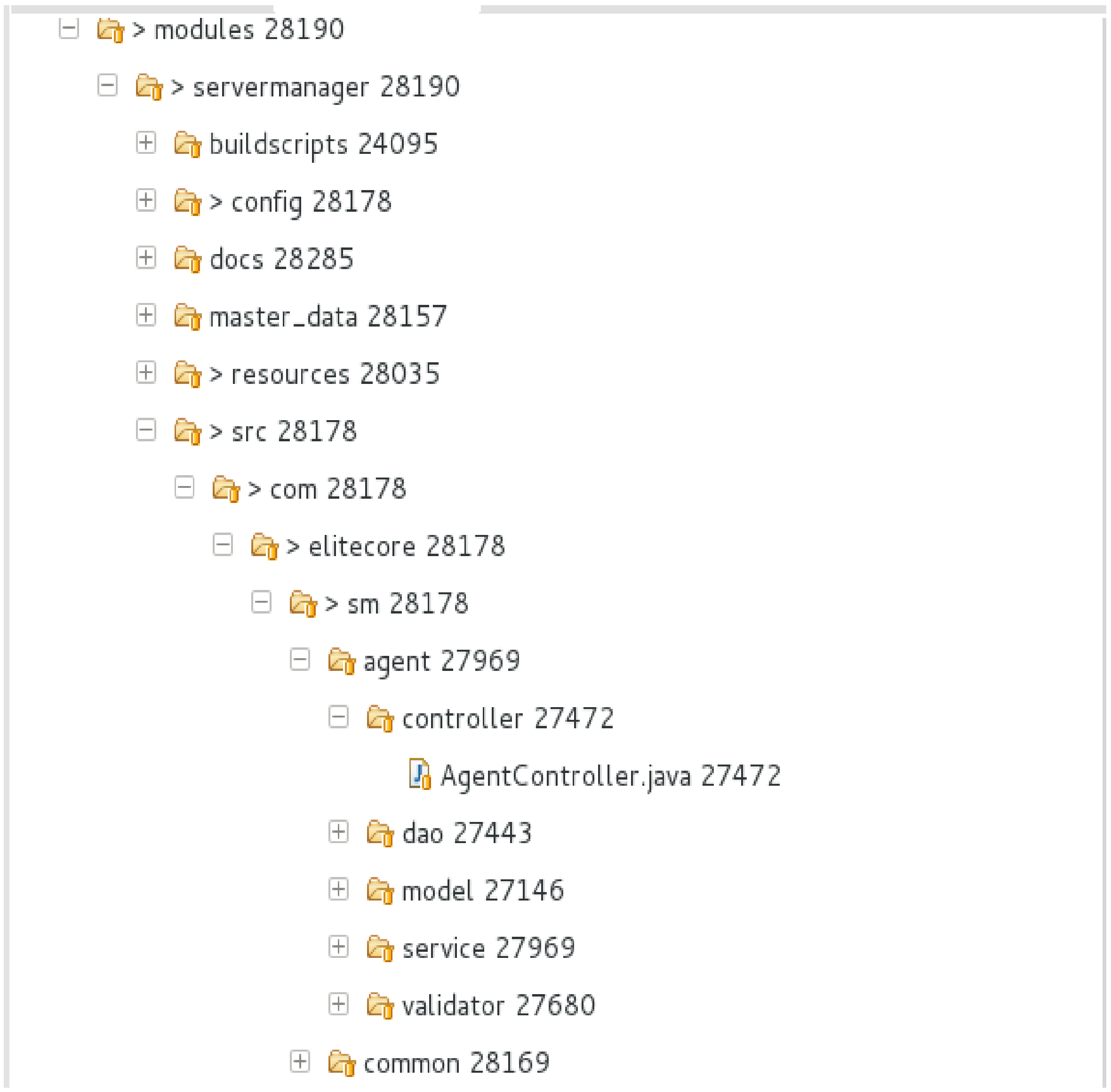
Step 1. Configure the project in Eclipse.

- Refer: http://192.168.3.122:3690/svn/cresteldocs/Product/Mediation-CGF/SM_Revamp_7.0/7.0/Design/Development_Guide
-
- SVN Path: http://192.168.0.181:3690/svn/crestelmediation/projects/SM_REVAMP/ServerManager/trunk

