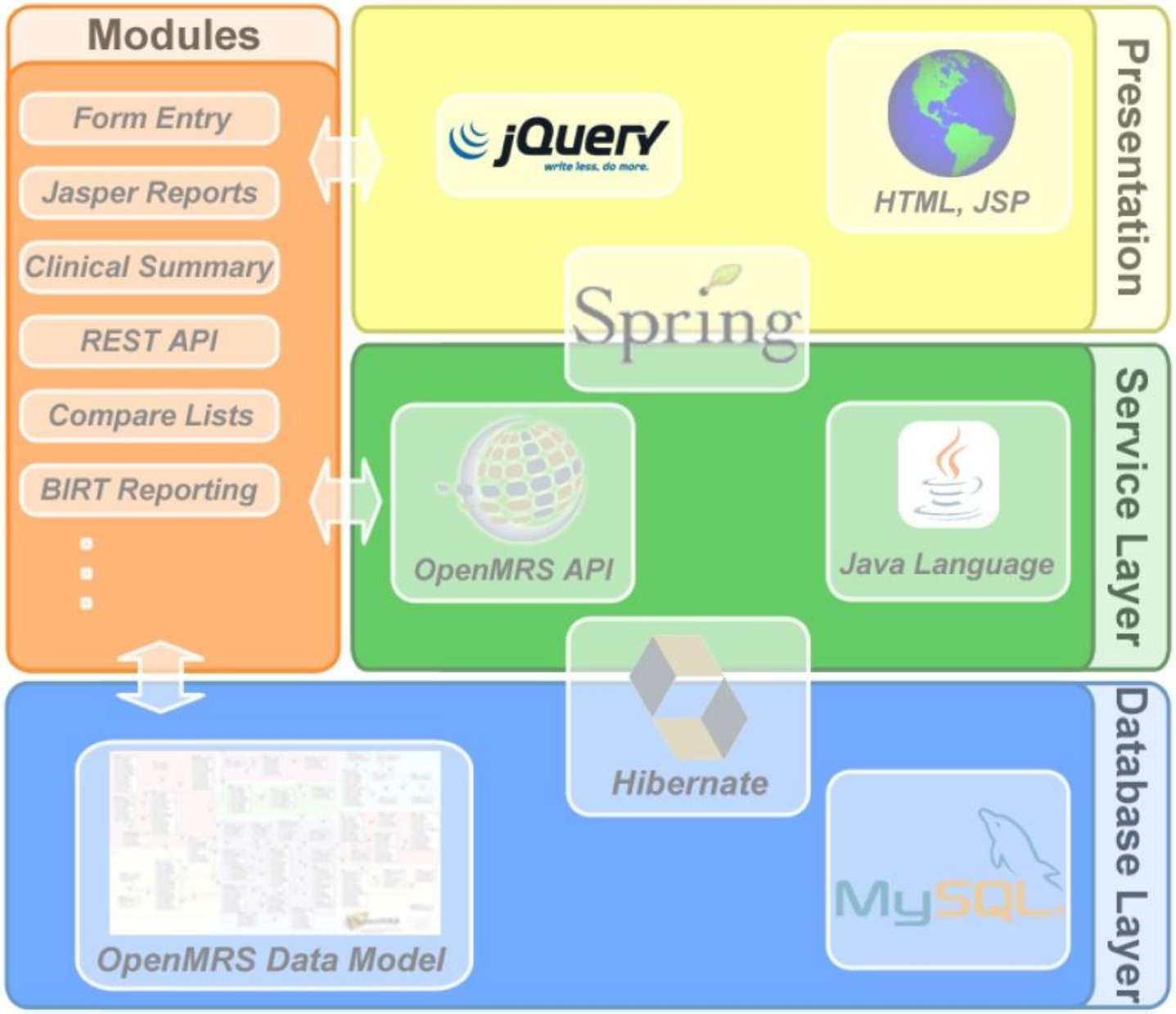
Understanding NCD 1.4

Written by Tom Hunter

- Foundation of OpenMRS
- Concept of OpenMRS Module
- Module layout: API, OMOD
- OpenMRS Database
- Efficient development of NCD 1.4
- How to Test NCD 1.4
- Use of Terser

