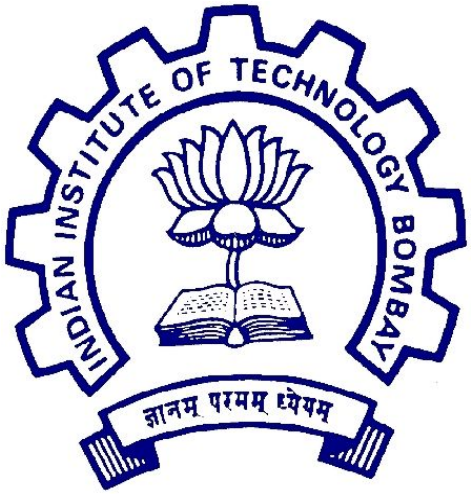


Summer Fellowship Report

On

Customization and Implementation of VIVO Open Source Software

Submitted by

Naveen Penta

Under the guidance of

Dr. Manju Naika

Chief Library Officer of Central Library

IIT Bombay

July 2, 2019

Acknowledgment

I wish to express our profound gratitude to our internship guide Dr. Manju Naika , Department of Library Science, IIT Bombay for his constant support and supervision throughout the internship.

I am highly indebted to my project mentor Mr. Pravin G and my project head Dr. Bella Tony for their continuous support, supervision, motivation and guidance throughout the tenure of my project in spite of their hectic schedule who truly remained driving spirit in my project and their experience gave me the light in handling this project and helped me in clarifying the abstract concepts,requiring knowledge and perception, handling critical situations and in understanding the objective of my work

Contents

Chapter 1

Introduction

VIVO is member-supported, open source software and an ontology for representing scholarship. VIVO supports recording, editing, searching, browsing, and visualizing scholarly activity. VIVO encourages showcasing the scholarly record, research discovery, expert finding, network analysis, and assessment of research impact. VIVO is easily extended to support additional domains of scholarly activity.

When installed and populated with researcher interests, activities, and accomplishments by an institution, VIVO enables the discovery of research and scholarship across disciplines at that institution and beyond. VIVO supports browsing and a search function which returns faceted results for rapid retrieval of desired information. Content in a VIVO installation may be maintained manually, brought into VIVO in automated ways from local systems of record, such as HR, grants, courses, and faculty activity databases, or from database providers such as publication aggregators and funding agencies.

VIVO creates an integrated record of the scholarly work of your organization.

VIVO creates a connected, integrated record of the scholarly work of your institution, ready for reporting, visualization, and analysis

Chapter 2

Installation of Vivo

2.1 Software Requirements:

Operating System: Linux
Java
MySQL
Apache-tomcat
Apache-Maven

1) Operating System :

Install Linux ubuntu 16.04 it is better to deploy the VIVO

2) JAVA :

Install java 8 SE version
(https://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html) and make sure that add path variables correctly . You have to add path in two files.

- IN home/.bashrc
  open this file through the gedit and enter the following commands:

  export PATH=$PATH:/usr/lib/jdk1.8.0_211
  enter above commands at the end of the file

- Add this path in /etc/environment file
  JAVA_HOME="/usr/lib/jdk1.8.0_211"

  execute this command $/etc/ source environment

3)MySQL :

Install mysql 8 from https://dev.mysql.com/downloads/repo/apt/
After Downloading this file execute following commands:

sudo dpkg -i mysql-8.0.deb
sudo apt-get update
sudo apt-get install mysql-server
set password as root

Create database with following commands:
$ mysql -u root -p

mysql> CREATE DATABASE vitrodb CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
mysql> CREATE USER 'vitrodbUsername'@'localhost' IDENTIFIED BY 'vitrodbPassword';
mysql> GRANT ALL PRIVILEGES ON vitrodb.* TO 'vitrodbUsername'@'localhost';

4) Apache Maven:

Install apache maven 3.6.1 http://maven.apache.org/download.html

Extract apache-maven.tar.gz at /opt directory rename with apache-maven

Add this path in /etc/environment file

M2_HOME="/opt/apache-maven"
MAVEN_HOME="/opt/apache-maven"

execute this command $/etc/ source environment

Edit according with your information at opt/apache-maven/conf/settings.xml

```
<settings>
        <proxies>
                <proxy>
                        <active>true</active>
                        <protocol>ftp</protocol>
                        <host>ip</host>
                        <port>8080</port>
                        <username>system username</username>
                        <password>system password</password>
                <nonProxyHosts>www.google.com*.somewhere.com</nonProxyHosts>
                </proxy>
        </proxies>
</settings>
```

