BBRB Biorepositories and Biospecimen Research Branch

leidos
Leidos Biomedical Research, Inc.

IT-9011 CDR-Lite Programmers' Guide

Version 1.01

11/7/2016

Version History

Version	Author	Effective Date	Description of Change
1.00	William Lander	11/4/2016	Initial Version
1.01	William Lander	11/7/2016	Administrative Update

Table of Contents

1 Introduction

Leidos Biomedical Research, under the direction of the NCI BBRB, developed the CDR-Lite. CDR-Lite is a Web-based application, custom built to support specimen collection, clinical data entry, specimen logistics, and curation and aggregation of study data. Its capabilities, which reflect the needs of the supported projects, include the following:

- Allow remote users (e.g., researchers, support staff) to securely enter, revise, and review data about biospecimen collection through a standard (HTTPS) Web interface using a series of electronic forms with a sophisticated role-driven workflow
- Connect to remote systems via Web service application program interfaces, such as laboratory information management systems, whole-slide imaging systems, and molecular analysis systems
- Send automatic email alerts to communicate timely information to project managers and data analysts, letting them know when to start working on something
- Assist with quality assurance by auditing process flows through data management and pathology teams
- Controlling display of personally identifiable information (PII) based on user entitlements and roles

The Comprehensive Data Resource (CDR) project is written in Groovy and Grails. Groovy is the language, and Grails is the framework. Grails is similar to—and based upon—the very successful Ruby on Rails, but it is written in Java, compiles down into Java bytecode, builds into a WAR file, and runs on an application (Tomcat) server.

A nice feature of Grails is that you can make a change, r1eload the web page, and see your change immediately. You do not need to rebuild the WAR file and restart the Tomcat or Jetty application server on your personal workstation. This has been a major shortcoming of Java for many years, but Grails fixes it. The technology behind the scenes that enables this functionality is called "spring loaded," and it makes developing in Java just as fast as developing in PHP or Python.

1.1 Purpose of This Document

This document gives an overview of the process a programmer may go through in customizing CDR-Lite for a specific study. It assumes that the reader has some knowledge of the architecture and system design of the CDR-lite development (Documented elsewhere). Its intended audience is development, and DevOps teams.

2 MVC Architecture in Grails

Groovy and Grails follow the time-honored software architecture of model-view-controller, or MVC, and embody the principle of "convention over configuration." The following shows how the features of the code map into the MVC paradigm.

- 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. No configuration is required. Domain class changes automatically propagate to the table structure in the database, except that you cannot delete a field.
- View: 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. It will use this view to display the CaseRecord model after executing the business logic in the `list()` method of the `CaseRecordController`.
- Controller: Finally, 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.

2.1 Assumptions, Constraints, and Standards

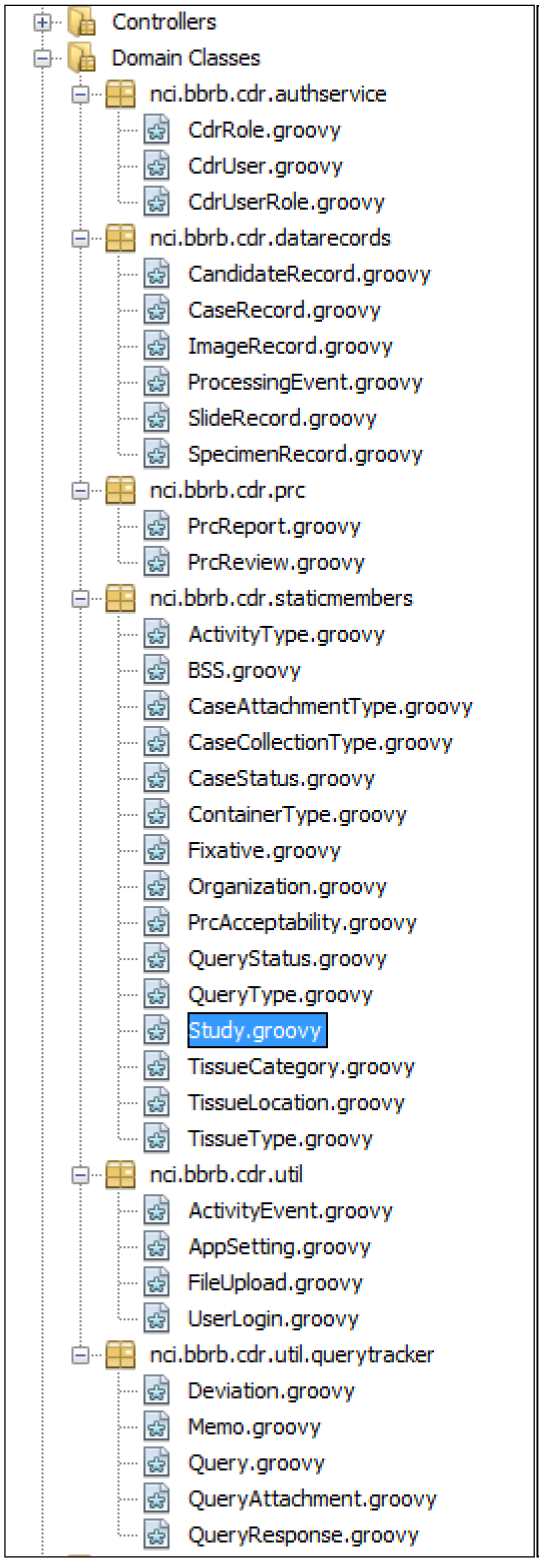
The CDR-Lite software is designed for maximum flexibility to meet changing requirements. As such, standard services (e.g., PostgreSQL database services, RESTful Web service interfaces, XML, and JSON data exchanges) in creating a framework into which a variety of documents may be "plugged-in."