Foundation of OpenMRS

- Overview of OpenMRS

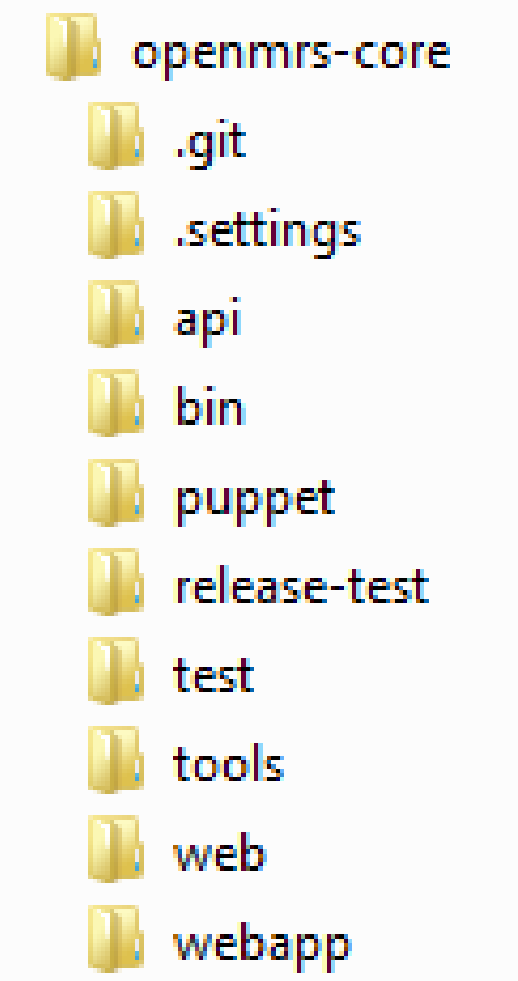


An overview of OpenMRS

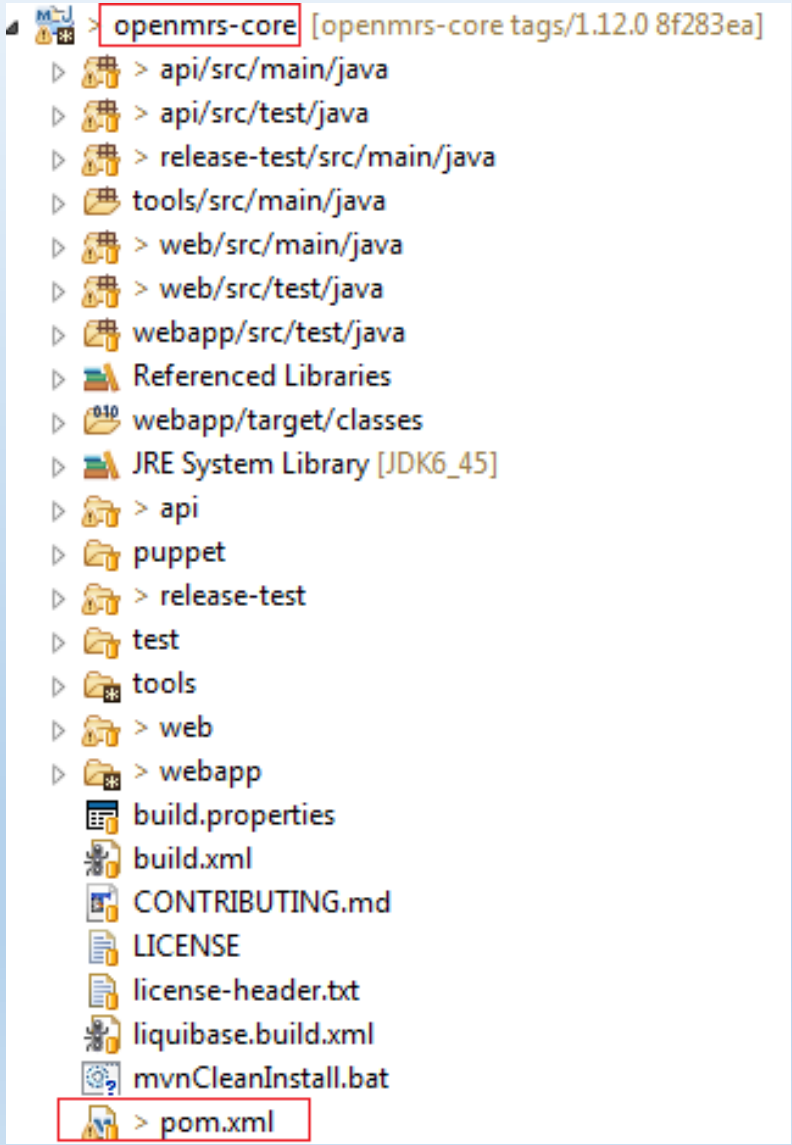
- NCD 1.4 is built targeting OpenMRS 1.12.0.
- Documentation: http://openmrs.org/
- Git Repos: https://github.com/openmrs/
  - *OpenMRS 1.7.1 was hosted on SVN, as was NCD 1.3.*
  - *NCD 1.4 is a migration from SVN to Git*
  - *NCD 1.4 is a migration from ANT to Maven*
  - *NCD 1.4 centralizes Java code into two projects: API and OMOD, while the earlier version had it scattered among several places.*