Note: Host Entry for crestelsvn

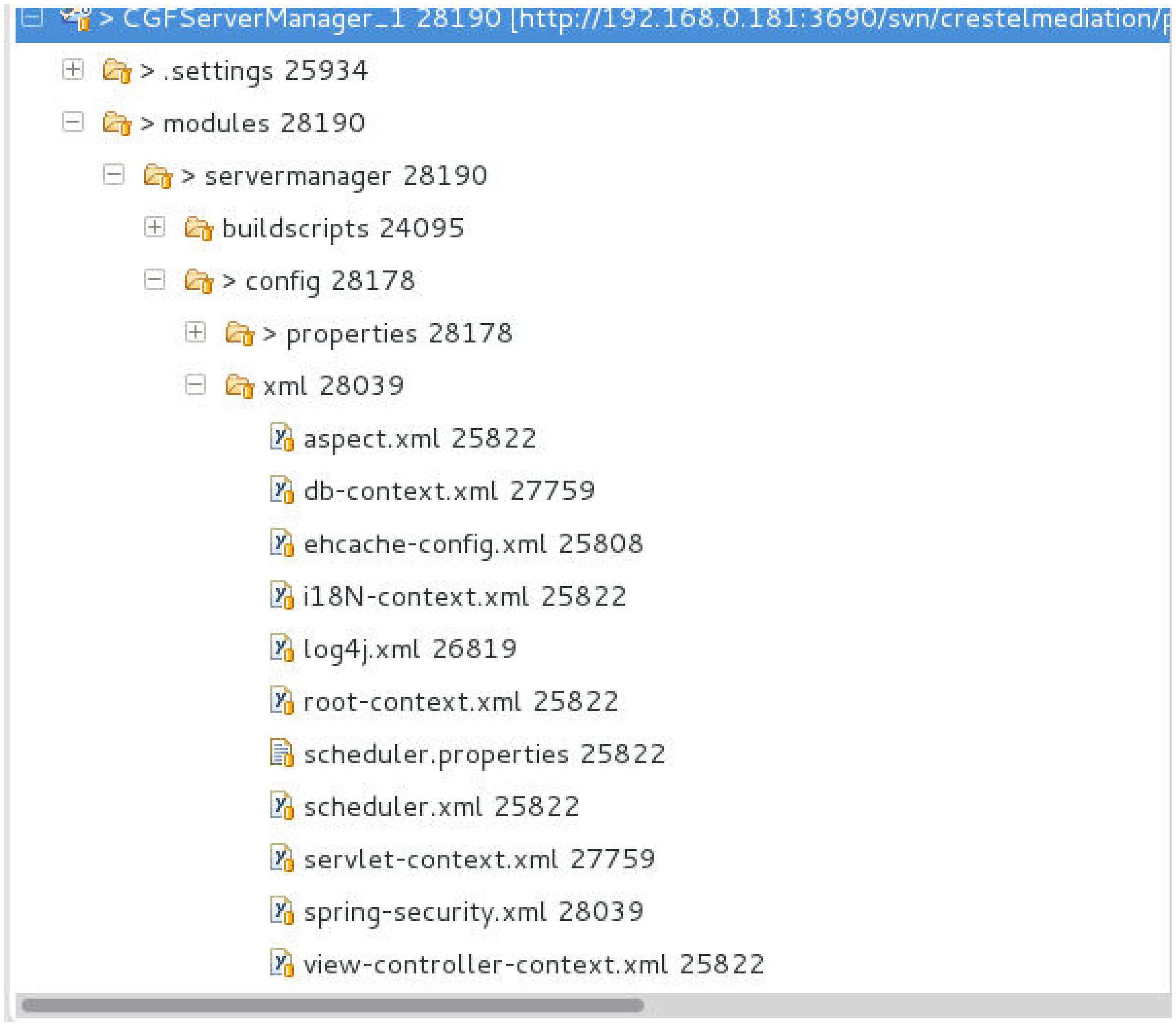
192.168.0.181 crestelsvn

Step 2. Project Structure



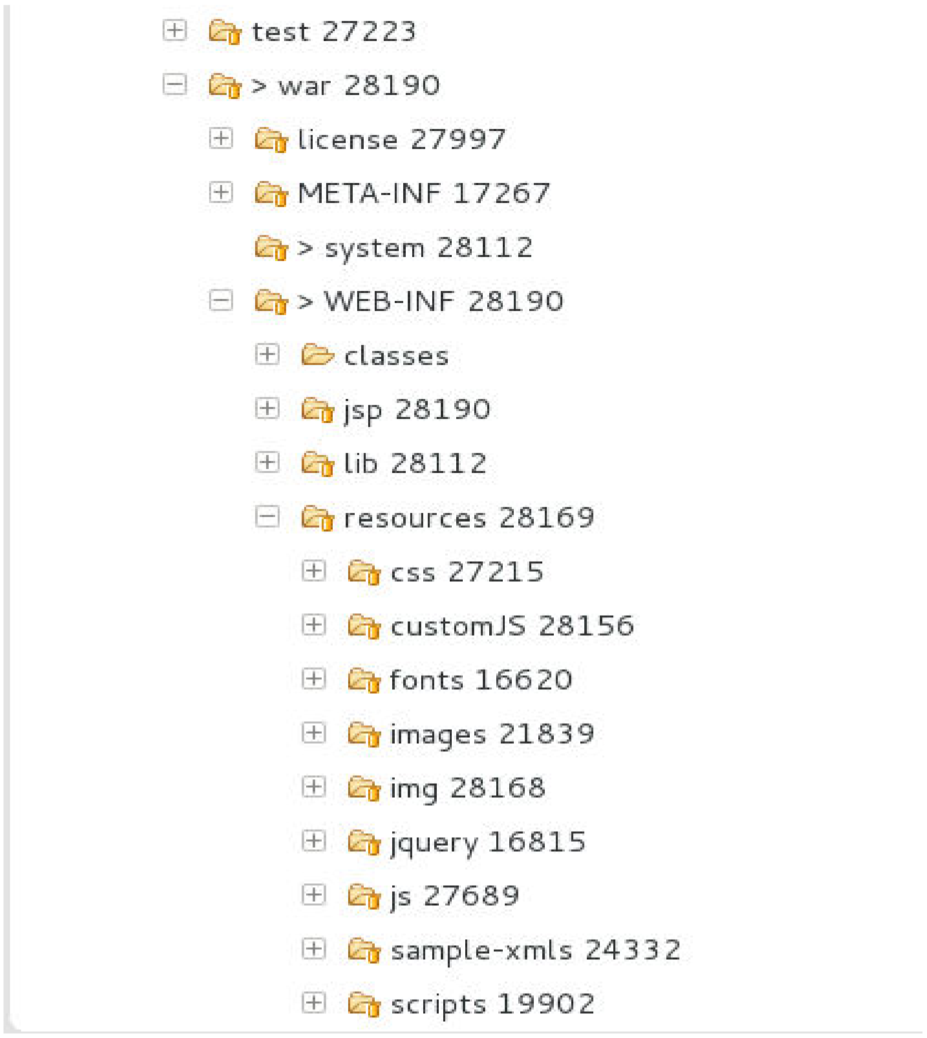
```
⊟ > modules 28190
  ⊟ > servermanager 28190
    ⊞ buildscripts 24095
    ⊞ > config 28178
    ⊞ docs 28285
    ⊞ master_data 28157
    ⊞ > resources 28035
    ⊟ > src 28178
      ⊟ > com 28178
        ⊟ > elitecore 28178
          ⊟ > sm 28178
            ⊟ agent 27969
              ⊟ controller 27472
                  AgentController.java 27472
              ⊞ dao 27443
              ⊞ model 27146
              ⊞ service 27969
              ⊞ validator 27680
            ⊞ common 28169
```

– Source (src) folder contains the Java Code. It have been sliced based on the modules (agent,common,composer,config,dashboard,device,server,serverinstance etc.)



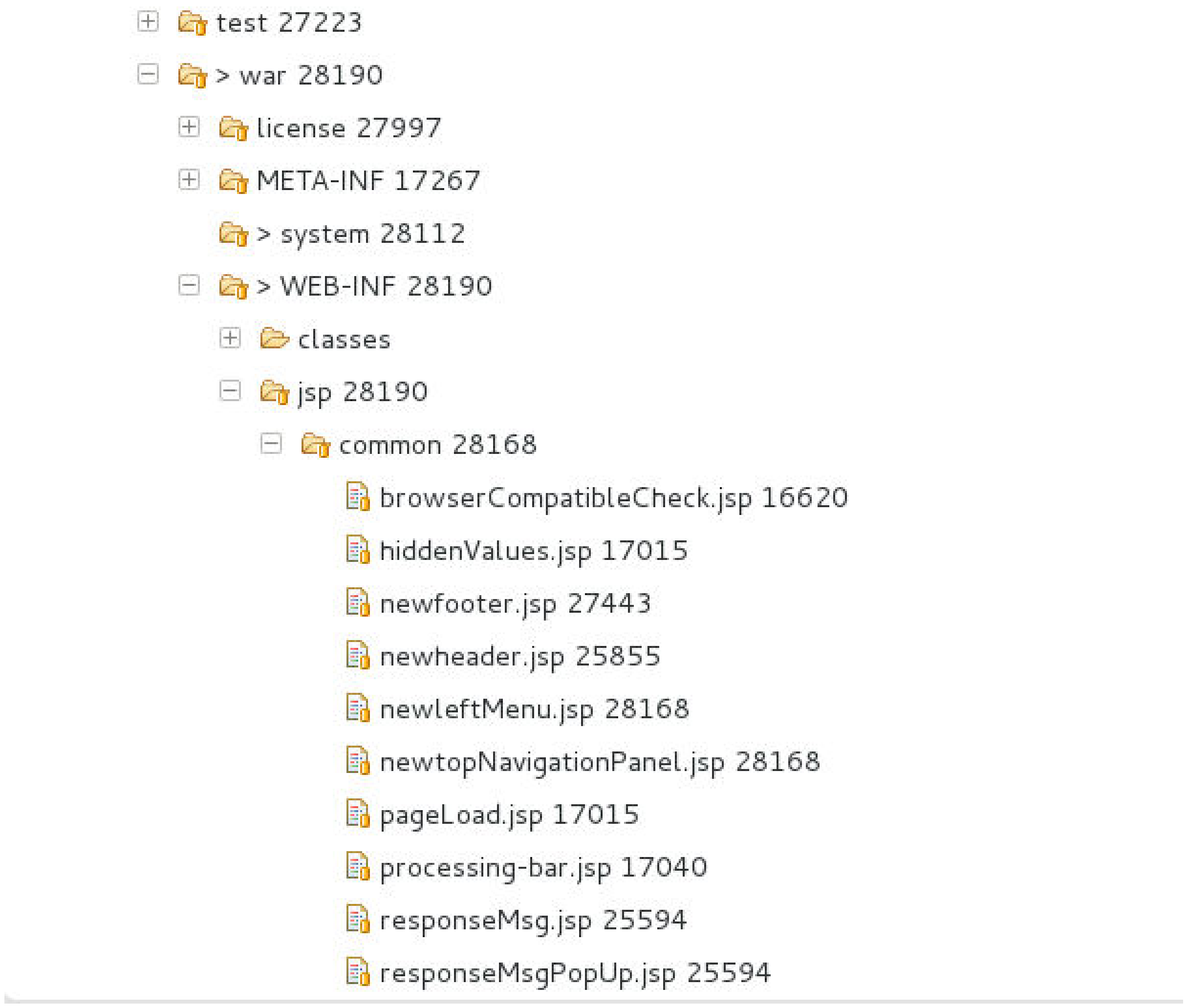
```
> CGFServerManager_1 28190 [http://192.168.0.181:3690/svn/crestelmediation/p
    > .settings 25934
    > modules 28190
        > servermanager 28190
            buildscripts 24095
            > config 28178
                > properties 28178
                xml 28039
                    aspect.xml 25822
                    db-context.xml 27759
                    ehcache-config.xml 25808
                    i18N-context.xml 25822
                    log4j.xml 26819
                    root-context.xml 25822
                    scheduler.properties 25822
                    scheduler.xml 25822
                    servlet-context.xml 27759
                    spring-security.xml 28039
                    view-controller-context.xml 25822
```

