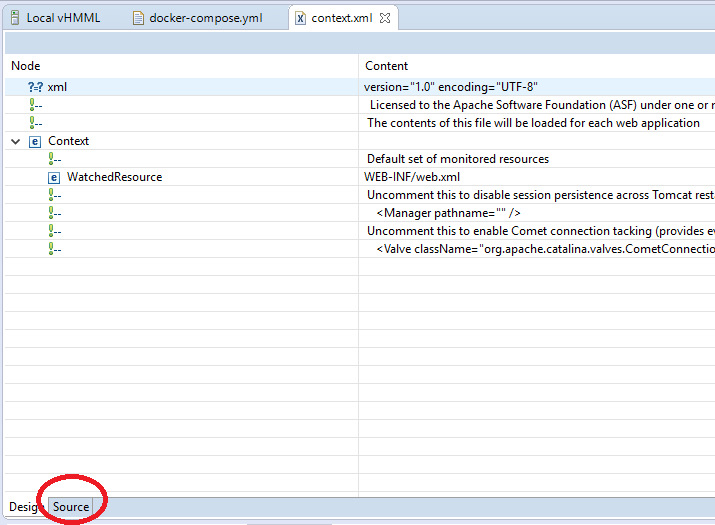
1. If you haven’t already, be sure to save any changes to your server by clicking the Save button or using the ctrl + s hot key.



1. In the Package Explorer panel under the Servers folder there will now be a Tomcat Folder with the same name as the server you just created. The folder contains some confirmation files. We need to edit these files to define the database connection that Tomcat will use to connect to the database. Open that folder and find the context.xml file and double click on it to open it.

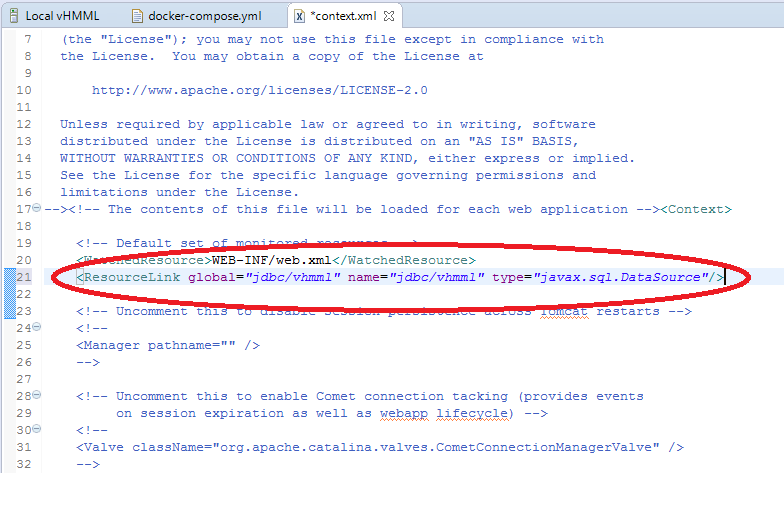


1. Click on the Source tab to view the source code of the context.xml file.

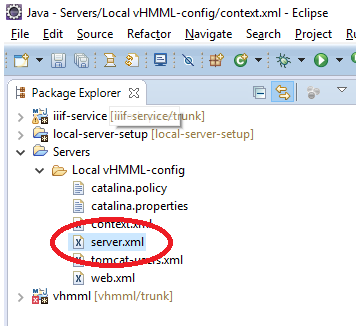


1. Enter the following right after the <WatchedResource> tag:

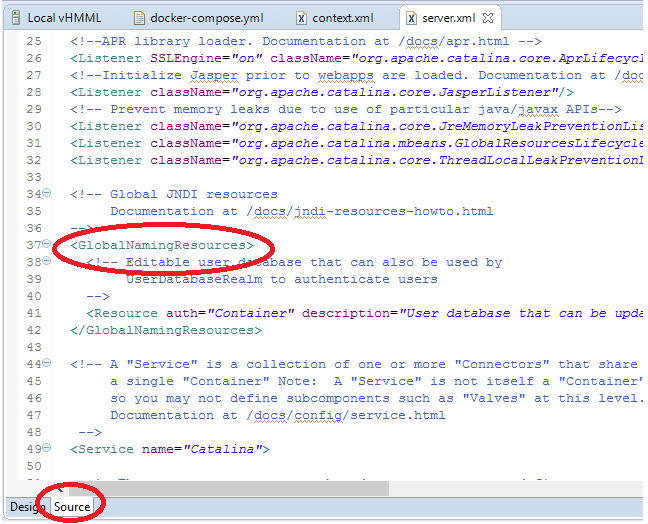
<ResourceLink global="jdbc/vhmml" name="jdbc/vhmml" type="javax.sql.DataSource"/>



1. Next, open the server.xml file that also resides in your Tomcat server folder.



1. Click on the Source tab to view the source code for the server.xml file and scroll down until you find the GlobalNamingResources tag.



