SOLA

Solution for open land administration

SLTR NIGERIA UNIQUE REPOSITORY

Contents

[Revision history 1](#_Toc395540357)

[SOLA SLTR NIGERIA REPOSITORY 2](#_Toc395540358)

[DATABASE 3](#_Toc395540359)

[CODE 4](#_Toc395540360)

[ADDING A NEW STATE 9](#_Toc395540361)

# Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | author | date | REASON for revision |
| PA1 | Maria Paola Rizzo | 22/07/2014 | First draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## SOLA SLTR NIGERIA REPOSITORY

SOLA SLTR NIGERIA uses the standard SOLA 10 Git repositories for managing its code base.

These repositories are:

1) code - Contains the Main POM file and

additional supporting files. This Git repository acts

as the super/parent repositories for all the other SOLA

Git repositories.

2) clients - Contains the Clients Desktop and Clients Admin projects

along with the other client side projects. Attached as

the clients subdirectory of the code repository.

3) common - Contains only the Common Utilities project. Attached as

the common/common subdirectory of the code repository.

4) rules - Contains only the Common Rules project. Attached as the

common/rules subdirectory of the code repository.

5) help - Contains only the Common Help project. Attached as the

common/help subdirectory of the code repository.

6) messaging - Contains only the Common Messaging project. Attached as

the common/messaging subdirectory of the code

repository.

7) boundary - Contains the Web Service Boundary projects. Attached as

the services/boundary subdirectory of the code

repository.

8) services - Conatins the EJB and common service projects. Attached

as the services subdirectory of the code repository.

9) database - Contains the SOLA database scripts. Attached relative to

the code repository in the ../database directory.

10) test - Contains the Fitness and Performance test projects.

Attached in the test subdirectory of the code

repository.

You can read the README.TXT files in the code and database repositories

to get information about how repositories are used and handled.

In addition to the standard 10 repositories there are few other repositories: one for each state

Those contain the database scripts specific of each state.

Thus, at the moment additional repositories are:

11) database-crossriver

12) database-jigawa

13) database-kaduna

14) database-kano

15) database-kogi

16) database-ondo

## DATABASE

All changes to the database are captured as a changeset script in the changeset folders (refer to the README.TXT file in the database folder):

\*) database/changeset for the changes that apply to all the state

\*) database-<state>/changeset for the changes that apply only to that specific state

When running the database/create\_soladb\_nigeria.bat you will now be asked also to specify:

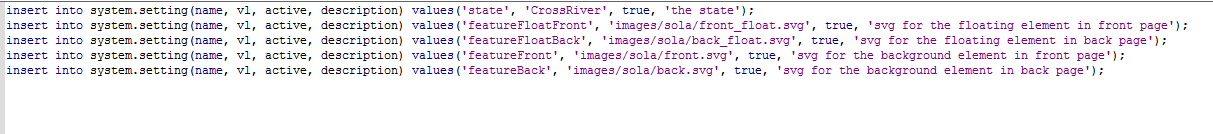
- State (this will lead the installation to get the scripts from the specific state repository)

- LGA office (this will properly set the system-id)

Notice that in each database-<state>/changeset folder there is a script named:

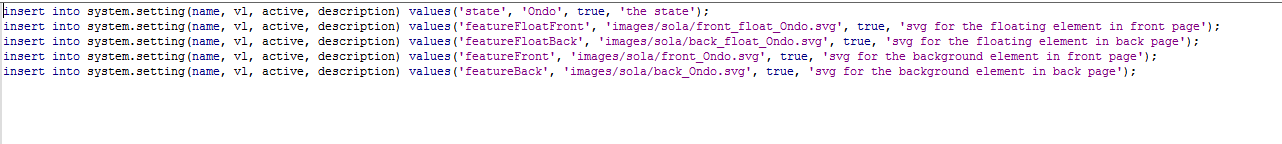
* **05-1408a\_ss.sql**

This script contains the name of the state and the path for the svg files used to customize the CofO.



If needed you can updated the svg path in order to retrieve customized svg files

For instance in case of Ondo the script has been updated as follows:



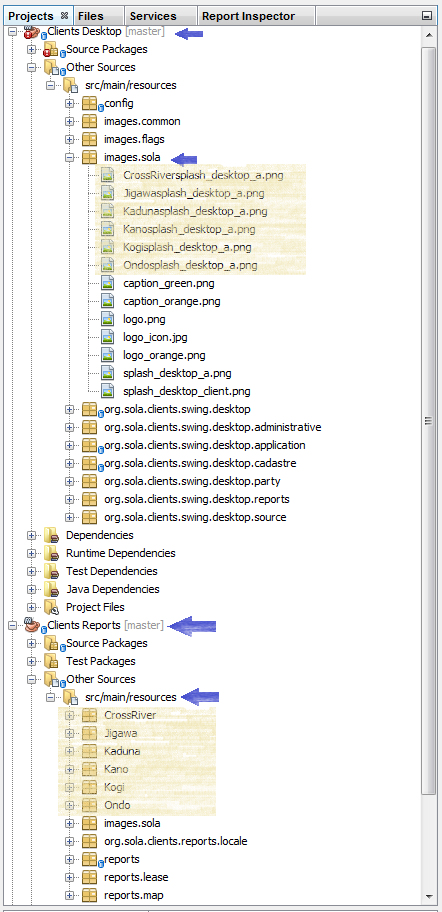
## CODE

#### Clients Desktop & Admin

Clients Desktop and Clients Admin projects have been customized in order to retrieve the splashscreen customized by each state.

These splashscreen are retrieved by the newly created servlet:

Boundary Web Services project 🡺 org.sola.services.boundary.servlets package 🡺StateServlet.java



Under **Client Desktop🡺Other Sources🡺src/main/resources🡺images.sola** package there is one .png image for each splashscreen

Thus, at the moment there are the following splash-screen images:

- CrossRiversplash\_desktop\_a

- Jigawasplash\_desktop\_a

- Kadunasplash\_desktop\_a

- Kanosplash\_desktop\_a

- Kogisplash\_desktop\_a

- Ondosplash\_desktop\_a

Under **Client Admin🡺Other Sources🡺src/main/resources🡺images.sola** package there is one .png image for each splashscreen

Thus, at the moment there are the following splash-screen images:

- CrossRiversplash\_admin\_a

- Jigawasplash\_admin\_a

- Kadunasplash\_admin\_a

- Kanosplash\_admin\_a

- Kogisplash\_admin\_a

- Ondosplash\_admin\_a

#### Clients Reports

Clients Reports project has been customized in order to retrieve the reports customized by each state and to customize some of the png and svg used.

Under **Client Report🡺Other Sources🡺src/main/resources** there are few other folders in addition to the standard resources (images,org and reports), one for each state which contain the reports specific to that state.

Thus, at the moment there are the following subfolders:

- CrossRiver

- Jigawa

- Kaduna

- Kano

- Kogi

- Ondo

- images

- org

- report

Under each of the <state> folder there are the reports customized by each state:

**e.g.**

- PD listings

- CofO

- SLTR PLAN

Under **Client Report🡺Other Sources🡺src/main/resources** 🡺 **images.sola**

there is a <state>logoMinistry.png for each state

[09:50:21] paolapaola.work: for each state

[09:50:23] paolapaola.work: so

[09:50:39] paolapaola.work: you should add also a KatsinalogoMinistry

[09:50:49] paolapaola.work: with the logo for Katsina

[09:51:06] klx500: thanks

[09:51:12] paolapaola.work: you might also notice that there are few .svg

[09:51:45] paolapaola.work: Page1.svg

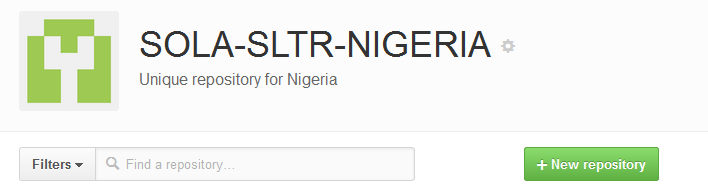
Page2.svg

Page3.svg

## ADDING A NEW STATE

The following steps allow for adding a new state.

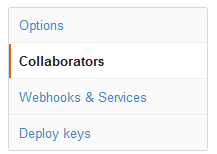
1. Create the database following the standard instructions
   1. Start pgAdmin III and connect to the PostgreSQL server.
   2. Right click the Databases node and choose New Database...
   3. On the Properties tab, enter the name of the database as **sola**. PostgreSQL is case sensitive, so use lower case for the database name.
   4. On the Definition tab, select **template\_postgis\_20** as the database template. Database Templates are a feature of PostgreSQL that allow all functions and structure of the template database to be inherited by the new database.
   5. Click OK. If you get an error indicating the database “template\_postgis\_20” is being accessed by other users, return to pgAdmin, right click the Databases node and choose the Refresh option before attempting to create the new database again.
2. Add interim\_data schema importing any spatial data for state, lga and wards from shapefiles or whatever source might be available
3. Login into GITHUB SOLA-SLTR\_NIGERIA
4. Click on the “New repository” button



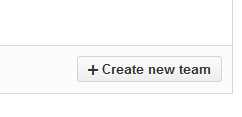
1. Fill the “repository name” field with the database-<NewStateName>
2. Check public
3. Check “Initialize this repository with a README”
4. Click “Create Repository”
5. Click on “settings” icon on the right side