– **config** folder contains all the spring configuration:



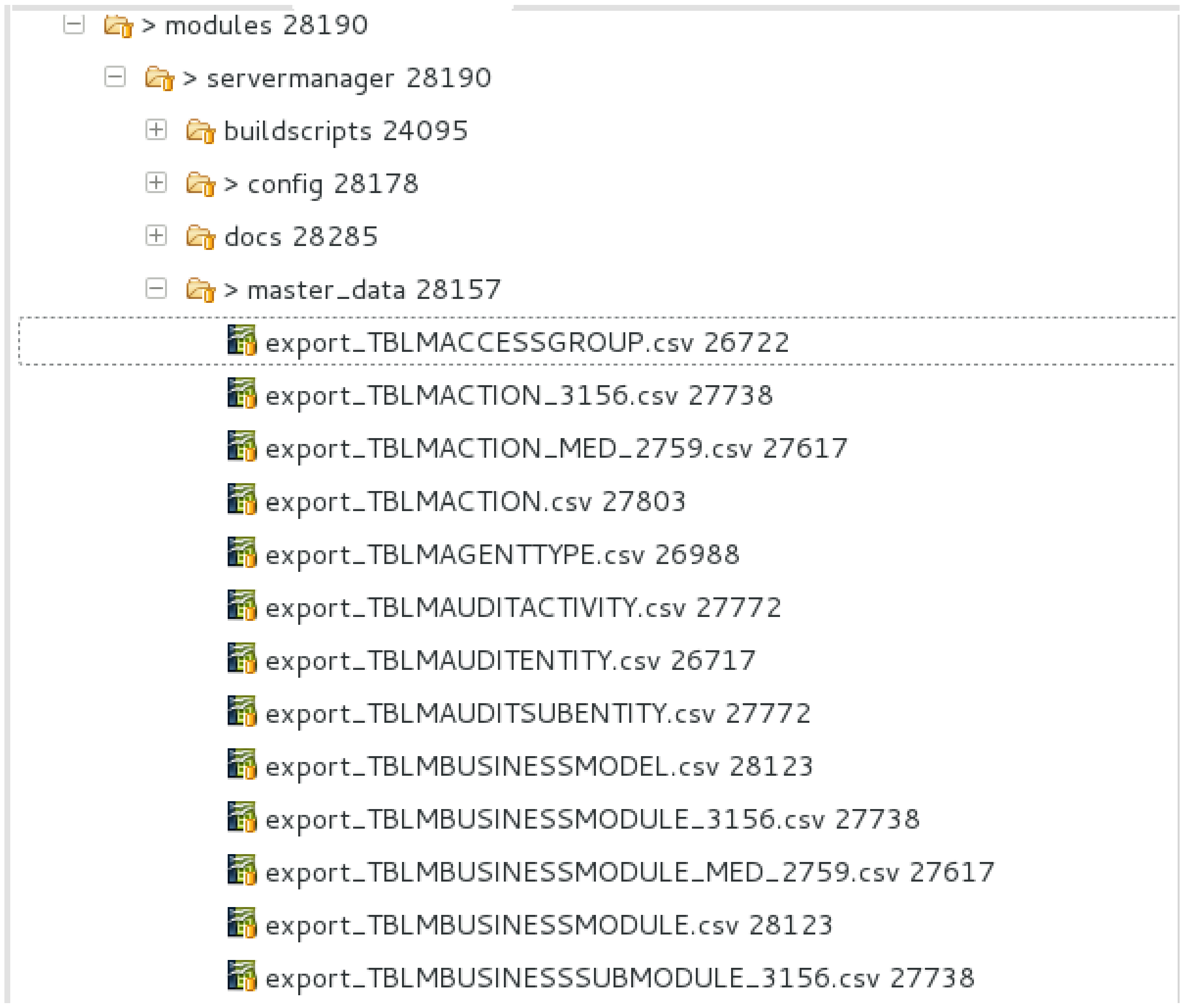
```
⊞ test 27223
⊟ > war 28190
    ⊞ license 27997
    ⊞ META-INF 17267
      > system 28112
    ⊟ > WEB-INF 28190
        ⊞ classes
        ⊞ jsp 28190
        ⊞ lib 28112
        ⊟ resources 28169
            ⊞ css 27215
            ⊞ customJS 28156
            ⊞ fonts 16620
            ⊞ images 21839
            ⊞ img 28168
            ⊞ jquery 16815
            ⊞ js 27689
            ⊞ sample-xmls 24332
            ⊞ scripts 19902
```

– **resources** contains css, scripts, images, fonts that will be used in jsp pages.



```
⊞ test 27223
⊟ > war 28190
    ⊞ license 27997
    ⊞ META-INF 17267
      > system 28112
    ⊟ > WEB-INF 28190
        ⊞ classes
        ⊟ jsp 28190
            ⊟ common 28168
                browserCompatibleCheck.jsp 16620
                hiddenValues.jsp 17015
                newfooter.jsp 27443
                newheader.jsp 25855
                newleftMenu.jsp 28168
                newtopNavigationPanel.jsp 28168
                pageLoad.jsp 17015
                processing-bar.jsp 17040
                responseMsg.jsp 25594
                responseMsgPopUp.jsp 25594
```

– **jsp** contains jsp pages sliced based on modules.



```
⊟ > modules 28190
  ⊟ > servermanager 28190
    ⊞ buildscripts 24095
    ⊞ > config 28178
    ⊞ docs 28285
    ⊟ > master_data 28157
        export_TBLMACCESSGROUP.csv 26722
        export_TBLMACTION_3156.csv 27738
        export_TBLMACTION_MED_2759.csv 27617
        export_TBLMACTION.csv 27803
        export_TBLMAGENTTYPE.csv 26988
        export_TBLMAUDITACTIVITY.csv 27772
        export_TBLMAUDITENTITY.csv 26717
        export_TBLMAUDITSUBENTITY.csv 27772
        export_TBLMBUSINESSMODEL.csv 28123
        export_TBLMBUSINESSMODULE_3156.csv 27738
        export_TBLMBUSINESSMODULE_MED_2759.csv 27617
        export_TBLMBUSINESSMODULE.csv 28123
        export_TBLMBUSINESSSUBMODULE_3156.csv 27738
```

– **masterdata** contains csv for master data to be inserted in db.

Step 3. Module Creation

1) First of all identify the Menu, Sub-Menu (Tab) and user actions from Wireframe and then add entries of Module, Sub-Module and Action in export_TBLMBUSINESSMODULE.csv, export_TBLMBUSINESSSUBMODULE_3156.csv and export_TBLMACTION.csv respectively. Then fire the ANT build using liquibase_build_export.xml.

2) Then give action rights to the Admin Staff. While development we can add data into export_TBLTACCESSGROUPACTIONREL.csv. *Then fire the ANT build using liquibase_build_export.xml.*

3) To add any system paramter, add data in export_TBLMSYSTEMPARAMETER.csv .*Then fire the ANT build using liquibase_build_export.xml.*

4) Create Model

5) Create interface DAO that will extend GenericDAO<T>

6) Create class DAOImpl that will extend GenericDAOImpl<T> and implement interface DAO (Step 5 DAO).

7) Create interface Service

8) Create class ServiceImpl that will implement Service (Step 7 Service).

9) Create class Controller that will extend BaseController.