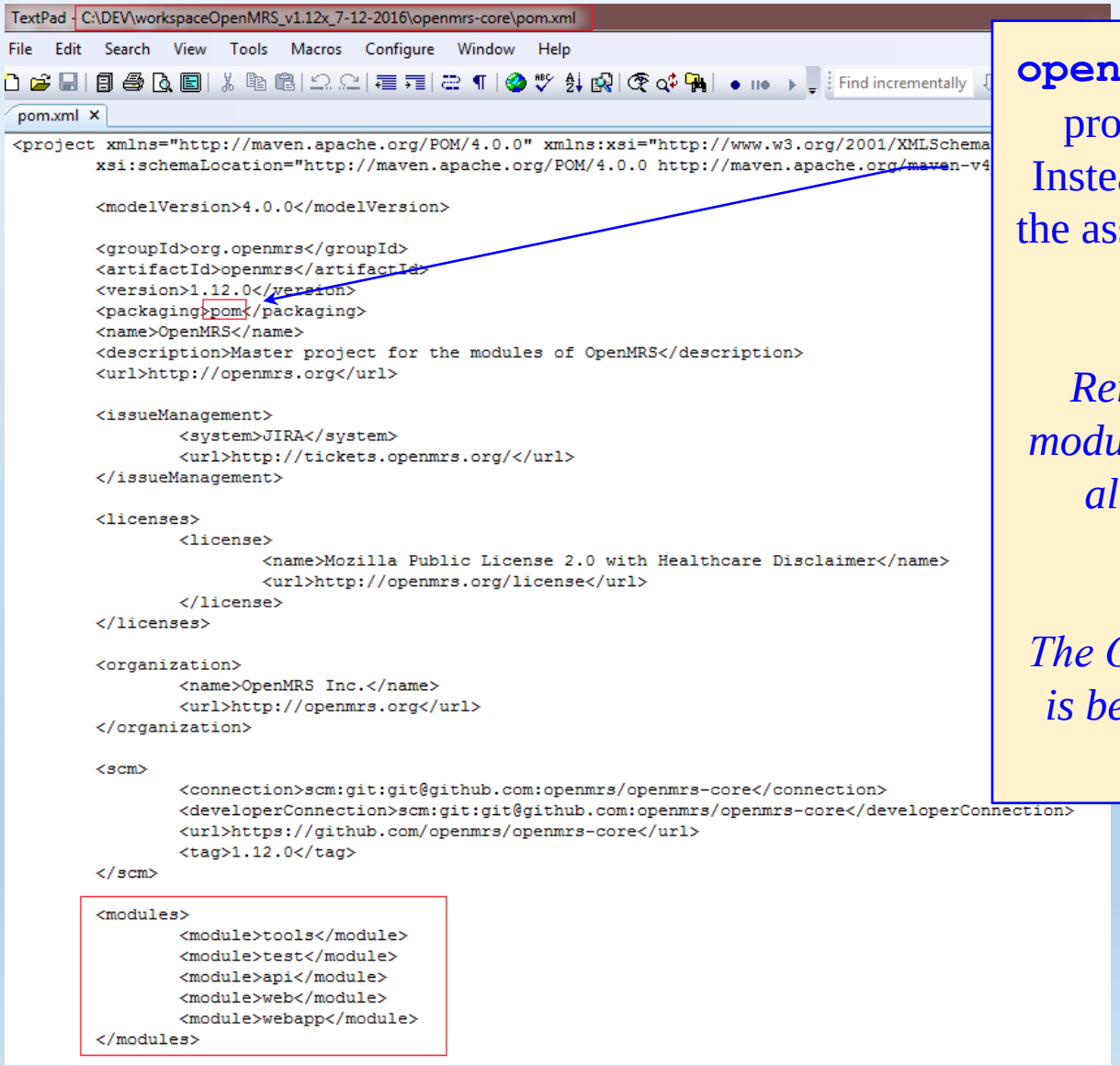
- **`openmrs-core`**—This contains the entirety of OpenMRS.
- **`openmrs-core`** is broken down into the standard Maven "parent" project and sub-projects that are "Maven modules"

(*Sorry the term "module" is overloaded.*)



• The top module `openmrs-core` has its own `pom.xml`.





`openmrs-core` does not produce any artifacts. Instead, it stage-manages the assembly of the Maven sub-modules.

Remember: a Maven module has no relation at all to an OpenMRS "Module".

The OpenMRS "Module" is better thought of as a "plugin".

- As much as possible, JAR dependencies should be placed in the top module **`openmrs-core`**'s **`pom.xml`**.