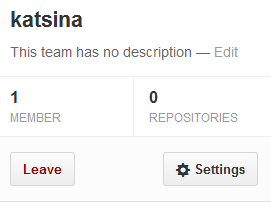
1. Click on “Collaborators” option on the left side



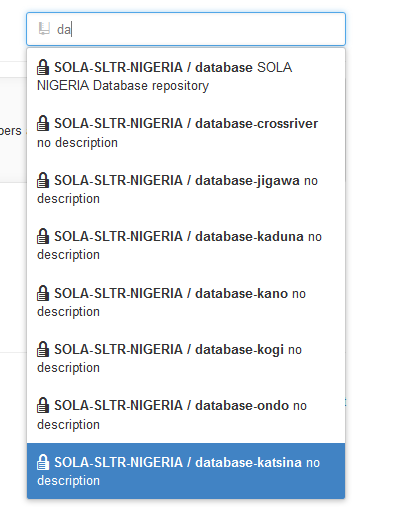
1. Click on “Create new team” button on the right side



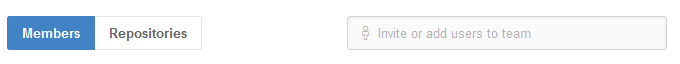
1. Enter the <new state name> as name of the new team
2. Check the Admin option
3. Click on “Create team” button
4. Click on “Repositories” on the left side



1. Search and select the database-<NewStateName> you have just created



1. Click on “Members” tab in the upper side



1. Invite or add user to the team using the search field on the left
2. Open a git bash window and use GIT commands to clone:
   1. **git clone** [git@github.com:SOLA-SLTR-NIGERIA/database<NewStateName>](mailto:git@github.com:SOLA-SLTR-NIGERIA/database<NewStateName>)
3. In the new database-<*new\_state\_name*> repository from another database-<state> reporitory copy the folders:
   1. changeset
   2. migration
   3. extension
4. Update the files in those folders in order to be customized for the new state
5. Run the database/create\_soladb\_nigeria.bat script
6. Check the build.log file in order to verify the creation of the new state database has been successful
7. Under SOLA-SLTR-NIGERIA\sola\code update the .gitignore file adding at the bottom the string:

"../database-<NewStateName>" "../database-<NewStateName>"

1. Use GIT commands to update the new database-<NewStateName> folders and push them:
   1. **gits -v add -A** - Stages any changed or new files to the index of the super of slave repositories that have outstanding changes
   2. **gits -v commit -a -m <commit message>** - Commits all staged changes to the approrpiate super or slave repository along with the specified (short) commit message. This command is useful to synchronize the commit message across multiple repositories
   3. **gits -v pull** - This will update the currently checked out branch for the super repository and the slave repositories on your local file system from the GitHub repositories
   4. **gits -v push** - This will push all newly committed changes on the super and slave repositories back to GitHub. This command may take up to 1 minute to run
2. Go again to github and verify the new folders have been added
3. Using a graphic editor create the desktop and admin spalshscreens for the new state
4. Save those new images under:
   1. **ClientDesktop🡺OtherSources🡺src/main/resources🡺images.sola/**

**<NewStateName>splash\_desktop\_a**

* 1. **ClientAdmin🡺OtherSources🡺src/main/resources🡺images.sola/**

**<NewStateName>splash\_admin\_a**

1. Under **Client Report🡺Other Sources🡺src/main/resources** copy one of the existing subfolder and re-name it as:

* <***NewStateName*>**

1. Customize:
   1. PD Listings
   2. CofO
   3. SLTR PLAN
2. Add new customized images under:

* **images.sola:**
  + **<newState>**logoMinistry

**If the svg for the CofO must be customized then add new svg files following the name convention:**

* + **front\_<newState>.svg**
  + **front\_float\_<newState>.svg**
  + **back\_<newState>.svg**
  + **back\_float\_<newState>.svg**

1. Save
2. Compile the customized reports
3. Clean&Build Clients POM
4. Use GIT commands to add new files and folders and push them:
   1. **gits -v add -A** - Stages any changed or new files to the index of the super of slave repositories that have outstanding changes
   2. **gits -v commit -a -m <commit message>** - Commits all staged changes to the approrpiate super or slave repository along with the specified (short) commit message. This command is useful to synchronize the commit message across multiple repositories
   3. **gits -v pull** - This will update the currently checked out branch for the super repository and the slave repositories on your local file system from the GitHub repositories
   4. **gits -v push** - This will push all newly committed changes on the super and slave repositories back to GitHub. This command may take up to 1 minute to run