10) Create class Validator (if any). If any regex needs to be added, then add that Regex to the export_TBLMREGEX.csv and if any range validation, add it in export_TBLMVALIDATIONRANGE.csv. *Then fire the ANT build using liquibase_build_export.xml.*

While using @Validated, the sequence of the BindingResult has to be next to the

@Validated Annotation parameter. Eg:-

```
        @Validated   @ModelAttribute(FormBeanConstants.ACCESS_GROUP_FORM_BEAN)
AccessGroup accessGroup ,
        BindingResult result,
        ...,
        ...
```

11） Add the package name to register the Model for hibernate mapping in db- context.xml:

```
        <beans:property name="packagesToScan" value="com.elitecore.sm.config.model,
    com.elitecore.sm.iam.model, com.elitecore.sm.systemparam.model" />
```

12） Add the package name for component scanning to register DAOImpl, ServiceImpl, Controller in servlet-context.xml:

```
        <context:component-scan base-package="com.elitecore.sm.config.dao"/>
<context:component-scan base-package="com.elitecore.sm.config.service" />

        <context:component-scan base- package="com.elitecore.sm.systemparam.controller" />

            <context:component-scan base- package="com.elitecore.sm.systemparam.validator"
    />
```

13） Create View (jsp page).

- While using jquery, we need to add the following statement..

```
<script src="js/jquery-ui.js"></script>
```

- To add header and footer,

```
<jsp:include    page="../common/newheader.jsp"></jsp:include>

<jsp:include    page="../common/newfooter.jsp"></jsp:include>
```

- Use of Custom Tags:

1. If needed, create a new .tag file with your custom name at location

   -/WEB-INF/tags

   - You can add attributes which will be pass from JSP.

   ```
   <%@ attribute name="type" required="true" %>
   <%@ attribute name="inputClassName" required="false" %>
   ```

   - You can use the different taglibs in your custom tag file.

   ```
   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
   <%@taglib uri="http://www.springframework.org/tags" prefix="spring"%>
   ```

2. In JSP,

   - Declare your tag at top:

```
<%@ taglib tagdir="/WEB-INF/tags" prefix="elitecore" %>
```

- Usage:

```
        <elitecore:inputHTML type="text" name="username" id="username"
i18NCode="index.page.username" inputClassName="form-control"
></elitecore:inputHTML>
```

■ Points to be kept in mind:

1. Don't use flush or clear in hibernate session.

2. If you want to trace spring web exception then in log4j.xml , change log level "DEBUG".

```
<logger name="org.springframework.web">
        <!-- To get the spring web exceptions -->
        <level value="DEBUG" />
        <appender-ref ref="sm-spring" />
</logger>
```

3. While enctype="multipart/form-data" use then not able to get value of hidden parameters using request.getParameter("name"), it always return Null.

4. If the spring config executes the config initializer 2 time using thread:

- localhost-startStop-1
- http-bio-8080-exec-3

Then reason can be there will be some Bean Initializing Problem or Autowiring problem. So spring retries to initialize the config.

5. After Some attempts if fail to get value of hidden parameters and requests are discarded then may be problem is increase request size on every attempt.

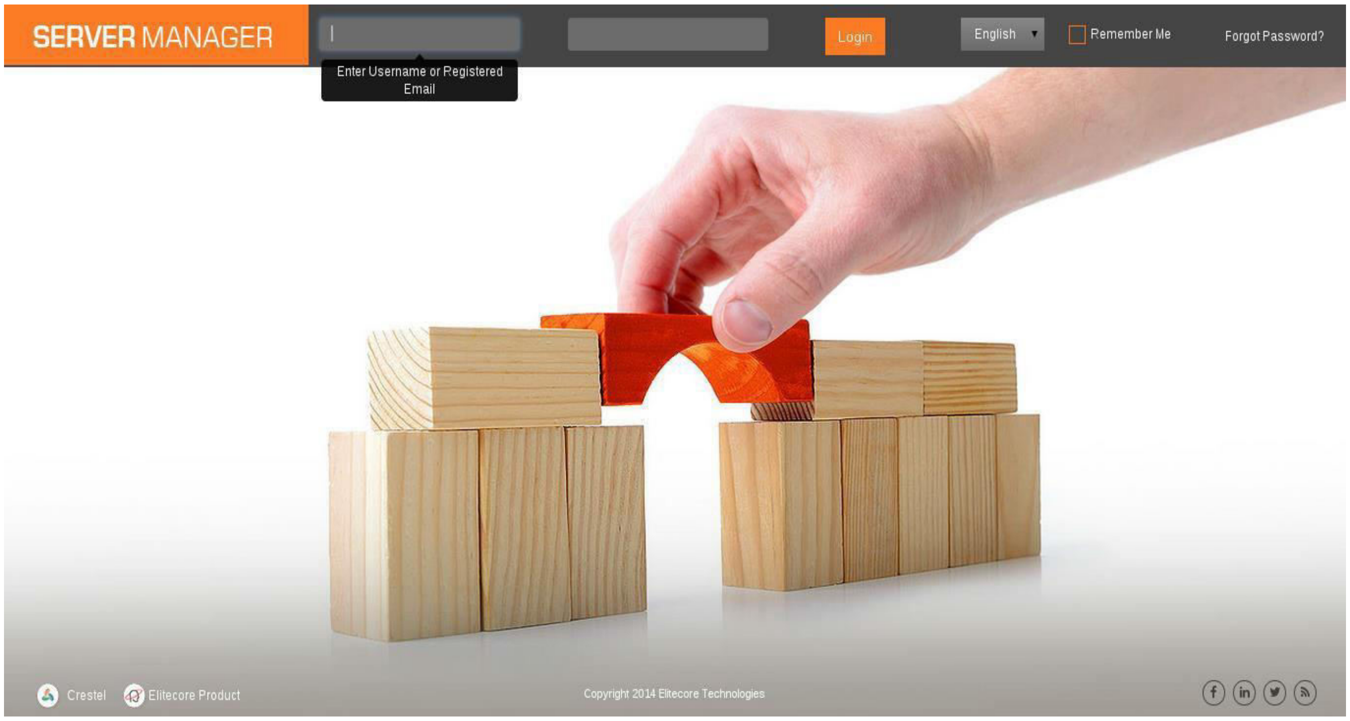
check Http request information using mozilla firefox add-on named "HTTPFOX".

Step 4. Flow between Browser and Controller

4.1) User enters the URL:

http://localhost:9191/CGFServerManager/
http://localhost:9191/CGFServerManager/welcome

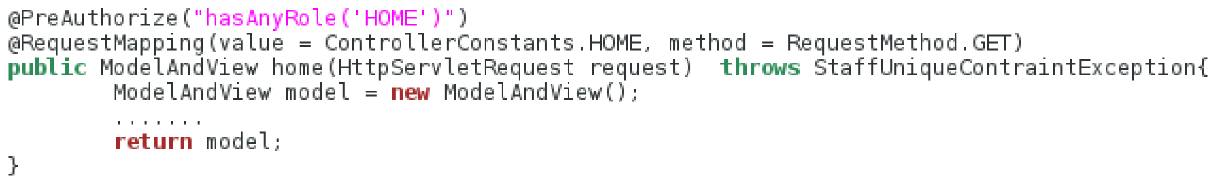
It will invoke the HomeController method, where the */* or */welcome* url is mapped. And user will be redirected to the Login Page.



4.2) User enters credentials and on Submit the request is sent to the *j_spring_security_check* which is part of Spring Security. Internally it invokes the *LimitLoginAuthenticationServiceImpl#authenticate* to authenticating the user credentials from *SpringLoginServiceImpl#loadUserByUsername*.

If username is available in database, User model of Spring is returned duly filled with Authorities user has. So that ACL is handed over to Spring Security rather managing at our end. After successful authentication user will be redirected to the

/home (*HomeController#home*).



```
@PreAuthorize("hasAnyRole('HOME')")
@RequestMapping(value = ControllerConstants.HOME, method = RequestMethod.GET)
public ModelAndView home(HttpServletRequest request)  throws StaffUniqueContraintException{
        ModelAndView model = new ModelAndView();
        .......
        return model;
}
```

@PreAuthorize – will check that logged-in user has authority to call this function or not. If not, then it will throw AccesDeniedException which will redirect user to error/403 (*ExceptionHandlingController#handleAccessDeniedException*)

@RequestMapping – value is the url which we want to make this method call and method depict by which http method can this method be called.

