

**Quick Start Framework**

***for Spring MVC***

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| *MAY-03-2010* | *1.0* | *Initial Version* |
|  |  |  |
|  |  |  |

**Table of Contents**

[1 Introduction 4](#_Toc355349876)

[2 Downloads 4](#_Toc355349877)

[3 Framework Installation 4](#_Toc355349878)

[4 Database Setup 4](#_Toc355349879)

[5 Security Setup 6](#_Toc355349881)

# Introduction

Quick start Framework helps to integrate a new driven application, easily and

consistent manner. QS-Framework configured with Spring 3.1, more easily annotation driven, also pre-configured with hibernate 3.x. Version and Apache tiles. QS-Framework also contains spring security with dynamic role access permission. The following modules are used in this framework:

The modules are,

Login

It consists of credential authentication by spring security and remember me functionality by cookies and forget password functionality, to send new password via e-mail configure e-mail Id at **FrameWork\war\WEB-INF\config\mail-context.xml**

User Management

It consists of user creation/modification and users list.

Role Management

It consists of Role creation/modification and roles list. In role creation view, create, edit & delete mode can be configured to menus and submenus.

# Downloads

To access QS-Framework, download from the configured location

**http://localhost:8080/UtilityTrackerDownload.**

# Framework Installation

Import downloaded QS-Framework project into eclipse as java project and add the library

files into build path.

Configure tomcat location at **FrameWork\build\build.properties**

Configure project location at **FrameWork\dist\FrameWork.xml**

# Database Setup

Find the database.properties file at **FrameWork\war\WEB-INF\config\database.properties** in QS-

Framework and modify with current working database name, username and password.

**4.1 Table Creation**

Proceeding, the initial setup is to be followed by creating tables by using scripts available in

location **FrameWork\docs\DataBaseScripts\TableScripts.sql** folder. After execution of all scripts, tables created are listed below,

|  |  |  |  |
| --- | --- | --- | --- |
| **Table: Menus** | | | |
| **S. No.** | **Field Name** | **Constraints** | **Description** |
| 1 | MenuId | PK | Primary Key of the table (Auto generated) |
| 2 | MenuName | - | Name of the menu to display. |
| 3 | MenuDescription | - | Description about menu. |
| 4 | ModifiedDate | - | Date of modified. |
| 5 | OrderId | - | Order of the menu. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table: SubMenus** | | | |
| S. No. | Field Name | Constraints | Description |
| 1 | SubMenuId | PK | Primary Key. Auto Generated |
| 2 | MenuId | FK | Mapped with **Menus** - **MenuId** |
| 3 | SubMenuName | - | Name of the sub menu to display |
| 4 | SubMenuDescription | - | Description about submenu |
| 5 | ModifiedDate | - | Date of modified |
| 5 | OrderId | - | Order of the sub menu |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table: Users** | | | |
| S. No. | Field Name | Constraints | Description |
| 1 | UserId | PK | Primary Key of the table (Auto generated) |
| 2 | FirstName | - | First name of the user |
| 3 | LastName | - | Last name of the user |
| 4 | RoleId | FK | Mapped with Table **Roles** – **RoleId** |
| 5 | EmployeeId | - | Unique Id of employee |
| 6 | Department | - | User’s department |
| 7 | UserName | - | Username to login |
| 8 | Password | - | Password of the user to login |
| 9 | WorkingPhone1 | - | Primary work phone number |
| 10 | WorkingPhone2 | - | Secondary work phone number |
| 11 | Extn1 | - | Extension number of primary work phone number |
| 12 | Extn2 | - | Extension number of Secondary work phone number |
| 13 | MobileNumber1 | - | User mobile number |
| 14 | MobileNumber2 | - | User mobile number |
| 15 | EmailId | - | Email – id of the user |
| 16 | Status | - | State of being Active or Inactive of a user |
| 17 | ModifiedTime | - | Modified date of user information |
| 18 | IsDeleted | - | Is soft deleted or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table: Roles** | | | |
| S. No. | Field Name | Constraints | Description |
| 1 | RoleId | PK | Primary key of the table (Auto generated) |
| 2 | RoleName | - | Name of the Role |
| 3 | Organization | - | Organization |
| 4 | Status | - | State of being Active or Inactive of a role |
| 5 | isDeleted | - | Is soft deleted or not |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table: Roles** | | | |
| S. No. | Field Name | Constraints | Description |
| 1 | PermissionId | PK | Primary key of the table (Auto generated) |
| 2 | RoleId | FK | Mapped with Table **Roles** – **RoleId** |
| 3 | MenuId | - | MenuId of belongs to Menus Table - MenuId |
| 4 | SubMenuId | - | SubMenuId of belongs to SubMenus Table - SubMenuId |
| 5 | IsViewEnabled | - | Boolean Value. True if View Enabled else false |
| 6 | IsCreateEnabled | - | Boolean Value. True if Create Enabled else false |
| 7 | IsEditEnabled | - | Boolean Value. True if Edit Enabled else false |
| 8 | IsDeleteEnabled | - | Boolean Value. True if Delete Enabled else false |

After successful execution of scripts, the recommended data entries are as follows,

1. Add the entire Main menu (Admin) in the menus table.

2. Add submenus in the Submenus (User Management, Role Management) Table and map with main menu id.

3. Add the role in the roles table.

4. Add the role access permission entries manually to access all the menus & submenus and map with above created role.

5. Add users in the users table and map the role id with above created role.

To create stored procedures for user and role list, execute scripts at **FrameWork\docs\DataBaseScripts\StoredProcedures.sql**

# Security Setup

Security Set up holds menus and submenus information and URLs that belongs to a menus and

Submenus.