1. Insert the following code between the GlobalNamingResources start and end tags:

<Resource

auth="Container"

connectionProperties="useUnicode=yes;characterEncoding=utf8;"

driverClassName="com.mysql.jdbc.Driver"

initialSize="5"

maxActive="120"

maxIdle="5"

maxWait="5000"

name="jdbc/vhmml"

username="vhmml"

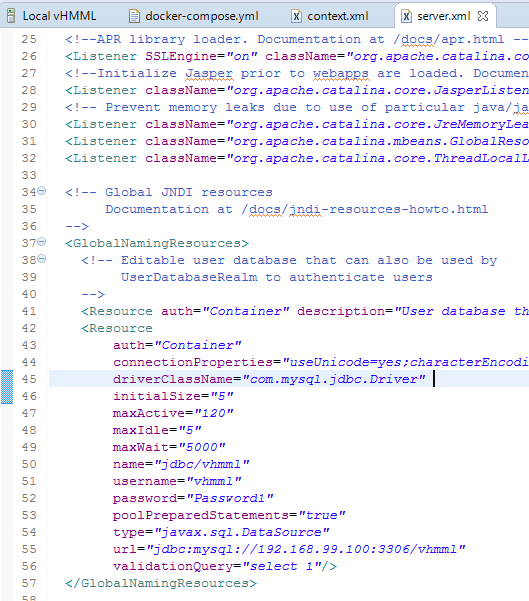
password="\*\*\*\*\*\*\*\*"

poolPreparedStatements="true"

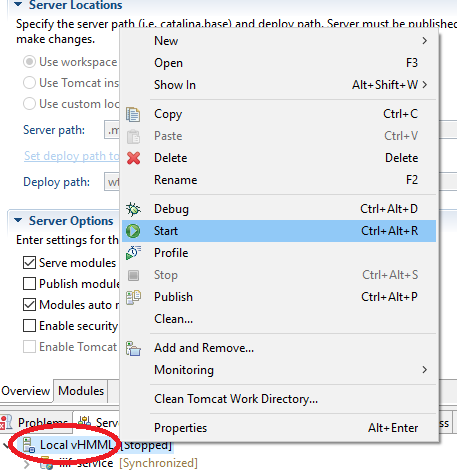
type="javax.sql.DataSource"

url="jdbc:mysql://192.168.99.100:3306/vhmml"

validationQuery="select 1"/>



1. Make sure you save the context.xml and server.xml files.
2. Start your new server by right clicking on it and selecting Start (or Debug if you would like to debug the vhmml or iiif-service applications).



1. Once the server finishes starting, you should be able to open the vHMML application in your browser at <http://localhost:8080/vhmml>.
2. The very first time you start your server, the vHMML application will only have one user named admin with the following password which you will be forced to change upon login:

\_\_\_\_\_\_\_\_\_\_\_

1. For security reasons the default admin user has no roles, someone with access the database has to grant the user roles. To grant this user the admin role execute the following SQL query using the database client of your choice (command line MySQL client, [MySQL Workbench](https://dev.mysql.com/downloads/workbench/5.2.html), etc.):

insert into user\_roles values(null, (select id from users where username = 'admin'), (select id from roles where name = 'ROLE\_ADMIN'));

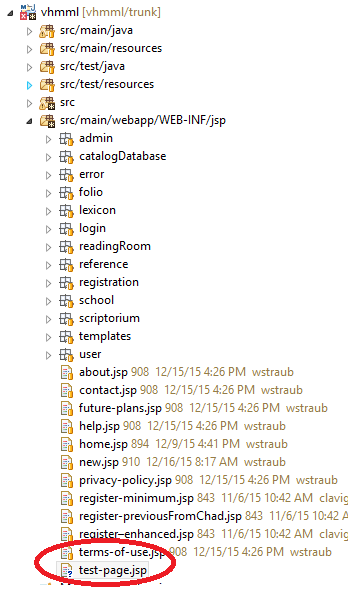
1. You will have to log out of vHMML and log back in for the role change to take effect.
2. After logging in as admin, be sure to navigate to the Administration screen and click the re-index button to initialize the search index.
3. You should now be able to use all of the features of vHMML locally.

# Development

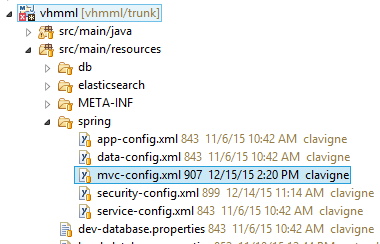
## Creating New Static Pages

Because we are using Apache Tiles for view layout and Spring MVC to intercept all requests to our web application, there is a little work to get Spring to serve a static JSP page without writing any Java code. This section explains the process for getting a static page working in vHMML. Please note that this process is for creating top level pages on the site, as opposed to pages that fall under a specific page category such as Lexicon or Reading Room. For information on adding a new page to a specific page category please see the section on [Adding New Pages to Existing Page Categories](#_Adding_New_Pages).

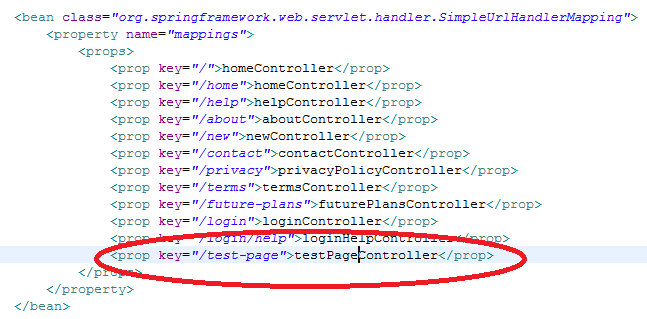
1. Place the page in the src/main/webapp/WEB-INF/jsp, for example the screen below shows a new page called test-page.jsp.



1. Open the src/main/resources/spring/mvc-config.xml config file, this is where you will define a controller and controller mapping for the page.