ModelAndView – this will have:

- View Name (eg: for login.jsp, view name would be *login*).

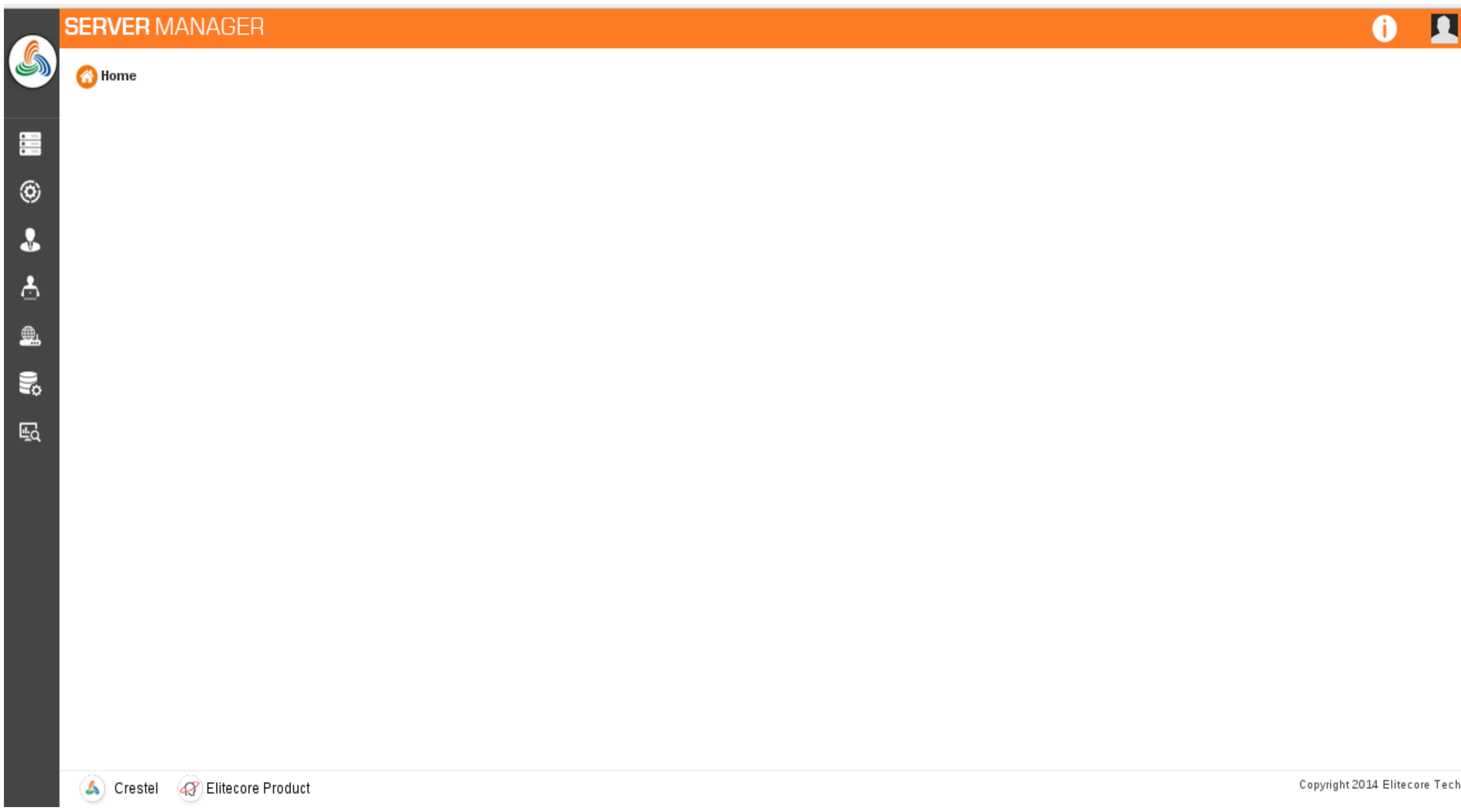
```
model.setViewName(ViewNameConstants.HOME_PAGE);
```

- Set of objects in form of key-value

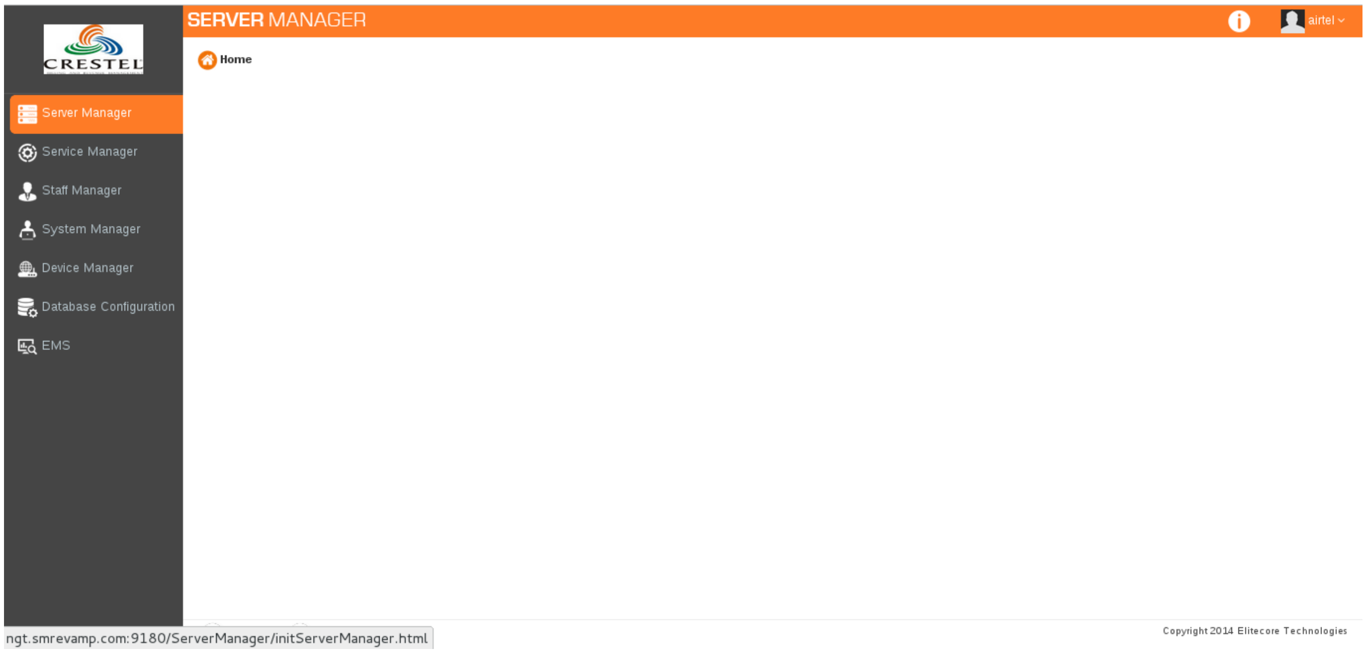
```
model.addObject("username",

springUserBean.getUsername());
```