As initial set up of security **ApplicationConfigLoader** carried out. Here several tasks done.

**1. Make ready for the Menus and Submenus info list:**

The menus information is Stored in HashMap<menu\_id, MenuInfo> , and another HashMap<main\_menu\_name,sub\_menus\_info>.

**2. Building Security Menu Tracker HashMaps:**

In order to identify the menu and submenus, by generating trackId (a Random alpha numeric value) for each sub menu name, are stored in a SecurityHolder. MENU\_TRACKER\_MAPPING HashMap with key as TrackId and value as Sunmenu\_Name and the same SecurityHolder. TRACKER\_MENU\_MAPPING HashMap with key as Submenu\_name and value as TrackId.

**3. Building Security Menu & Submenus Modules:**

All the urls used in the application need to specify in UrlConstants.

SubMouduleInfo – which hold the URLs belong to a submenu with four different modes, namely view, create, edit, delete.

Submenu Module contains four lists to hold the URL on appropriate modes specified above, the list are,

1. ViewModeEnabled

2. CreateModeEnabled

3. EditModeEnabled

4. DeleteModeEnabled

To hold the Submenus URLs on appropriate modes we need to add the URLs on it.

For Every URL used in the application should be added to the SubModuleInfo by their appropriate mode’s list.

In case of URL without view, edit, create, delete modes should be add to COMMON\_URLS.

**Example:-**

For Example if you are creating a module tenant, with four functionalities view, create, update, delete with urls viewTenants.do, createTenants.do, updateTenants.do, deleteTenants.do respectively. Add the Sub menu name “Tenants” in menu constants. Add all these four urls in UrlConstants.

**Sample MenuConstants,**

|  |
| --- |
| Private static String TENANT\_MANAGEMENT=”Tenant Management” |

\*\*The menu name should be same as specified in Menus & submenus table.

**Sample UrlConstants,**

|  |
| --- |
| Private static String TENANT\_VIEW\_URL = “/tenant/viewTenants.do”;  Private static String TENANT\_CREATE\_URL = “/tenant/createTenants.do”;  Private static String TENANT\_UPDATE\_URL = “/tenant/updateTenants.do”;  Private static String TENANT\_DELETE\_URL = “/tenant/deleteTenants.do”; |

Sample Load modules in application configuration loader, - This same procedure have to followed for all menus going to use in applications. And finally the subModuleInfo is added to SecurityHolder.SUBMODULE\_URL hash table with key of appropriate menu.

|  |
| --- |
| Sample Load modules code in application configuration loader |
| subModuleInfo = **new** SubModuleInfo();  subModuleInfo.setSubModuleName(MenuConstants.*TENANT\_MANAGEMENT*);  // view  subModuleInfo.setViewModeUrls (**new** ArrayList<String>());  subModuleInfo.getViewModeUrls().add(UrlConstants.TENANT\_VIEW\_URL);  //Add other view mode URL here…  // create  subModuleInfo.setCreateModeUrls(**new** ArrayList<String>());  subModuleInfo.getCreateModeUrls().add(UrlConstants.TENANT\_CREATE\_URL);  //Add other create mode URL here…  // edit  subModuleInfo.setEditModeUrls(**new** ArrayList<String>());  subModuleInfo.getEditModeUrls().add(UrlConstants.TENANT\_UPDATE\_URL);  //Add other edit mode URL here…  // delete  subModuleInfo.setDeleteModeUrls(**new** ArrayList<String>());  subModuleInfo.getDeleteModeUrls().add(UrlConstants.TENANT\_DELETE\_URL);  //Add other delete mode URL here…  SecurityHolder.*SUBMODULE\_URLS*.put(MenuConstants.*TENANT\_MANAGEMENT*, subModuleInfo); |

The Common Url’s should be added to the SecurityHolder.COMMON\_URLS.

**To use the above mention urls in tenant.jsp.**

|  |
| --- |
| Sample code in tenant.jsp |
| <%  UserSessionInfo user = **null**;  String trackId = "";  String sessionId = "";  **if** (session.getAttribute(ApplicationConstants.USER\_SESSION\_KEY) != **null**) {  user=(UserSessionInfo)session.getAttribute(ApplicationConstants.USER\_SESSION\_KEY);  sessionId = request.getSession().getId();  trackId=SecurityHolder.MENU\_TRACKER\_MAPPING.get(MenuConstants.TENANT\_MANAGEMENT) + "," + sessionId;  }    %>  <script>  Var uid= “<%=trackId%>”;  function toView(){  window.location.href = “../tenants/viewTenants.do?uid=”+uid;  }  </script> |

The trackId used above url is combination of sessionId, and Menu Track Id. The Menu TrackIds are generated while application started and stored in a Hash table.

**Validating URL on MyFilterSecurityMetadataSource**

Every URL hit the server – will be pass through MyFilterSecurityMetadataSource class exclude URLs specified in the SecurityContext.xml in security:filter-chain-map with filters attribute as none (eg. <security:filter-chain pattern="/styles/\*\*" filters="none" />). Each url must looks like above shown sample url, else the Session will be invalidated and redirected to login page.

The URL’s with TrackID will be manipulated and split into Current-URL, sub menu TrackID, sessionId.

From sub menu TrackID, can find the submenu name and SubModuleInfo which contains all URLs (view, create, edit, deletes).

The Current-URL is checked on SubModuleInfo view, create, edit, and delete modes and common URLs. If match found return null (grant access), else access is denied.