**Create Grails app:**

**grails create-app app\_name**

**Create Domain layer:**

As we are implementing a web application for handling Students, let's start with generating a domain class called Student:

**grails create-domain-class com.kht.grails.Student**

add the firstName and lastName properties to it

Grails applies its conventions and will set up an object-relational mapping for all classes located in grails-app/domain directory.

Moreover, thanks to the GormEntity trait, all domain classes will have access to all CRUD operations

**Create Service Layer:**

Our application will handle the following use cases:

* Viewing a list of students
* Creating new students
* Removing existing students

Let's implement these use cases. Generate a service class:

**grails create-service com.kht.grails.Student**

grails-app/services directory, find our newly created service in the appropriate package and add all necessary methods

**Note: that services don't support transactions by default. We can enable this feature by adding the @Transactional annotation to the class.**

**Controller Layer:**

In order to make the business logic available to the UI, let’s create a StudentController by invoking the following command:

**grails create-controller com.kht.grails.Student**

By default, Grails injects beans by names. It means that we can easily inject the StudentService singleton instance into our controller by declaring an instance variable called studentsService.

We can now define actions for reading, creating and deleting students.

By convention, the index() action from this controller will be mapped to the URI /student/index, the show() action to /student/show and so on.

**View Layer:**

GSP guides: - <https://gsp.grails.org/latest/guide/tags.html>

Having set up our controller actions, we can now proceed to create the UI views. We will create three Groovy Server Pages for listing, creating and removing Students.

Genrate views from cli based on domain: -

**generate-views com.kht.student**

By convention, Grails will render a view based on controller name and action. For example, the index() action from StudentController will resolve to /grails-app/views/student/index.gsp

Let's start with implementing the view /grails-app/views/student/index.gsp, which will display a list of students. We'll use the tag <f:table/> to create an HTML table displaying all students returned from the index() action in our controller.

By convention, when we respond with a list of objects, Grails will add the “List” suffix to the model name so that we can access the list of student objects with the variable studentList:

<!DOCTYPE html>

<html>

<head>

<meta name="layout" content="main" />

</head>

<body>

<div class="nav" role="navigation">

<ul>

<li><g:link class="create" action="create">Create</g:link></li>

</ul>

</div>

<div id="list-student" class="content scaffold-list" role="main">

<f:table collection="${studentList}"

properties="['firstName', 'lastName']" />

</div>

</body>

</html>

We'll now proceed to the view /grails-app/views/student/create.gsp, which allows the user to create new Students. We'll use the built-in <f:all/> tag, which displays a form for all properties of a given bean:

<!DOCTYPE html>

<html>

<head>

<meta name="layout" content="main" />

</head>

<body>

<div id="create-student" class="content scaffold-create" role="main">

<g:form resource="${this.student}" method="POST">

<fieldset class="form">

<f:all bean="student"/>

</fieldset>

<fieldset class="buttons">

<g:submitButton name="create" class="save" value="Create" />

</fieldset>

</g:form>

</div>

</body>

</html>

Finally, let's create the view /grails-app/views/student/show.gsp for viewing and eventually deleting students.

Among other tags, we'll take advantage of <f:display/>, which takes a bean as an argument and displays all its fields

<!DOCTYPE html>

<html>

<head>

<meta name="layout" content="main" />

</head>

<body>

<div class="nav" role="navigation">

<ul>

<li><g:link class="list" action="index">Students list</g:link></li>

</ul>

</div>

<div id="show-student" class="content scaffold-show" role="main">

<f:display bean="student" />

<g:form resource="${this.student}" method="DELETE">

<fieldset class="buttons">

<input class="delete" type="submit" value="delete" />

</fieldset>

</g:form>

</div>

</body>

</html>