**State Records NSW**

Search (API) Developer Guide

Version 0.10

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Draft** | **Date** | **Changes** | **Author** |
| 0.10 | 14/01/2013 | First draft | Wisanu Promthong |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| **Baseline Date** | **Comments** |

**Contents**

[1. Architecture Overview 3](#_Toc345677866)

[2. Development 3](#_Toc345677867)

[2.1 Developer’s prerequisite skills 3](#_Toc345677868)

[2.2 Development environment 3](#_Toc345677869)

[2.3 Running the application 6](#_Toc345677870)

[2.4 Build and Release 6](#_Toc345677871)

[2.5 Depolyment 7](#_Toc345677872)

[Setting up the webserver 7](#_Toc345677873)

[3 Database schema specification 8](#_Toc345677874)

[4 The Code 9](#_Toc345677875)

[4.1 Adding an entity (model) 9](#_Toc345677876)

[4.2 Adding a view 9](#_Toc345677877)

# Architecture Overview

The key technical functionalities of Search API are as following:

* Rendering HTML presentation of the records
* Rendering API format presentation of the records (Mods, eac\_cpf, JSON, XML, rdf\_zotero, etc)
* Searchability

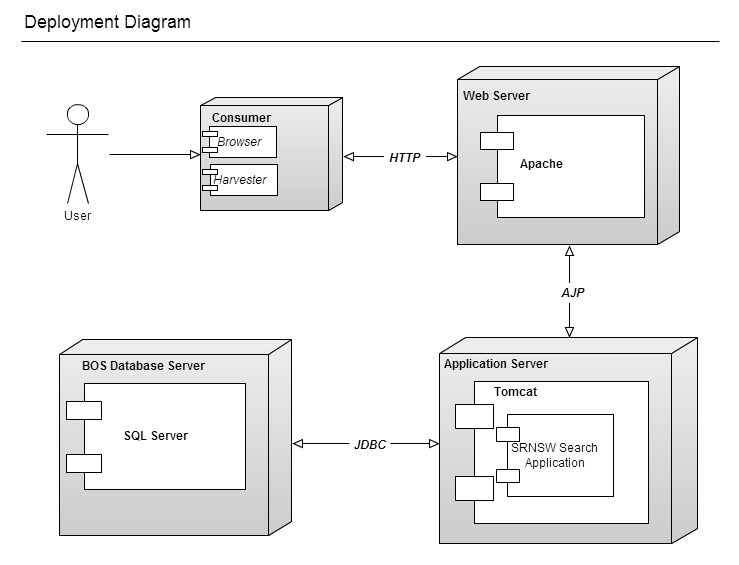


Figure Depolyment diagram

# Development

The aim of this section is to ensure that other developers can carry on the development of the Search API website to ensure the project can continue even in the absence of the original developer.

## Developer’s prerequisite skills

In order to continue developing the Search API website, ideally, the developer should have experience using the following technologies:

* J2EE development with Eclipse IDE
* MS SQL (or any RDBMS)
* Any source control system
* WebService
* Usage of open source libraries
* Spring Roo
* Apache Tomcat webserver

## Development environment

In order to develop the SRNSW Search API website, the following software and libraries are required on a Windows environment.

* STS (SpringSource Tool Suite) – Main IDE for Spring Roo web development
* Egit – GIT plugin for STS

#### Setting up the development environment

* Ensure that you have Egit plugin installed on STS
* Use Egit to clone the project from <https://github.com/srnsw/State-Records-Search.git> by using Egit plugin as illustrated below.

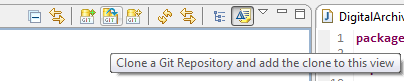


Figure Use Egit to get the source code from Github

**Note:** You will need to provide a user credential in order to clone the source code

* Follow the prompts to clone the project from Github.
* Once finished, right click on the recently cloned project to import to STS as a project by following the prompts.

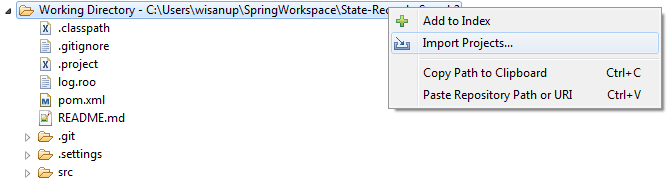


Figure Import project to current workspace

* If successful, you should see the Search API project under STS without any JAVA error.

#### Project structure in Eclipse

Once you have cloned the source code from GIT by Egit, you will see the following folder structure in STS.

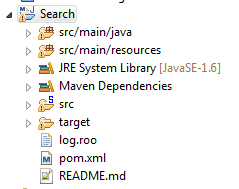


Figure Project structure

This application is just a basic Roo application where it provides database’s record reading functionalities. The project basically contains datamodel definition and also web service controller class, no additional webflow, email or messaging.

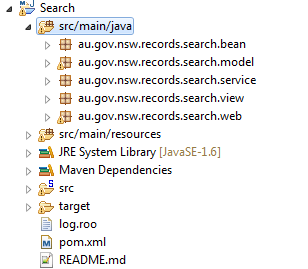


Figure Source folders

There are five packages as following:

* bean – JAVA bean classes definition
* model – Datamodel classes reflect with the schema defined in BOS database. This is the **M**odel definition part of **M**VC framework.
* service – Additional service classes
* view – the view finder for finding the proper view template of the requested media type.
* web – The **C**ontroller definition for MV**C** framework

There is also one additional path for storing **V**iew definition for M**V**C framework at /src/main/webapp/WEB-INF/views/\*

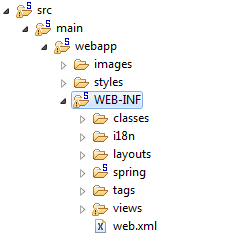


Figure View folder

## 2.3 Running the application

To start this application, right click on the project and then select Run As > Run on Server as shown in the figure below