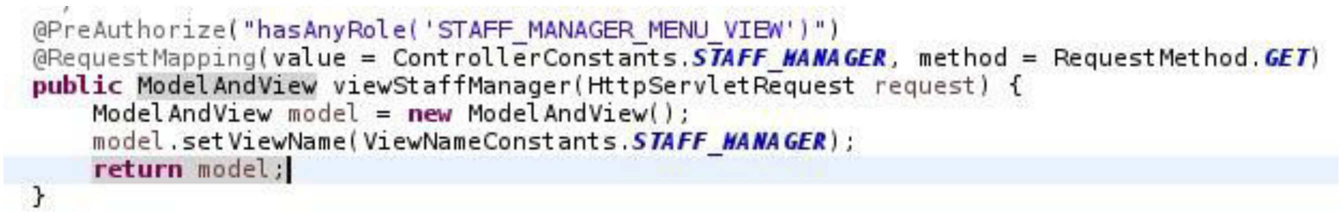
Home Page View



Left Menu View

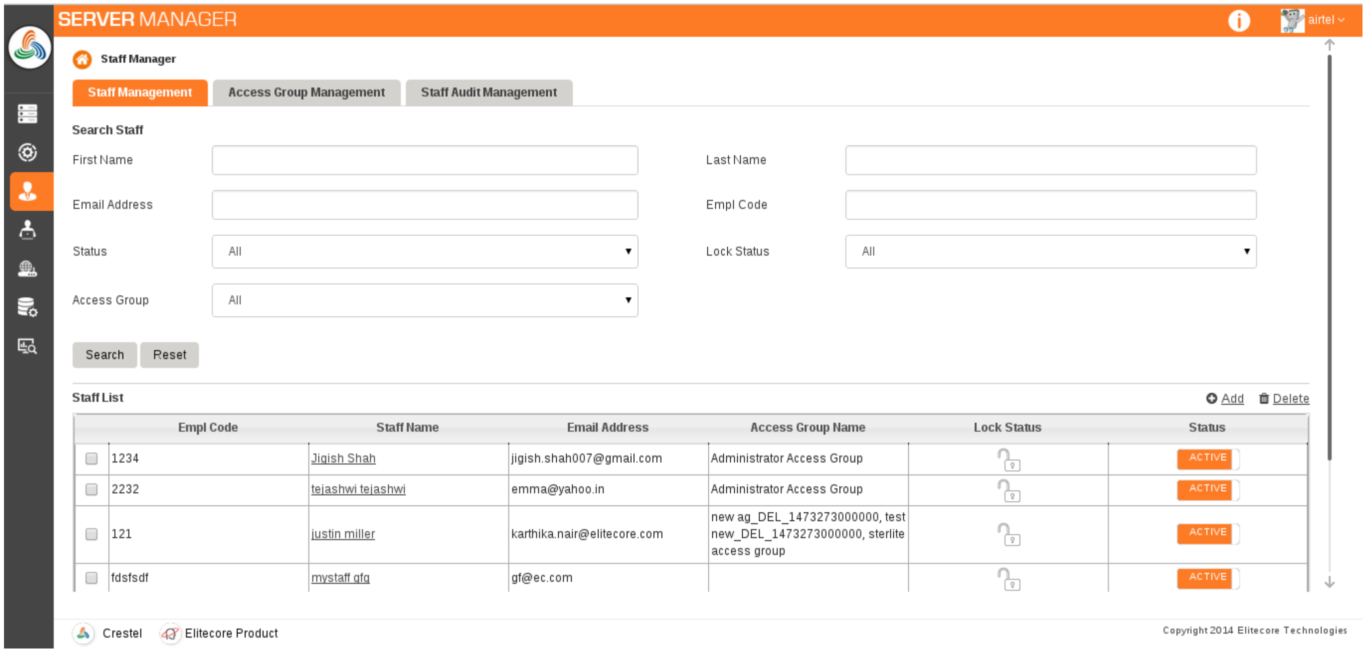


4.3) When user clicks “Staff Manager” from left menu, the /staffManager (StaffController#viewStaffManager) will be invoked:



```
@PreAuthorize("hasAnyRole('STAFF_MANAGER_MENU_VIEW')")
@RequestMapping(value = ControllerConstants.STAFF_MANAGER, method = RequestMethod.GET)
public ModelAndView viewStaffManager(HttpServletRequest request) {
    ModelAndView model = new ModelAndView();
    model.setViewName(ViewNameConstants.STAFF_MANAGER);
    return model;
}
```

This will redirect user to view iam/staffManager.


SERVER MANAGER

Staff Manager

Staff Management | Access Group Management | Staff Audit Management

Search Staff

First Name		Last Name	
Email Address		Empl Code	
Status	All	Lock Status	All
Access Group	All		

Search Reset

Staff List — Add Delete

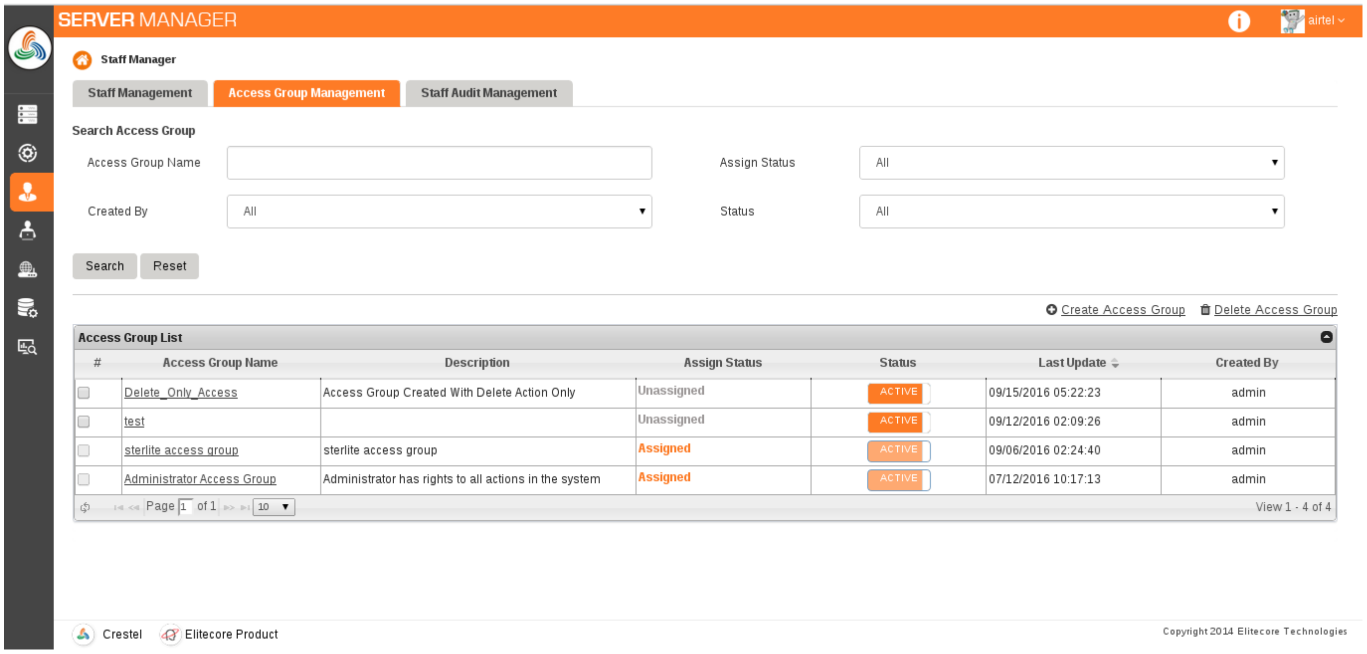
	Empl Code	Staff Name	Email Address	Access Group Name	Lock Status	Status
☐	1234	Jigish Shah	jigish.shah007@gmail.com	Administrator Access Group		ACTIVE
☐	2232	tejashwi tejashwi	emma@yahoo.in	Administrator Access Group		ACTIVE
☐	121	justin miller	karthika.nair@elitecore.com	new ag_DEL_1473273000000, test new_DEL_1473273000000, sterlite access group		ACTIVE
☐	fdsfsdf	mystaff gfg	gf@ec.com			ACTIVE