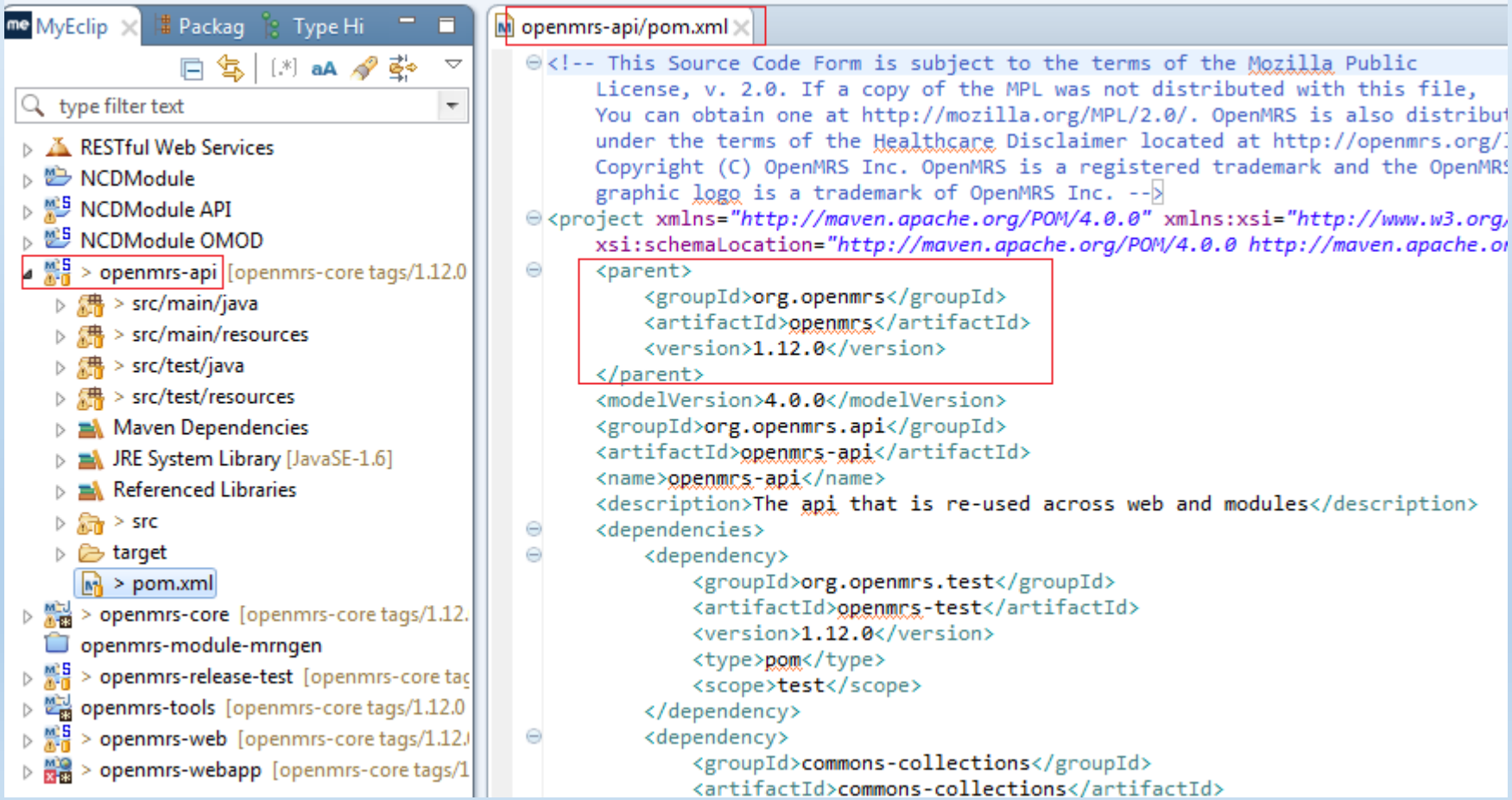
```xml
<!-- START: HAPI -->
<dependency>
    <groupId>ca.uhn.hapi</groupId>
    <artifactId>hapi-base</artifactId>
    <version>${hapi.version}</version>
</dependency>

<dependency>
    <groupId>ca.uhn.hapi</groupId>
    <artifactId>hapi-structures-v21</artifactId>
    <version>${hapi.version}</version>
</dependency>
<dependency>
    <groupId>ca.uhn.hapi</groupId>
    <artifactId>hapi-structures-v22</artifactId>
    <version>${hapi.version}</version>
</dependency>
<dependency>
    <groupId>ca.uhn.hapi</groupId>
    <artifactId>hapi-structures-v23</artifactId>
    <version>${hapi.version}</version>
</dependency>
```

```xml
<dependency>
    <groupId>ca.uhn.hapi</groupId>
    <artifactId>hapi-structures-v24</artifactId>
    <version>${hapi.version}</version>
</dependency>
<dependency>
    <groupId>ca.uhn.hapi</groupId>
    <artifactId>hapi-structures-v25</artifactId>
    <version>${hapi.version}</version>
</dependency>
<dependency>
    <groupId>ca.uhn.hapi</groupId>
    <artifactId>hapi-structures-v26</artifactId>
    <version>${hapi.version}</version>
</dependency>
<dependency>
    <groupId>ca.uhn.hapi</groupId>
    <artifactId>hapi-structures-v231</artifactId>
    <version>${hapi.version}</version>
</dependency>
<dependency>
    <groupId>ca.uhn.hapi</groupId>
    <artifactId>hapi-structures-v251</artifactId>
    <version>${hapi.version}</version>
</dependency>
<!-- END: HAPI -->
```