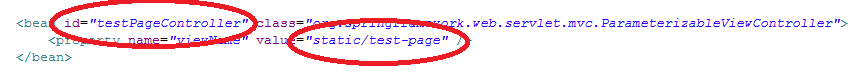


1. Add a controller mapping to the “mappings” property of the SimpleUrlHandlerMapping bean. This is the URL at which the page will be made available.



This configuration is essentially saying “when someone requests the URL [www.vhmml.org/test-page](http://www.vhmml.org/test-page), the testPageController will handle the request”. We define the testPageController in the next step.

1. Next add a controller bean definition to mvc-config.xml file. There are several examples you can copy already in the file. You just need to copy one of the examples and change the “id” attribute and “value” attribute of the “viewName” property. Please note that we have implemented a naming convention that must be followed for the viewName attribute’s “value” property. The convention is that the viewName must be “static/pagename” where “pagename” is the name of your new page without the file extension. For example, if your new page is test-page.jsp, then the value of the viewName property is “static/test-page”. This naming convention allows the Apache Tiles view to be automatically resolved without having to define a Tiles view in the tiles-defs.xml file.



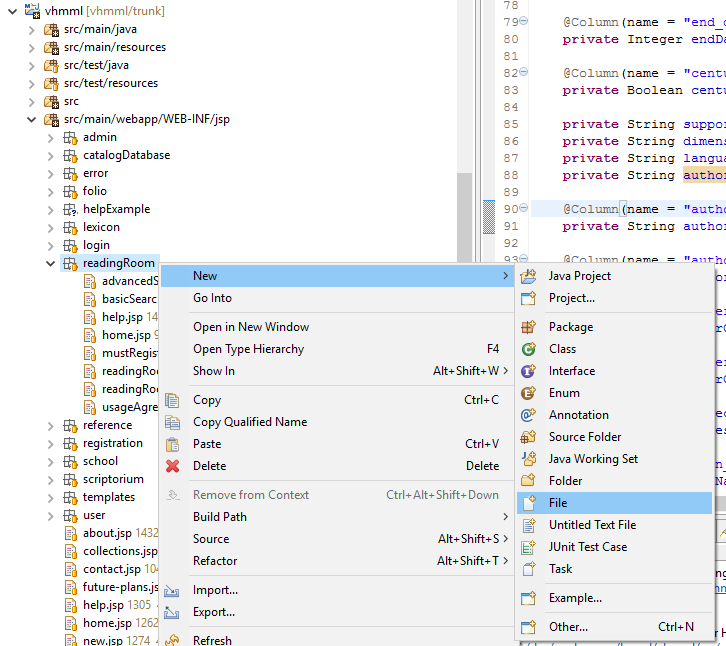
Your page should now be visible at <http://servername:port_number/vhmml/url_mapping>. For example, if you’re running the application on your local workstation on port 8080 the url for our test page would be <http://localhost:8080/vhmml/test-page>.

## Adding New Pages to Existing Page Categories

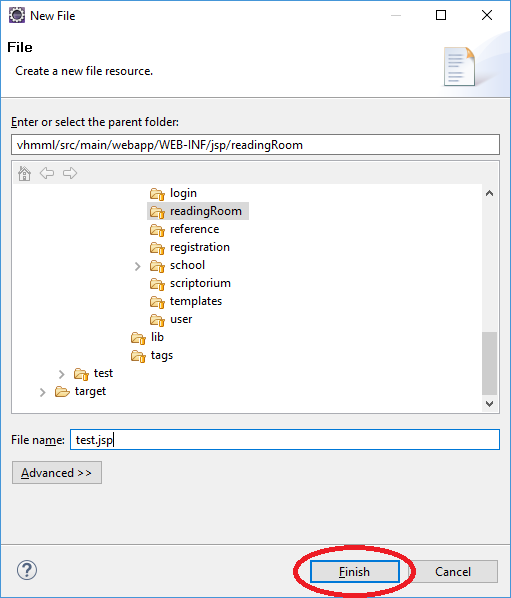
The following steps describe how to add a new page to an existing page category such as Lexicon or Reading Room.

### Create the page

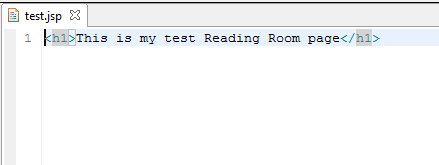
1. In Eclipse, right click on the folder that contains the pages for the page category to which you would like to add a page. The page category folders are located in /src/main/webapp/WEB-INF/jsp. For example, if you would like to add a page to Reading Room, right click on /src/main/webapp/WEB-INF/jsp/readingRoom. From the context menu, select New->File.



1. Enter a name for your page in the New File dialog and click Finish.



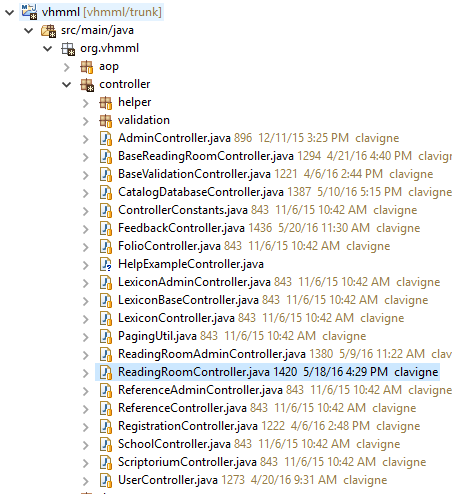
1. Create the content for your page.