5)Apache Tomcat :

Install apache tomcat 8 at http://tomcat.apache.org/download-80.cgi

Install apache at /usr/local with name tomcat it's better for VIV0

- IN home/.bashrc
  open this file through the gedit and enter the following command

  export CATALINA_HOME=/usr/local/tomcat

- Add this path in /etc/environment file

  CATALINA_HOME="/usr/local/tomcat"

  execute this command $/etc/ source environment

2.2 Installing VIVO through Git

Installation on vivo through github:

create vivo directory on /usr/local/
in that vivo folder run these commands:

$ git clone https://github.com/vivo-project/Vitro.git Vitro -b rel-1.10-maint
$ git clone https://github.com/vivo-project/VIVO.git VIVO -b rel-1.10-maint

Permissions:

- The maven user has write permission to the Tomcat webapps directory. Maven will fail silently if it cannot copy files to tomcat.
- The tomcat user has write permission to the <vivo-dir>/home directory.

Then go to the:
/usr/local/vivo$ cd VIVO
Run below command:
VIVO$ mvn install -s installer/example-settings.xml

Its show vivo success building.

Tomcat Configuration:

Goto the /usr/local/tomcat/conf/server.xml file

In every connecter tag add this : URIEncoding="UTF-8" address="127.0.0.1"

In host tag remove the hole host tag and add this

<Host name="localhost" appBase="/usr/local/tomcat/webapps"
unpackWARs="true" autoDeploy="true">
<ValveclassName="org.apache.catalina.valves.AccessLogValve"
directory="logs" prefix="localhost_access_log" suffix=".txt"
pattern="%h %l %u %t & quot;%r" %s %b" />

</Host>

Create setenv.sh file in tomcat's bin directory add this:

export CATALINA_OPTS="-Xms512m -Xmx512m -XX:MaxPermSize=128m"

Chapter 3

Running VIVO

3.1 Running tomcat

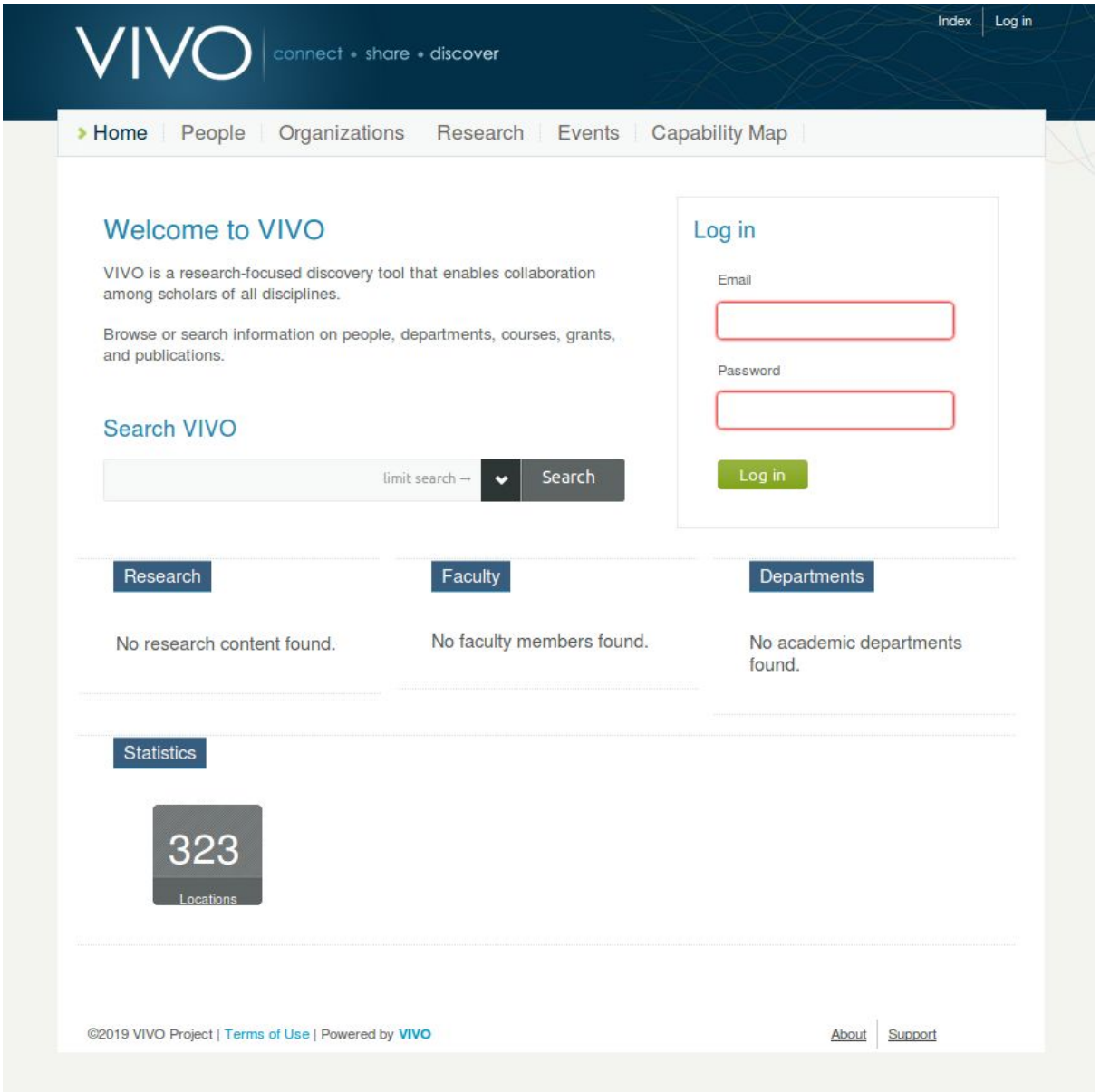
Run tomcat:

After doing this run tomcat with cd /usr/local/tomcat/bin/
$ sh startup.sh
after running the tomcat open the browser start vivo using localhost or IP

For deploying vivo it will take 20 to 30 minutes after this it will open vivo home page

3.2 VIVO Home page

On browser open link localhost:8080/vivo . It will take time for first time open Home page



login as root:

username: root_vivo@mydomain.edu

you can change this username by vivo/home/config/runtime.properties file

password: rootPassword

first time it will ask to change the password . We need to change that

Chapter 4

Customizing VIVO

4.1 Adding profiles:

Login as admin and click on site admin .
In data input dropdown box select profile.
Click on the Add individual of this class.
Next it will ask Your full Name and click on create.
The profile is created .
You can add more information to the profile.

4.2 Create, Assign and Use an Institutional Internal Class:

Create an Institutional Internal Class:

Create a local ontology if you do not already have one.
If you have one, add a class to your existing local ontology.
Go to Site Admin > Ontology list > Add new ontology

Ontology name:

Whatever you want. The name you give will appear in the list of VIVO ontologies.

Namespace:

Must be your domain name as specified in your runtime.properties, followed by "/ontology/" followed by a name of your choice, followed by the '#' sign

Namespace prefix:

a short word. This word will appear in the prefix list in your SPARQL windows and will be used by you in any SPARQL queries referring to your local ontology.

Submit Changes

Add a new class to your local ontology:

Go to 'Hierarchy of Classes Defined in This Namespace' > Add New Class

Class label:

Text, describes the class. This will be visible on the VIVO interface.

Class Group: People.

This allows the class to be used to restrict the display of people to those people who have the institutional class you are defining.

Ontology:

Select your previously created local ontology from the drop down menu

Internal Name:

This word will be used in your SPARQL queries, along with the local ontology prefix to refer to your local class. In this example, the complete reference in SPARQL would be.

Vlocal :MyEntity.

Short definition to display publicly:

Use language your users will understand

Example for ontology editors:

More detail. Can be very technical.

Description for ontology editors:

Will appear in the ontology editor. Remind future ontology editors of the purpose of the class and how one will know what entities should be in the class.

Display level:

Set to editor and above from the drop down

Update level:

Set to curator and Above from the drop down

Publish level:

All users including public

Assign your Institutional Internal Class:

Go to the person in the UI
Click Edit Individual
Click Add Type
Select your Institutional Internal Class from the drop down

Use your Institutional Internal Class:

Define institutional internal class

Go to Site Admin >Institutional internal class
Select your new class from the dropdown menu

To restrict the display to only those people in your institution,

Go to Site Admin > Page Management > People
Click the plus sign to expand the ‘Browse Class Group’ box

Check ‘Only display people within my institution'
Click "Save this Content'

4.3 Adding Site Information:

Using Site Admin→ Site Information, you can:

Set the site name :

(something like Scholars@Cornell).The site name is used on several pages when explaining the site to users.

Contact Email address:

provide an email address that will receive contact info from VIVO. Typically this is set to a generic email address or list that can be answered by the people responsible for user support. Example: info@vivo.myschool.edu

Set the theme: Select theme.