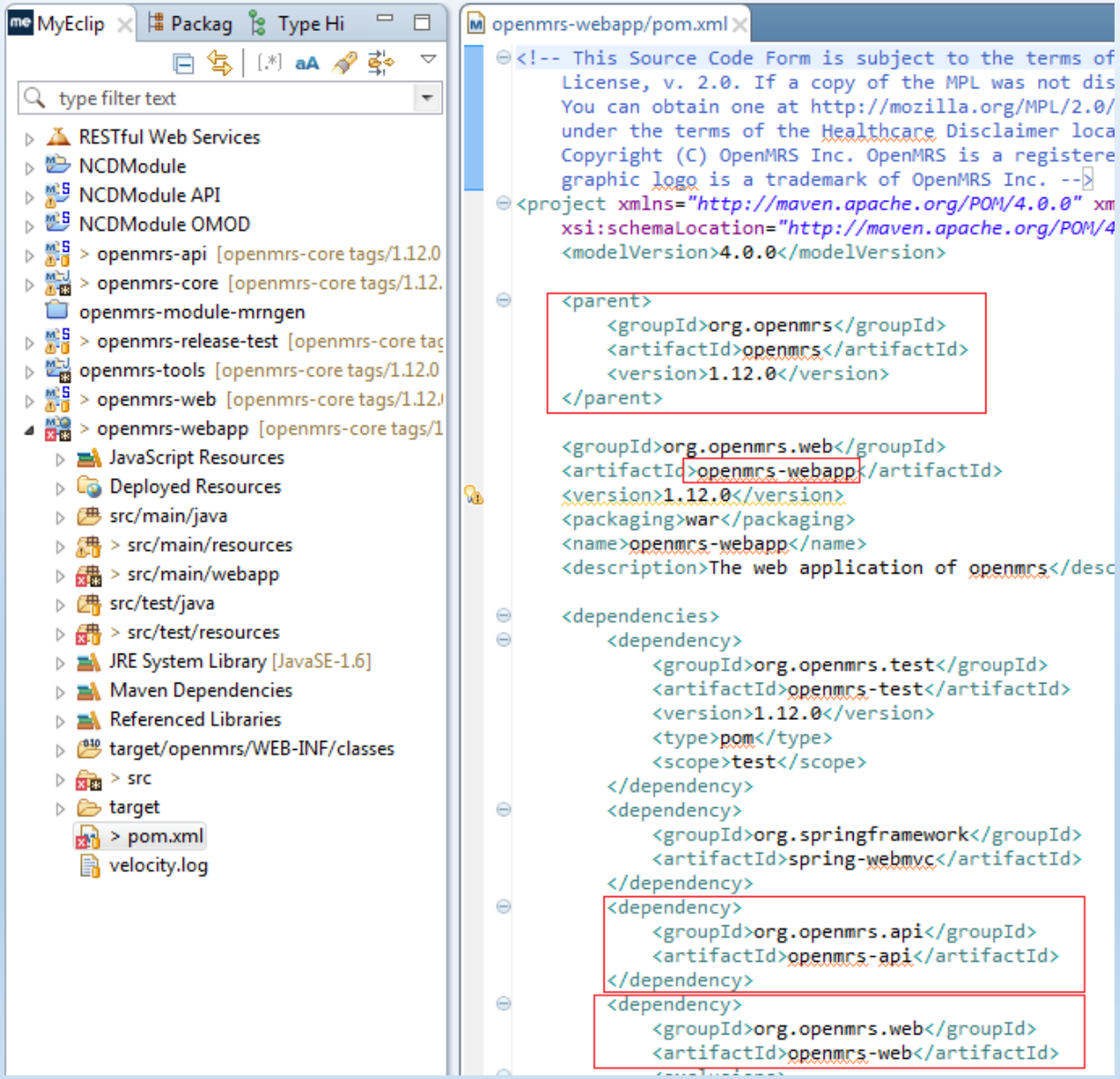
One of the many reasons for switching to OpenMRS 1.12.0 and for creating NCD 1.4 is to support the more modern **`hapi-structures`** *versions.*

```xml
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <javaCompilerVersion>1.6</javaCompilerVersion>
    <hapi.version>2.1</hapi.version>
    <maven.build.timestamp.format>yyyy-MM-dd HH:mm</maven.build.timestamp.format>
    <TIMESTAMP>${maven.build.timestamp}</TIMESTAMP>

    <openmrs.version.long>${parsedVersion.majorVersion}.${parsedVersion.minorVersion}.${parse
        ${parsedVersion.qualifier} Build ${revisionNumber}</openmrs.version.long>
    <openmrs.version.short>${parsedVersion.majorVersion}.${parsedVersion.minorVersion}.${pars
    <openmrs.version.shortnumericonly>${parsedVersion.majorVersion}.${parsedVersion.minorVers
    <openmrs.version>${project.version}</openmrs.version>

    <springVersion>3.2.7.RELEASE</springVersion>
    <hibernateVersion>3.6.5.Final</hibernateVersion>
    <customArgLineForTesting />
    <sonar.host.url>https://ci.openmrs.org/sonar</sonar.host.url>
    <sonar.issuesReport.html.enable>true</sonar.issuesReport.html.enable>
    <sonar.issuesReport.console.enable>true</sonar.issuesReport.console.enable>
    <sonar.analysis.mode>incremental</sonar.analysis.mode>

    <argLine>-Duser.language=en -Duser.region=US -Xmx512m
        -XX:MaxPermSize=512m ${customArgLineForTesting}</argLine>
</properties>
```

- The OpenMRS child modules have their own **pom.xml**, which indicate they are children of a 1.12.0 parent.