Crestel Elitecore Product

Copyright 2014 Elitecore Technologies

Step 5. Flow between Controller-Service-Dao

5.1) User clicks "Access Group Management" Tab


SERVER MANAGER

Staff Manager

Staff Management | Access Group Management | Staff Audit Management

Search Access Group

Access Group Name		Assign Status	All
Created By	All	Status	All

Search Reset

Create Access Group Delete Access Group

Access Group List

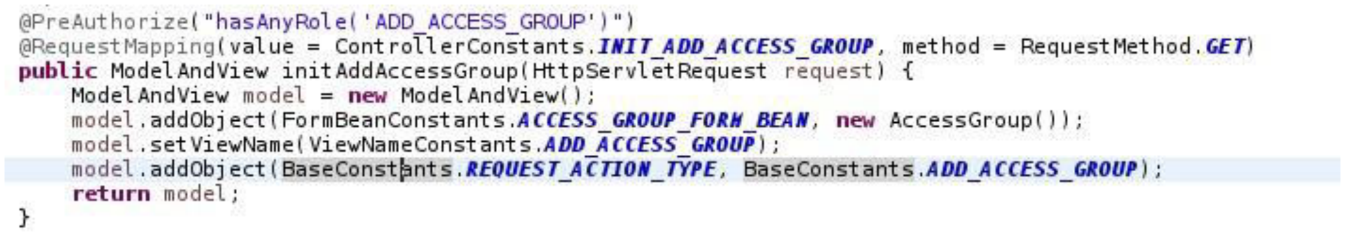
#	Access Group Name	Description	Assign Status	Status	Last Update	Created By
☐	Delete_Only_Access	Access Group Created With Delete Action Only	Unassigned	ACTIVE	09/15/2016 05:22:23	admin
☐	test		Unassigned	ACTIVE	09/12/2016 02:09:26	admin
☐	sterlite access group	sterlite access group	Assigned	ACTIVE	09/06/2016 02:24:40	admin
☐	Administrator Access Group	Administrator has rights to all actions in the system	Assigned	ACTIVE	07/12/2016 10:17:13	admin

Page 1 of 1 10 View 1 - 4 of 4

Crestel Elitecore Product

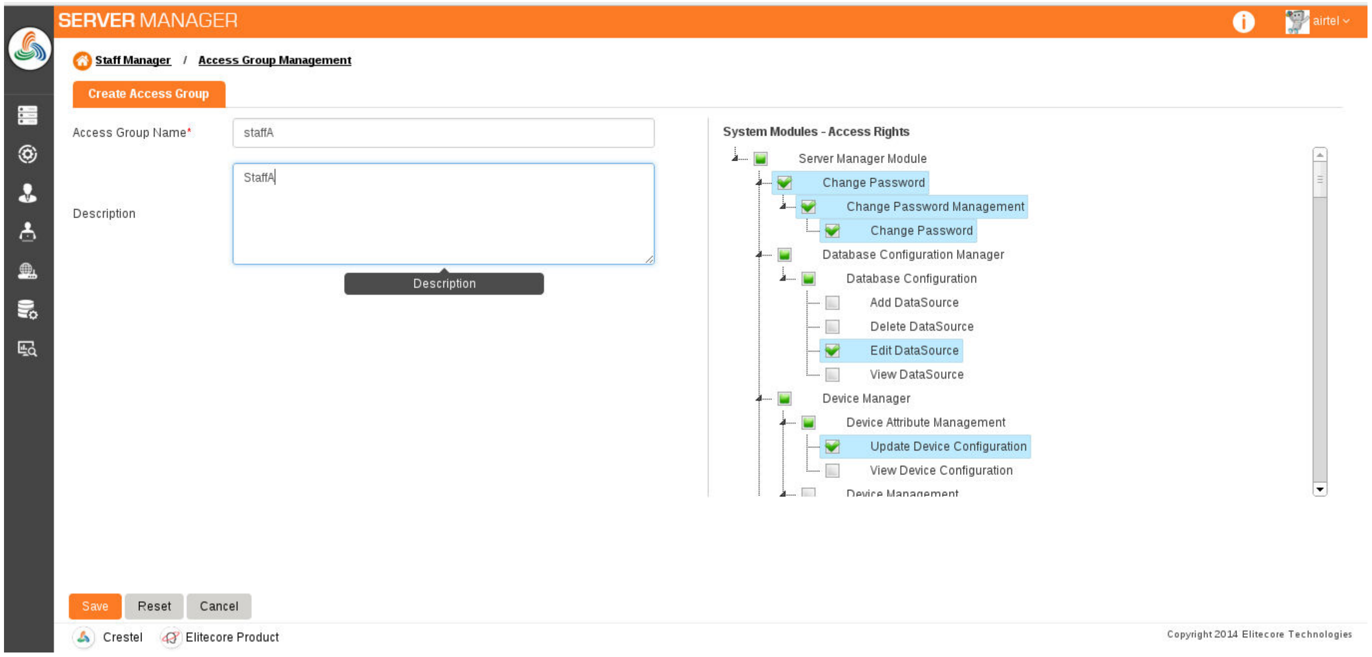
Copyright 2014 Elitecore Technologies

5.2) User clicks on “Create Access Group”. This will invoke /initAddAccessGroup (AccessGroupController#initAddAccessGroup)



```
@PreAuthorize("hasAnyRole('ADD_ACCESS_GROUP')")
@RequestMapping(value = ControllerConstants.INIT_ADD_ACCESS_GROUP, method = RequestMethod.GET)
public ModelAndView initAddAccessGroup(HttpServletRequest request) {
    ModelAndView model = new ModelAndView();
    model.addObject(FormBeanConstants.ACCESS_GROUP_FORM_BEAN, new AccessGroup());
    model.setViewName(ViewNameConstants.ADD_ACCESS_GROUP);
    model.addObject(BaseConstants.REQUEST_ACTION_TYPE, BaseConstants.ADD_ACCESS_GROUP);
    return model;
}
```

Here we set the AccessGroup Object to map to the addAccessGroup.jsp page form.



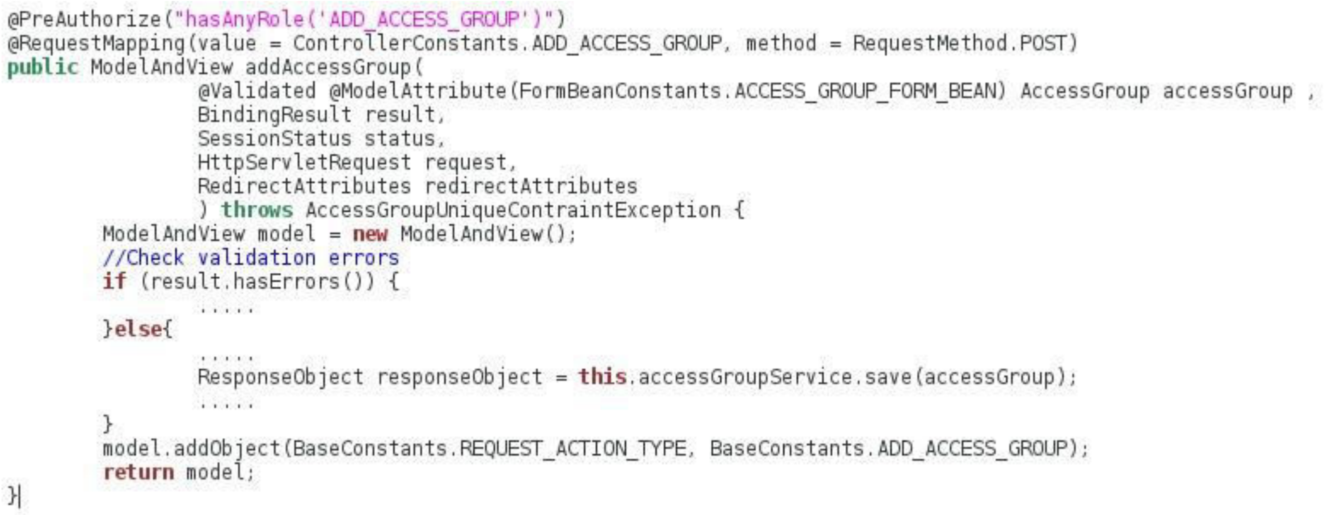
Access Group Name*: staffA

Description: StaffA

System Modules - Access Rights

- ✓ Server Manager Module
  - ✓ Change Password
    - ✓ Change Password Management
      - ✓ Change Password
  - ✓ Database Configuration Manager
    - ✓ Database Configuration
      - ☐ Add DataSource
      - ☐ Delete DataSource
      - ✓ Edit DataSource
      - ☐ View DataSource
  - ✓ Device Manager
    - ✓ Device Attribute Management
      - ✓ Update Device Configuration
      - ☐ View Device Configuration
    - ☐ Device Management

