Getting comfortable with URLs

**Goal:** In this lab, we will implement the URL scheme we designed in lab 1 in grails.  
  
**Instructions**  
  
Open up the grails-app/conf/UrlMappings.groovy file in your favorite IDE. This file contains all the URLs for your blog. The default Grails URL Mapping looks something like this:  
  
class UrlMappings {  
  
       static mappings = {  
               "/$controller/$action?/$id?"{  
                       constraints {  
                               // apply constraints here  
                       }  
               }  
  
               "/"(view:"/index")  
               "500"(view:'/error')  
       }  
}  
  
We will be adding our mappings to the mappings variable. Adding a new mapping simply involves adding a specially formatted string into the mappings variable.   
  
All path elements beginning with a $ are converted into parameters, which means they will be available in the params object within your views. When you add a ‘?’ to the end of a variable name in a URL mapping, it tells grails that that particular element is optional. So, given the current mapping, the following resources will all match the defined mappings:

1. /
2. /someController/
3. /someController/someAction/
4. /someController/someAction/someId
5. /someOtherController/someAction/someOtherId

The controller and action parameters both have special meanings, which we will use in subsequent labs. For our purposes now, it is safe to ignore them.  
  
The special “500” mapping tells grails which view to display when there is a 500 (Internal Server Error) error.  Notice how it does not start with a ‘/’. If you don’t start your mappings with a ‘/’ grails will think that you are trying to define an error code mapping, and will complain at you when it can’t convert your path to an integer.  
  
Now that we’re finally ready to create our mappings, add the following line to your mappings variable:

“/posts/$id”

In grails, it is customary (but not required) to use the $id parameter to take the place of the resource identifier. If you need to have parameters for two identifiers, name the identifier of the most nested subresource id, and name the other ones something else. For example, if we wanted to add a URL for comments, which are a subresource of posts, we might add the following mapping:

“/posts/$postId/comments/$id”

Once you are done, comment out everything you just wrote. Since we haven’t yet set up any controllers, these mappings are currently useless!