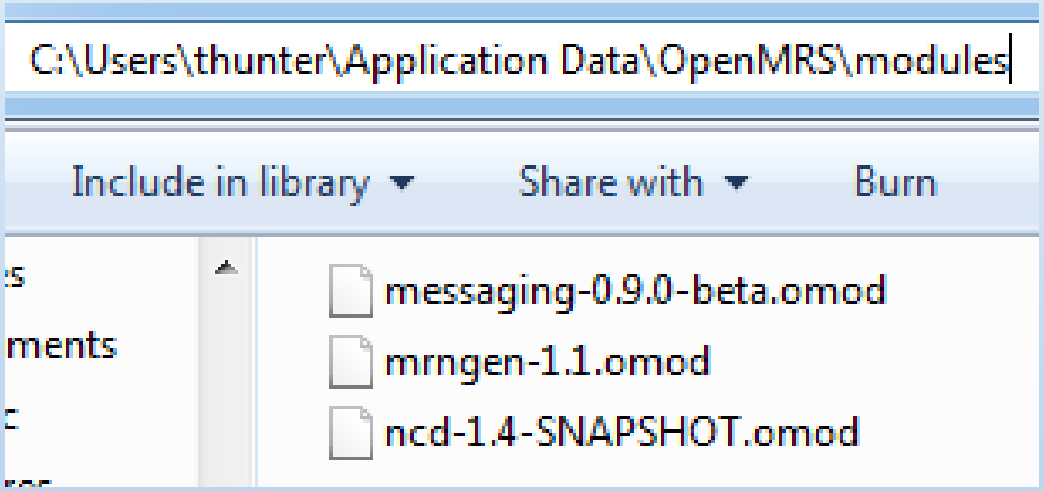


- The **openmrs-webapp** produces the WAR.
- As you can see, the webapp merely has Maven dependencies on the other projects.
- Note: **openmrs-web** is an entirely separate project from **openmrs-webapp**.



Concept of OpenMRS Module

- OpenMRS uses a variety of **.omod** files (think “plugins”) to do its job.
- In the Windows environment, **.omod** files are located here:



- In the Unix environment, **.omod** files are located here:

```
[root@ihie-ncd-test modules]# ll
total 3948
-rw-r--r-- 1 tomcat tomcat  531216 Sep 15 08:57 messaging-0.9.0-beta.omod
-rw-r--r-- 1 tomcat tomcat   87126 Sep 15 08:57 mrngen-1.1.omod
-rw-r--r-- 1 tomcat tomcat 3405737 Oct  5 15:45 ncd-1.4-SNAPSHOT.omod
[root@ihie-ncd-test modules]# pwd
/root/.OpenMRS/modules
[root@ihie-ncd-test modules]#
```

- In the Unix environment, **.omod** files are located here:

```
[root@ihie-ncd-test modules]# ll
total 3948
-rw-r--r-- 1 tomcat tomcat  531216 Sep 15 08:57 messaging-0.9.0-beta.omod
-rw-r--r-- 1 tomcat tomcat   87126 Sep 15 08:57 mrngen-1.1.omod
-rw-r--r-- 1 tomcat tomcat 3405737 Oct  5 15:45 ncd-1.4-SNAPSHOT.omod
[root@ihie-ncd-test modules]# pwd
/root/.OpenMRS/modules
[root@ihie-ncd-test modules]#
```

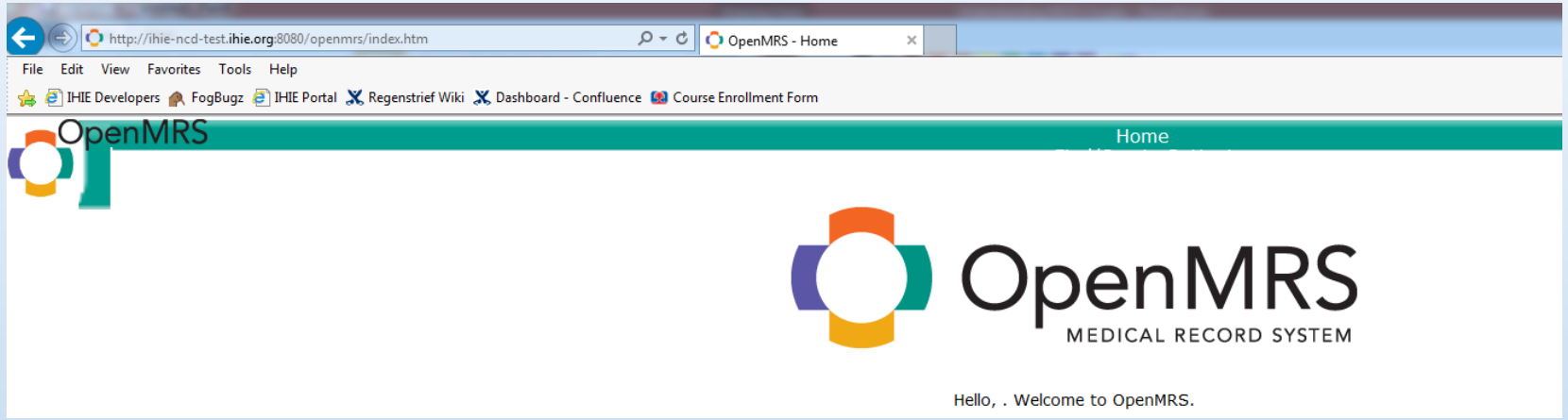
- The three OpenMRS modules you see here are all required to get NCD 1.4 running.
- Despite the presence of "-beta", for example, this is the latest stable version of that module.

