# CDR Lite programmers guide

Programmers picking up this code for the first time should have a pretty easy time figuring out what goes where. I know because I am a programmer and I have watched others do the same.

The CDR project is written in groovy and Grails. [Groovy](http://www.groovy-lang.org/documentation.html) is the language and [Grails](https://grails.org/) is the framework. It is similar to, and based upon, the very successful [Ruby on Rails](http://rubyonrails.org/), but it is written in Java, compiles down into java bytecode, builds into a war file, and runs on an application ([tomcat](http://tomcat.apache.org/)) server.

In addition, a very nice feature of Grails, is that you can make a change, reload the web page, and see your changes immediately. There is no need to rebuild the war file and restart the tomcat or [jetty](http://www.eclipse.org/jetty/) application server on your personal workstation. This has been a major shortcoming of java for many years, but Grails fixes all of that. The technology behind the scenes which enables this functionality is called [spring loaded](https://github.com/spring-projects/spring-loaded), and it makes developing in java just as fast as developing in PHP or Python.

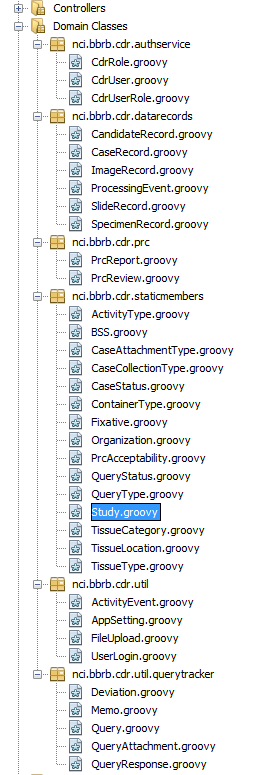
Grails and Groovy follow the time-honored software architecture of Model-View-Controller, or [MVC](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller).

## MVC Architecture in Grails:

Grails and Groovy embody the principle of “[convention over configuration](https://en.wikipedia.org/wiki/Convention_over_configuration)”

* Model: If there is a file called CaseRecord in the DomainClasses folder, you know it is a Domain class; grails will create a table for it in the database, and wire it up automatically using [hibernate](http://hibernate.org/). No configuration is required. Changes to Domain classes are automatically propagated to the table structure in the database, except you can’t delete a column.
* Controller: Similarly, if there is a file called CaseRecordController in the Controllers folder, grails will know that this is the controller for the CaseRecord Domain class. All of the methods in the controller need to have a corresponding view.
* View: Finally, if there is a file called list.gsp in the Views folder, under caseRecord, grails will know that this is the view for the list() method in the CaseRecord controller, and this view will be used to display the CaseRecord model, after executing the business logic in the list() method of the CaseRecordController.

As of today[[1]](#footnote-1), these are the Domain Classes in the CDR Lite:



For every Domain Class, there is an associated controller. For every method in each controller, there is an associated view with the same name.

The package names (they look like folders) you see here are more or less arbitrary. They were chosen to follow standard java naming conventions, as well as to help keep things organized.

If you want to add new functionality in groovy and grails, you do it by specifying your Domain class first. You can then right-click on “generate all” in the Netbeans IDE, and grails will generate a default set of controllers and views for your Domain class to give you basic “[CRUD](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete)” functionality.

Some important files to be aware of (These are all in the Configuration folder):

### Config.groovy:

An important setting in this file for security purposes is:

grails.plugin.springsecurity.rejectIfNoRule = true

This means that only URL’s that are specifically named in this folder will be allowed in the application. Whenever you add a Domain class, with associated controllers and views, the path needs to be added in config.groovy, or you will not be permitted to access it:

grails.plugin.springsecurity.controllerAnnotations.staticRules = [

//system setting controllers

'/user/\*\*': ['ROLE\_ADMIN','ROLE\_SUPER'],

'/role/\*\*': ['ROLE\_ADMIN','ROLE\_SUPER'],

'/userRole/\*\*': ['ROLE\_ADMIN','ROLE\_SUPER'],

'/activitycenter/\*\*': ['IS\_AUTHENTICATED\_FULLY']

]

### DataSource.groovy:

This is where you define the database connections. Usernames and passwords for database login are stored in clear text in this folder, so it is generally a good idea to keep it out of public code repositories like GitHub.

This is a good candidate for the .gitignore file.

If you add an Activity, you will have to update the ActivityEventController, as well as caHUB.js with the hard-coded name of the new activity.

The Study (‘BPS’ = Biospecimen Preservation Study) is hard-coded.

1. 11-Sep-2015 [↑](#footnote-ref-1)