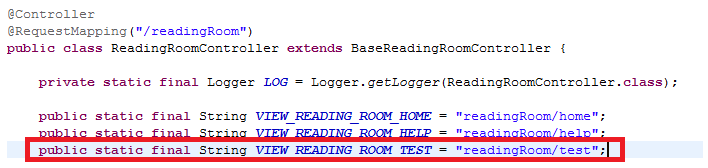
### Add the Controller Method

Next you will have to add a new Spring controller method. A Spring controller is a Java class that is responsible for intercepting HTTP requests. Each controller has methods (i.e. functions) for processing requests. Each page category in vHMML has a controller that handles requests for that page category. For example, the LexiconController has methods for processing things like a lexicon search request or a request to view a lexicon definition. When you’re adding a new page to a category, you need to make a method that will map a URL request to your new page.

1. In Eclipse, open the controller class responsible for servicing requests to the page category you are adding to. For example, if we are adding a page to Reading Room, the controller class is ReadingRoomController. All controller classes are in the /src/main/java/org/vhmml/controller directory.

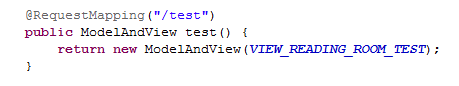


1. Add a view name constant for your new page; this is just a name for your page. The reason we create the constant is so that we can reference the constant from any methods that send the user to this specific page. This is useful because multiple requests might send the user to the same page. For example, we have several actions the user can take that would take them back to the Reading Room home page. Having the name in one place helps to ease maintenance.



This value for the constant directly correlates with the page we just created. In our example, we create a page called “test.jsp” in the “readingRoom” directory. Spring is configured to know that all pages are in the /src/main/webapp/WEB-INF/jsp directory, so this value must be relative to that directory. Spring also automatically appends “.jsp” to the end of the page name.

1. The final step is to create a controller method that maps a URL to the page.



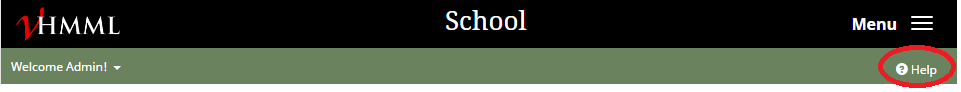
All we are doing here is using the @RequestMapping annotation to map the URL “/test” to our new page. Note that the full URL will be http://<host>:<port>/<contextPath>/readingRoom/test, for example, <http://localhost:8080/vhmml/readingRoom/test>. If you look at the top of the class definition you’ll notice that the @RequestMapping annotation is also used there. That annotation maps all methods in the entire class to /readingRoom, which is why we don’t have to specify /readingRoom/test on our method. This is the simplest example of a controller method. There are many configuration options for the @RequestMapping annotation. For example, the following configuration would map /test and /test/ to our method and would only allow GET requests:



For more information see the [Spring MVC documentation](http://docs.spring.io/autorepo/docs/spring/4.2.4.RELEASE/spring-framework-reference/html/mvc.html). You should now be able to view your page in your web browser using the URL you specified.

## Creating Help Pages

The vHMML application has a mechanism for providing context-sensitive help pages, meaning help pages that are specific to where the user is in the application. For example, if the user is looking at one of the lessons in the School portion of the application and they click on the help icon, we show them a help page that is specific to School. Users always access help from the same help icon which is located on the navigation bar right below the main menu button:



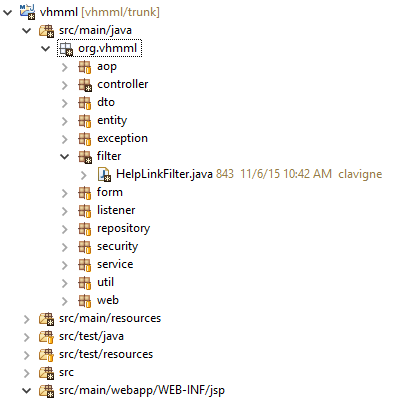
This means the help link must be dynamic, in other words the help link is always in the same place but the page it links to changes. This functionality is provided by a Java class called the HelpLinkFilter. This is a custom Java servlet filter that intercepts every request and assembles the help link based on the user’s current request. The mechanism used to determine the help link URL is fairly simple. The HelpLinkFilter has a list of page categories that are mapped to specific help pages. For example, if the user has just navigated to <http://yoursite/school/lesson/basics-paleography/overview>, the HelpLinkFilter will parse the current request and see that the user is navigating to School because the first part of the URL is /school. The following section describes how to add a help page to a specific page category.

### Create the Page & Controller Method

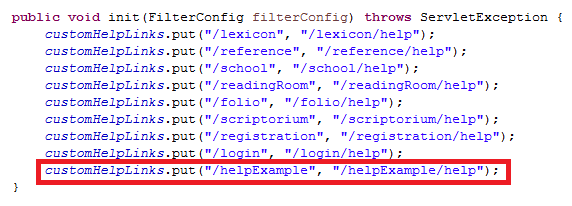
To add a new help page, first create the page and corresponding controller method by following the instructions in the [Adding New Pages to Existing Page Categories](#_Adding_New_Pages) section.

### Update the HelpLinkFilter

1. In Eclipse, open the HelpLinkFilter java class. It is in the /src/main/java/org/vhmml/filter directory.



1. Scroll down to the init method and add a url-to-page mapping for your help page. For example, in the following code we have added a new entry that maps the URL “/helpExample” to the page located at “/helpExample/help”. The first URL is the URL that will appear in the help link, the second URL is the URL that actually points to the help page, i.e. the URL that you specified in the @RequestMapping annotation of your controller method.