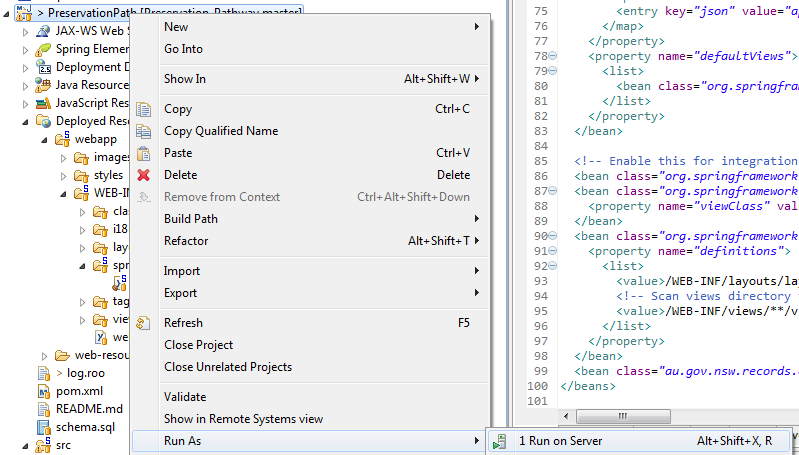


Figure Starting the application

## 2.4 Build and Release

The build and release procedure is quite simple. There are two steps to perform to accomplish the task.

Fist step: Modify attribute <version> in pom.xml file

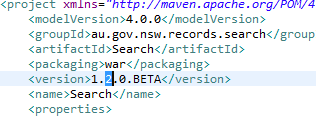


Figure Update version number

Second step: Build the package. Currently, we use Roo’s functionality to build the release package.

Simply open Roo Shell and then issue the command perform package. As a result, the release package will be available at ./target/ directory.

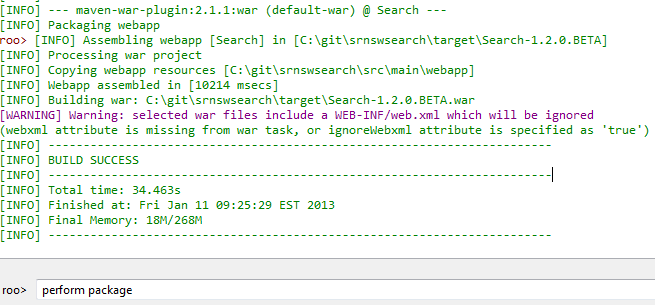


Figure Issue ‘perform package’ command at Roo Shell

Search Application can be deployed as a TomCat web application. So basically it needs an instance of TomCat to be started first.

## 2.5 Depolyment

## Setting up the webserver

* Ensure that you have JAVA installed on your system (JAVA 1.6 minimum) by issuing the command java –version from the shell prompt.
* Rename the released package to ROOT.war
* Start TomCat application
* Open <http://localhost>:8080 in a browser to test the TomCat web server, the default TomCate webpage should be displayed.

**Depoly Search application**

* Place the WAR file at {tomcat\_home}/webapps/ path
* Setup database configuration at {tomcat\_home}/webapps/ROOT/WEB-INF/classes/META-INF/spring/database.properties file
* Setup lucene configuration at {tomcat\_home}/webapps/ROOT/WEB-INF/classes/META-INF/spring/lucene.properties file
* Setup log4j configuration at {tomcat\_home}/webapps/ROOT/WEB-INF/classes/ /log4j.properties file
* Restart TomCat to reload all configurations and the Search API web application should be ready at <http://localhost>:8080 .

# 3 Database

Currently, the application expects to connect to the MS SQL database with the following tables or views definition (double click to open).



In order to use this Search application you must ensure that the database has schema as shown in the attach file.

Note that this application performs the read operation only.

# 4 The Code

This section is mostly about application development using Spring MVC framework

## 4.1 Adding an entity (model)

To add a new entity, issue the entity command in Roo’s console to create the database model class.

entity jpa --class au.gov.nsw.records.search.model.{Entity Name}

Then subsequently add desired fields to the newly created entity by using command field in Roo’s console. Note that the field must reflect to the database schema definition.

## 4.2 Adding a view

To create a view related to the created model, issue the web mvc scaffold command in Roo’s shell. The relevant controller and view templates are created after we issued the command.

web mvc scaffold --class ~.web.{Controller Name}

At this point, you can modify view template to show or not show some fields.

Note that relevant view templates resides at src/main/webapp/WEB-INF/views/

**Adding a non HTML view**

By default, Spring Roo application renders only HTML view. The custom class is introduced to allow adding custom view (JSON, XML, RIF, MODS, etc). The au.gov.nsw.records.search.view.SimpleMediaTypeView is a custom class to handle the view according to the request format type.

To utilize the specified format to SimpleMediaTypeView class, the webmvc-config.xml must be configured as following example to ensure that the application can handle the request of the certain format.

<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">

<property name="mediaTypes">

<map>

<entry key="rif" value="application/rif"/>

</map>

</property>

<property name="defaultViews">

<list>

<bean

class="au.gov.nsw.records.search.view.SimpleMediaTypeView" id="rifView">

<property name="mediaType" value="application/rif" />

</bean>

</list>

</property>

</bean>

After defined an entry in webmvc-config.xml, we need to add a view template class to handle that specific format rendering. Ideally the custom view template file should be (but not limited to) resided at the same pathas the jspx file of that entity.

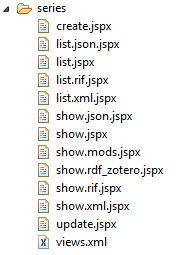


Figure Example of custom view template files