5.3) User fills details and click on Save. It will invoke addAccessGroup (AccessGroupController#addAccessGroup).



```
@PreAuthorize("hasAnyRole('ADD_ACCESS_GROUP')")
@RequestMapping(value = ControllerConstants.ADD_ACCESS_GROUP, method = RequestMethod.POST)
public ModelAndView addAccessGroup(
                @Validated @ModelAttribute(FormBeanConstants.ACCESS_GROUP_FORM_BEAN) AccessGroup accessGroup ,
                BindingResult result,
                SessionStatus status,
                HttpServletRequest request,
                RedirectAttributes redirectAttributes
                ) throws AccessGroupUniqueContraintException {
        ModelAndView model = new ModelAndView();
        //Check validation errors
        if (result.hasErrors()) {
                .....
        }else{
                .....
                ResponseObject responseObject = this.accessGroupService.save(accessGroup);
                .....
        }
        model.addObject(BaseConstants.REQUEST_ACTION_TYPE, BaseConstants.ADD_ACCESS_GROUP);
        return model;
}
```

AccessGroup - Model will automatically be set by the spring.

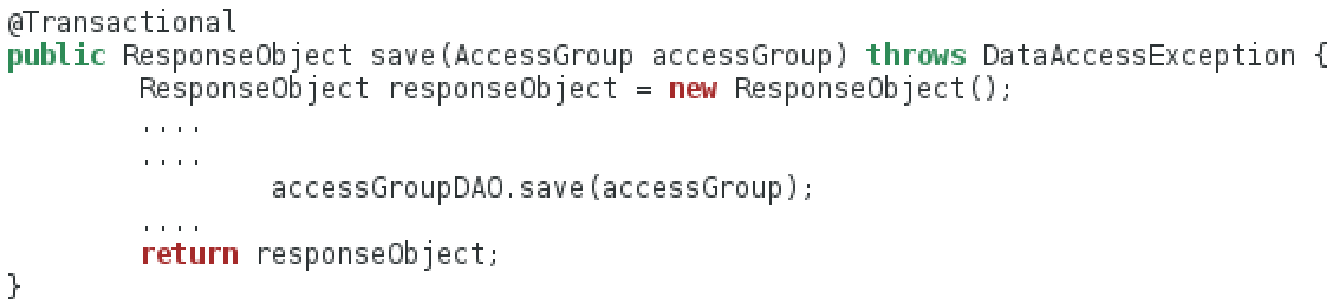
@Validated - Will call the AccessGroupValidator#validate for validation.

BindingResult - This parameter should always be next to the @Validated parameter. This will contain errors that are set by Validator class.

5.4) In above code of addAccessGroup:

this.accessGroupService.save(accessGroup);

will invoke the AccessGroupServiceImpl#save

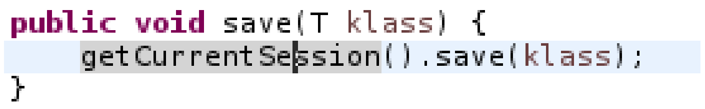


```
@Transactional
public ResponseObject save(AccessGroup accessGroup) throws DataAccessException {
        ResponseObject responseObject = new ResponseObject();
        ....
        ....
                accessGroupDAO.save(accessGroup);
        ....
        return responseObject;
}
```

5.5) In above code of save:

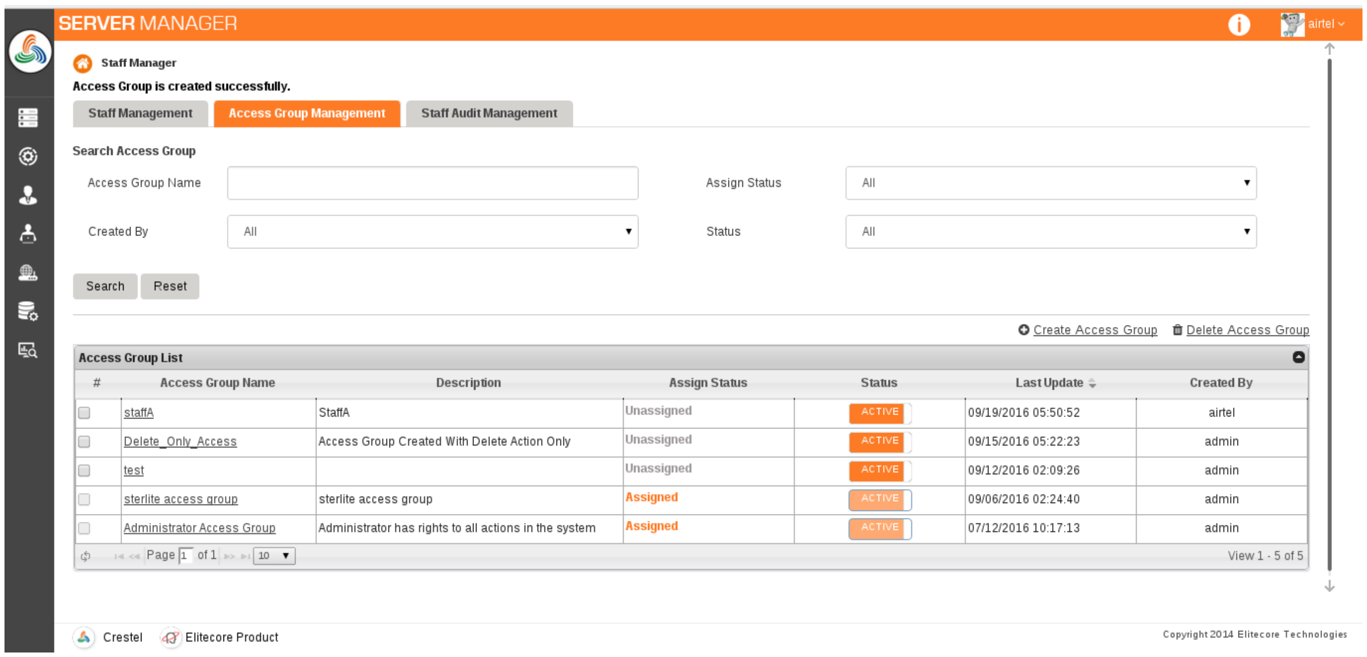
accessGroupDAO.save(accessGroup);

will invoke the AccessGroupDAOImpl->GenericDAOImpl<T>#save



```
public void save(T klass) {
    getCurrentSession().save(klass);
}
```

Common method that can be used for all models are kept under GenericDAOImpl<T>.



SERVER MANAGER

airtel

Staff Manager

Access Group is created successfully.

Staff Management | Access Group Management | Staff Audit Management

Search Access Group

Access Group Name		Assign Status	All
Created By	All	Status	All

Search Reset

Create Access Group | Delete Access Group

Access Group List

#	Access Group Name	Description	Assign Status	Status	Last Update	Created By
	staffA	StaffA	Unassigned	ACTIVE	09/19/2016 05:50:52	airtel
	Delete_Only_Access	Access Group Created With Delete Action Only	Unassigned	ACTIVE	09/15/2016 05:22:23	admin
	test		Unassigned	ACTIVE	09/12/2016 02:09:26	admin
	sterlite access group	sterlite access group	Assigned	ACTIVE	09/06/2016 02:24:40	admin
	Administrator Access Group	Administrator has rights to all actions in the system	Assigned	ACTIVE	07/12/2016 10:17:13	admin

Page 1 of 1 10 View 1 - 5 of 5

Crestel Elitecore Product

Copyright 2014 Elitecore Technologies

So, out flow depicts:

1) **View** – UI part that will be displayed to user.

2) **Controller** – Mapping the request url to forward request and validate the data.

3) **Service** – Business Logic and transaction management of hibernate.

4) **Dao** – Interaction with database.

****