1. Restart Tomcat and the help link should now point to your new help page.

## Elastic Search

### Synonyms

We use synonym files in Elastic Search to have searches for different words that mean the same thing return the same result. For example, a search for “Lebanon” and “Liban” are considered the same. These are simple comma delimited files where each line represents words that mean the same things. For example, the Lebanon/Liban example would look like this:

lebanon, liban

To make another word mean the same thing we would just add a comma after liban followed by the new synonym, like this:

lebanon, liban,new country

We currently have 2 synonym files, british\_english\_synonyms.txt and vhmml\_synonyms.txt. The first file contains a list of British to English synonyms that will not change much. The second list contains synonyms that are specific to vHMML such as the lebanon/right arrow liban example, if you need to add synonyms this is most likely the file you would want to add them to. The synonym files are located in:

**Production Search Server** /etc/elasticsearch/

If you update a synonym file you must re-index Elastic Search for changes to take effect.

# Database Reference

## Import/Export

It is often useful or necessary to export the data from the database. For example, it is best practice to export the data from the production database before migrating a build to production so that the database can be restored to its former state if the build needs to be rolled back.

### Data Export

For data export, the [mysqldump](https://dev.mysql.com/doc/refman/5.5/en/mysqldump.html) command can be used to export all data from a MySQL database to a file. If you’re going to import the data on the same server that you’re exporting it from, then you can simply follow the [basic export](#_Basic_Export) instructions below. If you’re going to import the data on a different server the command is slightly different, be sure to see the [Exporting to Different Servers](#_Exporting_to_Different) section.

#### Basic Export

mysqldump –u<user\_name> -p <schema\_name> > filename

The values for “user\_name” and “schema\_name” need to be replaced with the username you use to connect to the database, the filename can be any valid filename. For example, the following command will connect using the username “jello” to the “broccoli” schema and output the data to a file named backup.sql.

mysqldump -jello –p broccoli> backup.sql

After entering this command you will be prompted to enter a password for the database user you are connecting with. Please note that this example assumes you are connecting to a database that is running on the machine on which you are running the command. To export data from a database running remotely, the following command can be used:

mysqldump –h<host\_name> –u<user\_name> -p <schema\_name> > backup.sql

Replace host\_name with the IP Address or hostname of the machine where MySQL is running. For example, the following command will connect to MySQL running on a server named yourdb using the username “jello” to the “broccoli” schema and output the data to a file named backup.sql.

mysqldump -hyourdb -jello -p broccoli > backup.sql

The mysqldump command has many other options. For example, you can specify the port number if your database is running on a non-standard port. You can also enter the password as part of the command, allowing you use the database export in a shell script. For more information see the [MySQL documentation](https://dev.mysql.com/doc/refman/5.5/en/mysqldump.html).

#### Exporting to Different Servers

When exporting data that will be imported on a different server, the following clause needs to be added right after the schema name in the mysqldump command:

| grep -v 'SQL SECURITY DEFINER'

For example:

mysqldump -hyourdb –ujello -p broccoli | grep -v 'SQL SECURITY DEFINER' > backup-2017mmdd.sql

This magical incantation strips the “definer” clause from the CREATE VIEW statements in the dump file. A “definer” in MySQL terms is the user that created a view or procedure. These users are qualified with the host name of the machine, for example jello@yourdb. Since that user most likely doesn’t exist on the server where the data is going to be imported, the import will fail with errors. The mysqldump command doesn’t provide an option to suppress outputting the definer clause, so the above grep command works around this by stripping the definer from the mysqldump output before writing it to the dump file.

#### Exporting Permanent Links

The following query can be used to get a list of permanent links for a collection and output them into a CSV file that can be opened in a spreadsheet program. To specify the collection just update the hmml\_project\_number value in the where clause, ZMT in the example below. To specify the output location and filename, change the OUTFILE parameter, C:/dev/clients/vhmml/zmt-links.csv in the example below.

SELECT

hmml\_project\_number,

CONCAT('https://yourdomain/vhmml/readingRoom/view/', id)

FROM

reading\_room\_objects

WHERE

hmml\_project\_number LIKE 'ZMT%'

ORDER BY hmml\_project\_number

INTO OUTFILE 'C:/dev/clients/vhmml/zmt-links.csv'

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n';

### Data Import

When importing data from a mysqldump file, you will most likely need to first clean out the database you are importing data into to avoid duplicate key errors, etc. The basic steps are:

1. Drop the database
   1. drop database vhmml;
2. Re-create the database
   1. create database vhmml;
3. Open a command prompt
4. Navigate to the directory containing the mysql dump file
5. Import the data using the following mysql command:

IMPORTANT: This command has to be slightly different on Windows Systems:

**Windows Systems**:

&cmd.exe /c "mysql -h<host\_name> -u<user\_name> -p <schema\_name> < <export\_filename>"

For example:

&cmd.exe /c "mysql -hyourdb -ujello -p broccoli < backup-12-1-2016.sql"

Note the quotes around the command.