- One benefit of the OpenMRS module architecture is the ability to drop in or replace a module at runtime, without restarting OpenMRS.
- You can drop the updated **`.omod`** file in the directory,

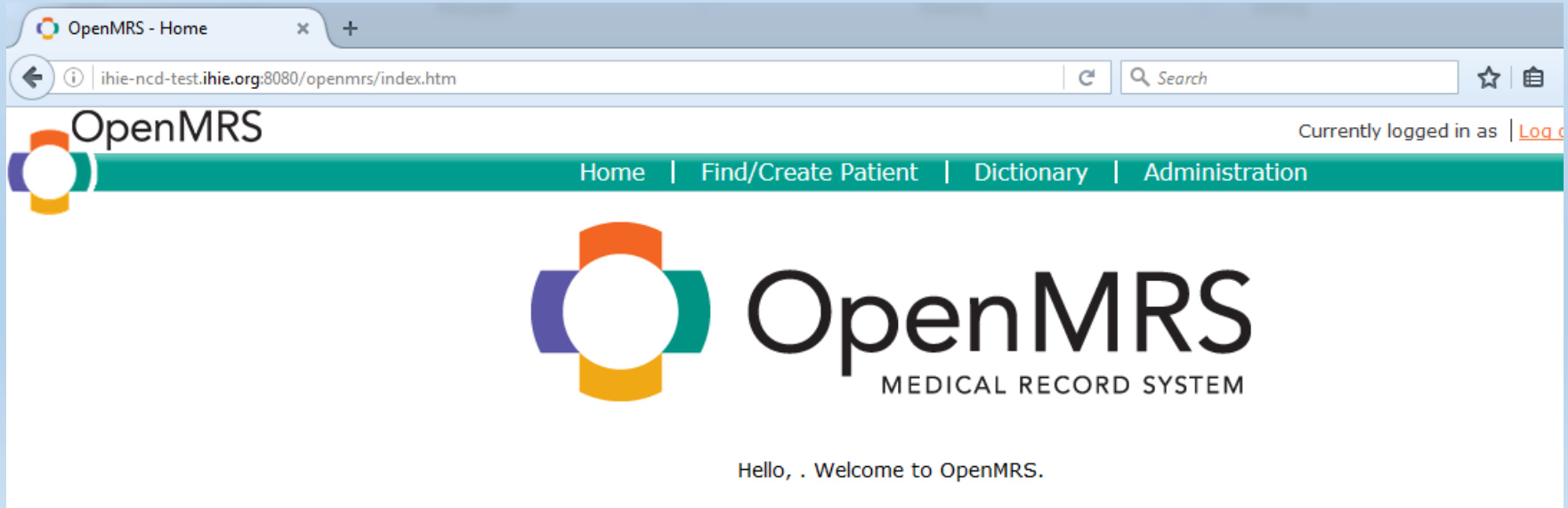
[`C:\Users\thunter\Application Data\OpenMRS\modules` or `/root/.OpenMRS\modules`]

or use the OpenMRS UI feature designed for that purpose.

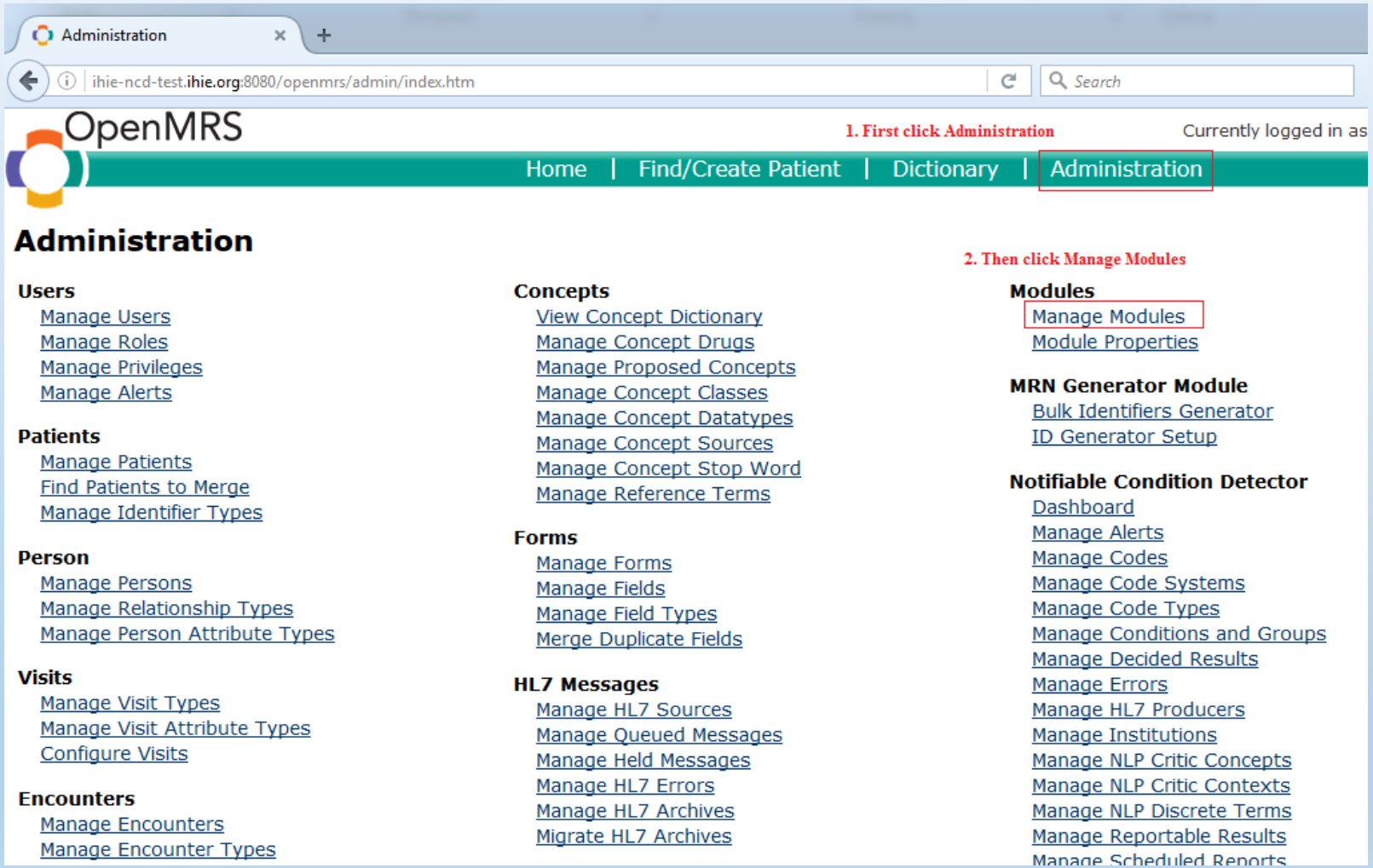
- OpenMRS 1.12.0 does not work correctly on IE11.