The architecture of CDR-Lite reflects the need to store PII and PHI in an environment where not all users have a need to know all information. For this reason, it includes a variety of roles that can be assigned to users, limiting their ability to access sensitive information. Details of roles, privileges, and responsibilities are in the User guide.

2.2 Setting up the Environment

The CDR-lite was developed using the NetBeans IDE. Details of how to obtain the source code, setting up the various software tools, and installing the war file are` available in the document Building and deploying CDR-Lite.

These are the Domain Classes in the CDR-Lite:



- Controllers
- Domain Classes
  - nci.bbrb.cdr.authservice
    - CdrRole.groovy
    - CdrUser.groovy
    - CdrUserRole.groovy
  - nci.bbrb.cdr.datarecords
    - CandidateRecord.groovy
    - CaseRecord.groovy
    - ImageRecord.groovy
    - ProcessingEvent.groovy
    - SlideRecord.groovy
    - SpecimenRecord.groovy
  - nci.bbrb.cdr.prc
    - PrcReport.groovy
    - PrcReview.groovy
  - nci.bbrb.cdr.staticmembers
    - ActivityType.groovy
    - BSS.groovy
    - CaseAttachmentType.groovy
    - CaseCollectionType.groovy
    - CaseStatus.groovy
    - ContainerType.groovy
    - Fixative.groovy
    - Organization.groovy
    - PrcAcceptability.groovy
    - QueryStatus.groovy
    - QueryType.groovy
    - Study.groovy
    - TissueCategory.groovy
    - TissueLocation.groovy
    - TissueType.groovy
  - nci.bbrb.cdr.util
    - ActivityEvent.groovy
    - AppSetting.groovy
    - FileUpload.groovy
    - UserLogin.groovy
  - nci.bbrb.cdr.util.querytracker
    - Deviation.groovy
    - Memo.groovy
    - Query.groovy
    - QueryAttachment.groovy
    - QueryResponse.groovy

Every Domain Class has an associated controller. Every method in each controller has an associated view with the same name.

The package names you see here (they look like folders) are more or less arbitrary. The names follow standard Java naming conventions and help keep things organized.

If you are using the NetBeans integrated development environment (IDE) and you want to add new functionality in Groovy and Grails, first specify your Domain class. Then right-click on "generate all," and Grails will generate a default set of controllers and views for your Domain class to give you basic "CRUD" (create, read, update, and delete) functionality. Other IDEs may offer similar functionality.

Figure 1- Domain Classes in CDR-Lite

2.3 Configuration Folder Details

The `Configuration` folder contains some important files that you should be aware of:

- **Config.groovy**

  The file `config.groovy` contains an environments section. You may need to customize this section to describe your local system configuration. Because environments may be set up in so many ways, describing all possible configurations is not possible. Examples of configurations to customize include addresses of various services, such as email.

  CDR-Lite sends automated email messages at various times. See the CDR-Lite Architecture Guide for more details on this process. You will need to configure the email addresses for each local site.

  The following setting in this file is important for security purposes:

  ```
  grails.plugin.springsecurity.rejectIfNoRule = true
  ```

  This setting means that only URLs that are specifically named in this file will be allowed in the application. You can restrict URL access by user roles so that only a limited set of users may access certain pages. Roles are discussed in the CDR-Lite User Guide and the CDR-Lite Architecture Guide. Whenever you add a Domain class with associated controllers and views, you will need to add the path in `config.groovy`; otherwise, you will not be permitted to access the Domain class:

  ```
  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 file is where you define the database connections. Usernames and passwords for database login are stored in clear text in this file, 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.

- **ActivityEvent.groovy**

  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. Activities include adding support for a new feature, such as a new form. The CDR-Lite Architecture Guide discusses Activities in more detail.

The Biospecimen Preservation Study (BPS) is hard-coded in CDR-Lite. You can add studies using the mechanisms described in the CDR-Lite User Guide.