**Linux & Mac Systems**:

mysql –h<host\_name> -u<user\_name> -p <schema\_name> < <export\_filename>

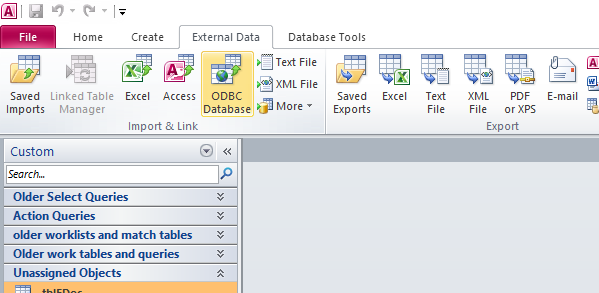
For example:

mysql -hyourdb -ujello -p broccoli < backup-12-1-2016.sql

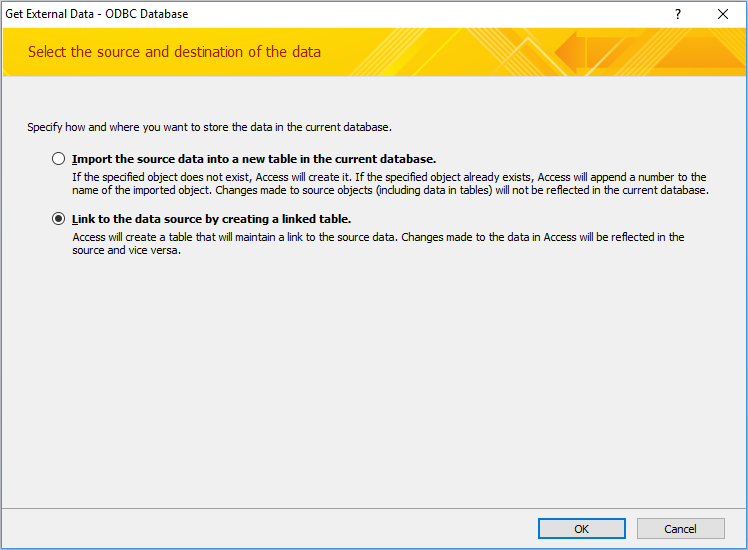
If you are importing the data on the same machine where you are running the command, the –h parameter (host) can be omitted.

## Connecting to MySQL with MS Access

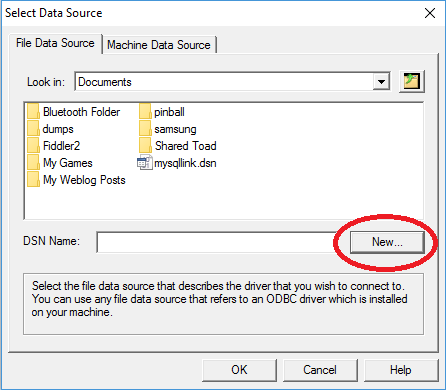
1. In Microsoft Access, click on the External Data tab and click the ODBC Database button.



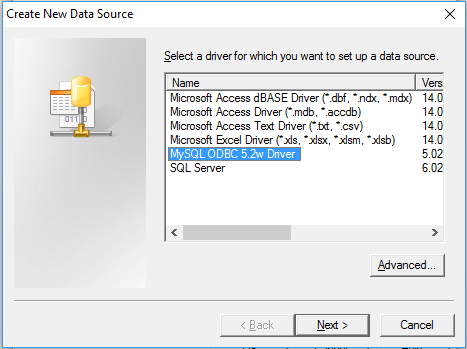
1. In the Get External Data dialog, select the Link to data source by creating a linked table radio button and click OK.



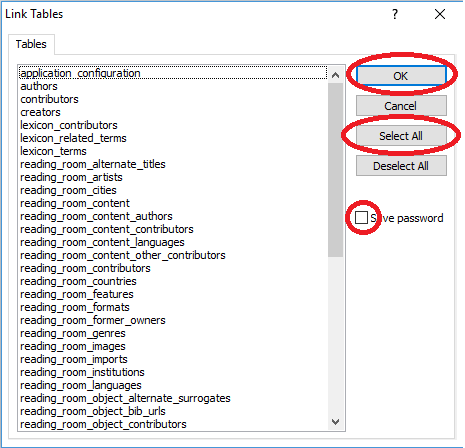
1. On the Select Data Source dialog, click the New button.



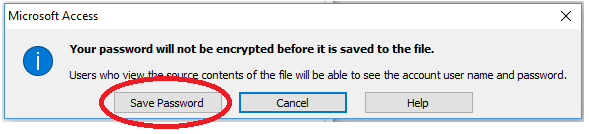
1. On the Create New Data Source dialog, select the MySQL ODBC Driver option and click Next.



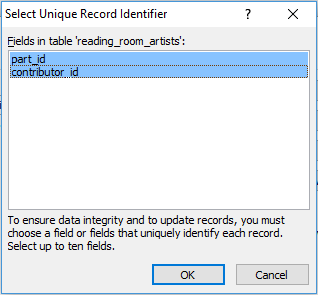
1. On the next screen, enter a name for the connection and click Next. This can be any name you choose, it is just the name of the file that will be created by MS Access to save the connection information.
2. Click Finish to bring up the MySQL Connector/ODBC Data Source Configuration dialog.
3. On the MySQL Connector/ODBC Data Source Configuration dialog, enter the MySQL host, username, password and database name (schema) for the MySQL database you are going to connect to. Click the Test button to make sure your connection information is correct. Once you have confirmed successful connectivity, click OK.
4. This will bring up the Link Tables dialog. You can select individual tables if you’re only working with a few tables or just click the Select All button if you would like to link all tables in the MySQL schema to MS Access. You may also wish to check the Save Password checkbox so you don’t have to authenticate every time you connect. Please note that the Save Password feature will store your password unencrypted in the MS Access connection file. If this is a security concern, then don’t check the Save Password checkbox. If it’s a local workstation and you’re the only one who has access to the file system on the computer, then it’s probably not a concern. When you are finished click OK.