Copyright holder:

This is typically the name of your institution. Example: IIT Bombay

Copyright URL:

A URL that will be visited when a user clicks on the copyright link at the bottom of each VIVO page.

Site information is stored in the VIVO configuration triple store.

4.4 Theming:

Create Theme:

- VIVO comes with a standard theme, called wilma.wilma is in the folder in vivo/installer/webapp/target/vivo/themes.

- To create a new theme, choose a name for your new theme. In these examples below we will call the new theme fred.

- Copy the wilma directory and its contents to a new directory called fred. fred must also in vivo/installer/webapp/target/vivo/themes.

- Your new theme will contain CSS files, image files, and FreeMarker templates.

- Run the Maven install to deploy your new theme to the Tomcat container. Restart the VIVO Tomcat process. You can then go to the **Site Admin** page and choose **Site Information**, to select your theme as the current one.

Modify Header with Logo:

Now that you have a theme, we can modify the templates in the theme to change VIVO's look.Let's start with the header.You can replace VIVO's default logo ("VIVO: connect * share * discover") with your organization's logo. VIVO will look best if your logo has a height of 59px.

- Add your logo to fred/images
- Edit fred/css/wilma.css to replace the VIVO logo with your logo. Find h1.vivo-logo and change the URL of the background image to the name of your logo in images.
- Build VIVO with Maven and restart Tomcat.
- Check to see that your logo is in place

Modify Footer :

The footer for VIVO can be found in fred/templates/footer.ftl. You may want to change it's look,or merely change where VIVO sends people for Support (by default, VIVO send people to the VIVO web site– you may have a preferred location, or mail address for support.

- Edit fred/templates/footer.ftl to specify a URL for Support. Find menu_contactus and change the href in the anchor tag to point at the desired Support location– this could be a mailto:email address or a URL for your support web site.

- Build VIVO with Maven and restart Tomcat

- Check to see that your support link is in place

Chapter 5

Data Exporting

5.1 Convert CSV data to RDF using ingest tool:

Process:

1. Prepare CSV file
2. Create workspace models for ingesting and constructing data
3. Pull CSV file into RDF
4. Confirm data property URIs and RDF structure
5. Convert temporary RDF into VIVO/ISF ontology using SPARQL CONSTRUCT
6. Load data to the current web model

1: Prepare CSV file

CSV template files can be downloaded here (http://sourceforge.net/projects/vivo/files/Data%20Ingest/people.csv/download)

2: Create Workspace Models

Highlighted in red are the three Ingest Menu options we will be using for this demonstration.

We will create two temporary data models titled 'csv-ingest' and 'csv-construct' to keep our work separate from the main VIVO models.

1. Select "Ingest Tools" from the Advanced Tools Menu
2. Select "Manage Jena Models"
3. Click on configuration models
4. Click the "Create Model" button then type in a name for your model, 'csv-ingest'.
5. Repeat step 3 and name the model 'csv-construct'

3: Pull CSV File into RDF

Now click 'Convert CSV to RDF' on the Ingest Menu. Begin by supplying a URL for your CSV file, or by uploading the file directly from your computer. Start with people.csv.

Complete the fields in the form

Namespace for Classes and Properties ('in what namespace should these properties be created?':

This namespace can be temporary since we will later map the tool's output to the VIVO/ISF ontology. For example, you can use http://localhost/vivo/

Class Name:

The class name is also a temporary value for this example. This value does not follow the created entity since you will shift the properties from the format they come in into the ontologies format. For this example, the suggested class names are “ws_ppl”

Destination Models:

The data and ontology model option dropdown menus should list the model to ingest into. This is where we select one of those 'workspace' models we created earlier, "csv-ingest" for example.
From the Ingest Menu, select the 'Manage Jena Models' and then select the ingest model's 'output model.' This is something you can open with WordPad and see the created triples for your CSV file.

Click 'Next Step.' Here you will create the URI for your new individuals.

Select URI prefix:

It is most convenient if this matches the URL for your VIVO instance plus '/individual/', e.g. http://vivo.mydomain.edu/individual/.

URI suffix:

This can be a random number or a created based on a pattern plus the value from your CSV file.

Now, click 'Convert CSV.' The CSV data should now be converted into temporary RDF and inserted into the 'csv-ingest' model.

You can now repeat step 3 for both organizations.csv and positions.csv before continuing to assure positions will be attached to both people and organizations, or for simplicity, you can ignore organizations and positions for now.

