Seeing your results

**Goals:** In this lab, you will learn how to create controller classes, which will take your app and help to connect it to a web interface.

**Instructions:**  
  
Right now you have domain classes, but no way of accessing them online. Luckily, grails comes with a quick way to visualize your classes. We’re now going to create controllers for our resources. To do this, open up the grails prompt, and enter in

grails> create-controller post

This command should create a file grails-app/controllers/zynxblog/PostController.groovy. It should contain the following content:

package zynxblog

class PostController {

   def index() { }

}

We’re going to have grails generate a scaffold for us. This will allow us to perform CRUD operations on our posts. Add the following line to the class declaration:

static scaffold = true

Also, delete the line with def index, otherwise you will get errors displaying your page. We are now ready to run the application for the first time. In your Grails prompt, type

grails> run-app

If you navigate to <http://localhost:8080/zynxblog/posts/> you should now see a list of all your posts, with buttons to add and delete posts.  
  
Now, we’re going to modify this class to better suit our needs. First, we’re going to create an *action* to list posts. Copy the following function into the PostController definition:

def list = {

   [posts:Post.list(sort:'lastUpdated',

                    order:'desc')]

}

This action renders the ‘list’ view, passing it the parameters posts and sort. As we’ll see in an upcoming lab, this will show the user a list of posts.  
  
Now we’re going to add the edit and view actions. Add the following code to the PostController class:

def edit = {

   def post = Post.get(params.id)

   if(!post) {

       post = new Post()

   }

   [post:post]

}

def view = {

   [post:Post.get(params.id)]

}

Finally, we add code for the save action, which handles saving the new post form.

def save = {

   def post = loadPost(params.id)

   post.properties = params

   if(post.save()) {

       redirect(action:'list')

   } else {

       render(view:'edit', model:[post:post])

   }

}

private loadPost(id) {

   def post = new Post();

   if(id) {

       post = Post.get(id)

   }

   return post

}

That’s it! That completes our PostController class. HOWEVER, if you run-app right now, it won’t work. We’ll fix that soon!