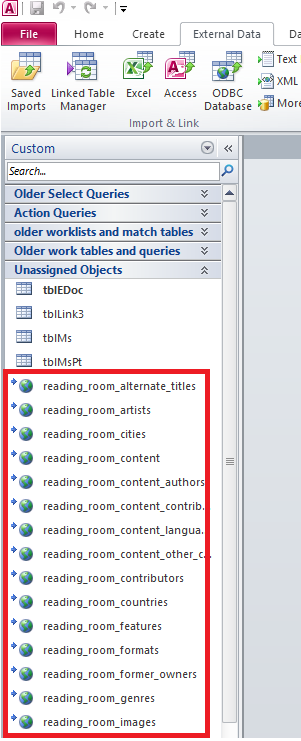
1. If you chose to save your password you will receive a warning dialog telling you that your password is going to be stored unencrypted in the MS Access connection file. Click Save Password to continue, you will have to do this for every table. Please note that if you click Cancel you will have to start the entire process over.



1. If there are any tables or views that don’t have a defined primary key, you will be prompted with the Select Unique Record Identifier dialog to select a column or columns to unique identify records. For these tables/views, select the column or columns that would uniquely identify a record and click OK. For example the reading\_room\_artists view doesn’t have a primary key because it’s not a table, it’s a view (a.k.a. saved query). So we select both the part\_id and contributor\_id fields since the combination of those two fields would always be unique.



1. When you are finished completing all Save Password and Unique Identifier record dialogs, you will see your linked tables in the left column in MS Access. You can click on any table and see the data in MS Access.



# Appendix

## Useful Linux Commands

|  |  |
| --- | --- |
| Command Example | Description |
| df -m | Show the amount of free disk space in megabytes |
| free -m | Show the amount of free memory in megabytes |
| top | View current resource usage |
| ssh username@host | Log in to machine “host” as user “username” |
| scp filename username@host:/path/to/copyto | Copy file “filename” to directory “/path/to/copyto” on machine “host” as user “username” |
| ps -ef | grep java | Show java process information (ID, JVM args. etc.) |
| netstat -an | grep :8080 | wc -l | Show the number of connections on port 8080 |

## Database Table Reference

### General Application Tables

|  |  |
| --- | --- |
| Table | Description |
| application\_configuration | Configuration properties that can be set via the admin screen in the application |
| users | List of users in the system |
| roles | List of roles in the system |
| user\_roles | Table used to assign specific roles to users |
| schema\_version | Table containing version information about all schema updates, created and updated by [Flyway database schema management tool](https://flywaydb.org/) during the build process |

### Lexicon Tables

|  |  |
| --- | --- |
| Table | Description |
| contributors | List of people that contribute to Lexicon terms |
| lexicon\_contributors | Table that links a contributor to a specific lexicon term |
| lexicon\_terms | List of lexicon terms |
| lexicon\_related\_terms | Table that links one lexicon term to another |

### Reference Tables

|  |  |
| --- | --- |
| Table | Description |
| creators | List of reference entry creators, e.g. author, editor, etc. |
| reference\_creators | Table that links a creator to a specific reference entry |
| reference\_entries | List of reference entries |
| tags | List of tags used to tag reference entries |
| reference\_tags | Table that links a tag to a specific reference entry |
| reference\_list (view) |  |

### Reading Room Tables

|  |  |
| --- | --- |
| Table | Description |
| reading\_room\_objects | List of Reading Room objects |
| reading\_room\_object\_parts | List of Reading Room object parts, linked to parent object by reading\_room\_object\_id column |
| reading\_room\_content | List of Reading Room content items, linked to parent part by reading\_room\_object\_part\_id column |
| reading\_room\_images | List of Reading Room images (currently unused, imported from Oliver), linked to parent content by reading\_room\_content\_id column |
| reading\_room\_countries | List of countries used in Reading Room, a country is linked to a Reading Room object by the country\_id column on the reading\_room\_objects table |
| reading\_room\_cities | List of cities used in Reading Room, a city is linked to a Reading Room object by the city\_id column on the reading\_room\_objects table |
| reading\_room\_repositories | List of repositories used in Reading Room, a repository is linked to a Reading Room object by the repository\_id column on the reading\_room\_objects table |
| reading\_room\_institutions | List of holding institutions used in Reading Room, an institution is linked to a Reading Room object by the holding\_institution\_id column on the reading\_room\_objects table |
| reading\_room\_formats | List of formats used in Reading Room, a format is linked to a Reading Room object by the surrogate\_format\_id column on the reading\_room\_objects table |
| reading\_room\_object\_alternate\_surrogates | Table that links an alternate surrogate to a specific reading room object |
| reading\_room\_object\_bib\_urls | Table that links an external bibliography URL to a specific reading room object |
| reading\_room\_object\_contributors | Table that links a contributor to a specific reading room object |
| reading\_room\_object\_extents | Table that links an extent to a specific reading room object |
| reading\_room\_features | List of features used in Reading Room |
| reading\_room\_object\_features | Table that links a feature to a specific reading room object |
| reading\_room\_genres | List of genre/form terms used in Reading Room |
| reading\_room\_object\_genres | Table that links a genre/form to a specific reading room object |
| reading\_room\_subjects | List of subjects used in Reading Room |
| reading\_room\_object\_subjects | Table that links a subject to a specific reading room object |
| reading\_room\_part\_centuries | Table that links a century to a specific part |
| reading\_room\_part\_contributors | Table that links a contributor to a specific part |
| reading\_room\_alternate\_titles | Table that links an alternate title to a piece of content |
| reading\_room\_content\_contributors | Table that links a contributor to a specific content item |
| reading\_room\_languages | List of languages used in Reading Room |
| reading\_room\_content\_languages | Table that links a language to a specific content item |
| reading\_room\_contributors | List of Reading Room contributors |
| reading\_room\_imports | List of imports that have been run |
| reading\_room\_scripts | List of scripts used in Reading Room |
| reading\_room\_uniform\_titles | List of uniform titles, linked to specific piece of content by the uniform\_title\_id column on the reading\_room\_content table |
| reading\_room\_writing\_systems | List of writing systems used in Reading Room |
| reading\_room\_overview (view) | View used to aggregate overview data for search indexing and meta data panel |
| reading\_room\_artists (view) | View used to aggregate artist contributors, used by overview view to get artist list |
| reading\_room\_content\_authors (view) | View used to aggregate author contributors, used by overview view to get author list |
| reading\_room\_content\_other\_contributors (view) | View used to aggregate other contributors, used by overview view to get other contributor list |
| reading\_room\_former\_owners (view) | View used to aggregate former owner contributors, used by overview view to get former owner list |
| reading\_room\_scribes – (view) | View used to aggregate scribe contributors, used by overview view to get scribe list |