4: Confirm data property URIs and RDF structure

The CSV to RDF tool converts the CSV into temporary RDF that we can query using SPARQL. This temporary RDF cannot be displayed in VIVO. We will transform the RDF into the VIVO/ISF ontology format in the next step. First, take a look at the temporary RDF we have created.

Ingested Data URIs:

Confirm the data property URI's that were created for each of the columns in your csv file by navigating back to the 'Manage Jena Models' page and clicking 'output model' below csv-ingest. A description of the format is described in the figure below. The predicates are the properties we need for our SPARQL Query. If you used the 'http://localhost/vivo/' format used above

5: Convert temporary RDF into VIVO/ISF ontology using SPARQL

Now, we must query and CONSTRUCT new RDF that is mapped to the VIVO/ISF ontology. Diagrams to help visualize the VIVO/ISF ontology model are available on the wiki at https://wiki.duraspace.org/x/ycCdB. Some helpful links to information on SPARQL queries can be found at https://wiki.duraspace.org/x/lwUGAg. Basically, we will query the database for the triples in Step 4 to pull out the resource URIs and store them in variables (e.g. http://vivo.university.edu/individual/2674803 → ?person) in the WHERE part of the query. We then CONSTRUCT new triples that are in the proper VIVO/ISF format using those variables for the resource URIs (e.g. ?person) and data values (e.g. ?fullname).

Return to the ingest menu and click 'Execute SPARQL CONSTRUCT'
Example SPARQL query for people.csv:

```
CONSTRUCT {
?person <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://vivoweb.org/ontology/core#FacultyMember> .
?person <http://www.w3.org/2000/01/rdf-schema#label> ?fullname .
?person <http://purl.obolibrary.org/obo/ARG_2000028> ?vcard .
?vcard <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2006/vcard/ns#Individual> .
```

```
?vcard <http://www.w3.org/2006/vcard/ns#hasName> ?vcard_name .
?vcard_name <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2006/vcard/ns#Name> .
?vcard_name <http://www.w3.org/2006/vcard/ns#givenName> ?first .
?vcard_name <http://vivoweb.org/ontology/core#middleName> ?middle .
?vcard_name <http://www.w3.org/2006/vcard/ns#familyName> ?last .
?vcard <http://www.w3.org/2006/vcard/ns#hasEmail> ?vcard_email .
?vcard_email <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2006/vcard/ns#Email> .
?vcard_email <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2006/vcard/ns#Work> .
?vcard_email <http://www.w3.org/2006/vcard/ns#email> ?email .
}
WHERE {
?person <http://localhost/vivo/ws_ppl_name> ?fullname .
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_first> ?first . }
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_middle> ?middle . }
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_last> ?last . }
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_email> ?email . }
?person <http://localhost/vivo/ws_ppl_person_ID> ?hrid .
BIND(URI(CONCAT(STR( ?person ), "_vcard")) AS ?vcard ) .
BIND(URI(CONCAT(STR( ?person ), "_vcardname")) AS ?vcard_name ) .
BIND(URI(CONCAT(STR( ?person ), "_vcardemail")) AS ?vcard_email ) .
}
```

Next:

- Select Source Model (‘csv-ingest’)
- Select Destination Model (‘csv-construct’)
- Select "Execute CONSTRUCT"
- Upon completion, the system will report ‘n statements CONSTRUCTed’.

6: Load to Webapp

The live webapp that is indexed does not allow for models to just be attached. Attaching models works well for seeing what we have constructed or ingested or smushed, but those models are lost when Tomcat refreshes.

The final step is to output the final model and add it into the current model
From the Ingest Menu, select "Manage Jena Models"

1. Click "output model" below 'csv-construct'
2. Save the resulting file

3. Navigate back to the Site Administration page
4. Select Add/Remove RDF
5. Browse to the file previously saved
6. Select N3 as import format*
7. Confirmation should state ‘Added RDF from file people_rdf. Added 415 statements.’

The output engine uses N3, not RDF/XML. This is important to note when adding the ‘output mode’ RDF data into the webapp. RDF/XML is the default setting for the drop down list as most ontologies are written in RDF/XML.

The ingested data should now display in both the index and the search results. It is part of the main webapp and will be retained upon a Tomcat restart

Chapter 6

References

The following sources were referred to while working on this project:

- **https://duraspace.org/vivo/**
- **https://wiki.duraspace.org/display/VIVO/VIVO**
- **https://wiki.duraspace.org/display/VIVODOC110x/VIVO+1.10.x+Documentation**
- **https://github.com/vivo-project/VIVO**
- **https://github.com/vivo-project/Vitro**
- **https://github.com/mconlon17/vivo-pump**