## Useful Queries

The following are a list of useful queries to help you get started working with the vHMML database and hopefully provide examples that can be used to create other queries to suit your needs.

### Show RR Objects in Collection without thumbnail selected

select hmml\_project\_number from reading\_room\_objects where icon\_name is null and hmml\_project\_number like 'GARZ%';

### Grant role to user by username

insert into user\_roles values(null, (select id from users where username = 'herman'), (select id from roles where name = 'ROLE\_LIBRARY\_CREATOR'));

### Show roles for user

select r.name from users u left join user\_roles ur on u.id = ur.user\_id left join roles r on ur.role\_id = r.id where u.username = 'admin';

### Create country, city, repository, etc.

insert into reading\_room\_countries (name, authority\_uri\_lc, authority\_uri\_viaf, name\_iso\_3166) values('Mexico', 'http://lc.gov/12345', 'http://viaf.org/12345', 'isoname');

### Update country, city, repository, etc.

update reading\_room\_countries set authority\_uri\_lc = 'the correct lc url goes here' where name = 'Mexico';

### Show Reading Room object by contributor

select

     o.hmml\_project\_number

from

     reading\_room\_objects o left outer join

     reading\_room\_object\_parts p on o.id = p.reading\_room\_object\_id left outer join  
     reading\_room\_content c on p.id = c.reading\_room\_object\_part\_id left outer join  
     reading\_room\_content\_contributors cc on c.id = cc.reading\_room\_content\_id left outer join  
     reading\_room\_contributors rc on cc.reading\_room\_contributor\_id = rc.id  
where

     rc.name = 'Al-Burlasī, Šahāb Al-Dīn Aḥmad famous as \'Umayrah, and his student Aḥmad Bin Qāsim Al-\'Ubādī';

### Show Reading Room object for a piece of content

select  
     o.\*  
from  
     reading\_room\_objects o left join  
    reading\_room\_object\_parts p on o.id = p.reading\_room\_object\_id left join  
    reading\_room\_content c on p.id = c.reading\_room\_object\_part\_id  
where c.id = 223075;

### Show Reading Room content by HMML Project Number

select

c.\*

from

reading\_room\_content c join

reading\_room\_object\_parts p on c.reading\_room\_object\_part\_id = p.id join

reading\_room\_objects o on p.reading\_room\_object\_id = o.id

where

o.hmml\_project\_number = 'GARZ 00007';

### Show Reference entry by id

select  
     re.title,  
     re.short\_title,  
     c.name,  
     c.first\_name,  
     c.last\_name  
from  
     reference\_entries re left join  
     reference\_creators rc on re.id = rc.reference\_entry\_id left join  
     creators c on rc.creator\_id = c.id  
where re.id = 13001;

### Show Lexicon entry by id

select  
     t.term,  
     c.name  
from  
     lexicon\_terms t left outer join  
     lexicon\_contributors lc on t.id = lc.lexicon\_id left outer join  
     contributors c on lc.contributor\_id = c.id  
where t.id = 14357;

### Delete Reading Room data

delete from reading\_room\_imports;

delete from reading\_room\_images;

delete from reading\_room\_content\_contributors;

delete from reading\_room\_content\_languages;

delete from reading\_room\_alternate\_titles;

delete from reading\_room\_content;

delete from reading\_room\_object\_alternate\_surrogates;

delete from reading\_room\_object\_bib\_urls;

delete from reading\_room\_object\_contributors;

delete from reading\_room\_object\_extents;

delete from reading\_room\_object\_features;

delete from reading\_room\_object\_genres;

delete from reading\_room\_object\_subjects;

delete from reading\_room\_part\_contributors;

delete from reading\_room\_object\_parts;

delete from reading\_room\_archival\_content\_centuries;

delete from reading\_room\_archival\_content;

delete from reading\_room\_archival\_data\_centuries;

delete from reading\_room\_archival\_contributors;

delete from reading\_room\_archival\_associated\_names;

delete from reading\_room\_archival\_data;

delete from reading\_room\_objects;

End of document