- Use Chrome or Firefox



- This is the OpenMRS UI feature for updating a module:



OpenMRS – Module – ExampleModule

openmrs-module-basicmodule

- **pom.xml**
- **BasicModuleActivator.java**

api

- **pom.xml**

```java
/**
 * This class contains the logic that is run every time
 * this module is either started or shutdown
 */
public class BasicModuleActivator implements Activator
{
    private Log log = LogFactory.getLog(this.getClass());

    public void startup()
    {
        log.info("Starting Basic Module");
    }

    public void shutdown()
    {
        log.info("Shutting down Basic Module");
    }

}
```

omod

- **AdminList.java**

```java
/**
 * This class defines the links that will appear on the administration page under the
 * "basicmodule.title" heading. This extension is enabled by defining (uncommenting) it in the
 * /metadata/config.xml file.
 */
public class AdminList extends AdministrationSectionExt
{
    public Extension.MEDIA_TYPE getMediaType()
    {
        return Extension.MEDIA_TYPE.html;
    }

    public String getTitle()
    {
        return "basicmodule.title";
    }

    public Map<String, String> getLinks()
    {
        Map<String, String> map = new HashMap<String, String>();
        map.put("module/basicmodule/basicmoduleLink.form", "basicmodule.replace.this.link.name");

        return map;
    }

}
```

- **BasicModuleFormController.java**

```java
/**
 * This class configured as controller using annotation and mapped with the URL of
 * 'module/basicmodule/basicmoduleLink.form'.
 */
@Controller
@RequestMapping(value = "module/basicmodule/basicmoduleLink.form")
public class BasicModuleFormController
{
    protected final Log log = LogFactory.getLog(getClass());

    /** Success form view name */
    private final String SUCCESS_FORM_VIEW = "/module/basicmodule/basicmoduleForm";

    /**
     * Initially called after the formBackingObject method to get the landing form name
     * @return String form view name */
    @RequestMapping(method = RequestMethod.GET)
    public String showForm()
    {
        return SUCCESS_FORM_VIEW;
    }

    /**
     * All the parameters are optional based on the necessity
     * @param httpSession
     * @param anyRequestObject
     * @param errors
     * @return  */
    @RequestMapping(method = RequestMethod.POST)
    public String onSubmit(HttpSession httpSession, @ModelAttribute("anyRequestObject") Object anyRequestObject, BindingResult errors)
    {
        if (errors.hasErrors())
        {
            // return error view
        }

        return SUCCESS_FORM_VIEW;
    }

    /**
     * This class returns the form backing object. This can be a string, a boolean, or a normal java
     * pojo. The bean name defined in the ModelAttribute annotation and the type can be just defined
     * by the return type of this method */
    @ModelAttribute("thePatientList")
    protected Collection<Patient> formBackingObject(HttpServletRequest request) throws Exception
    {
        // get all patients that have an identifier "101" (from the demo sample data)
        // see http://resources.openmrs.org/doc/index.html?org/openmrs/api/PatientService.html for
        // a list of all PatientService methods
        Collection<Patient> patients = Context.getPatientService().findPatients("101", false);

        // this object will be made available to the jsp page under the variable name
        // that is defined in the @ModuleAttribute tag
        return patients;
    }
}
```

- **src/main/resources/webModuleApplicationContext.xml**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:jee="http://www.springframework.org/schema/jee"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:util="http://www.springframework.org/schema/util"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context-3.0.xsd
           http://www.springframework.org/schema/jee
           http://www.springframework.org/schema/jee/spring-jee-3.0.xsd
           http://www.springframework.org/schema/tx
           http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
           http://www.springframework.org/schema/util
           http://www.springframework.org/schema/util/spring-util-3.0.xsd">

    <!-- Add here beans related to the web context -->

    <!-- Annotation based controllers -->
    <bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"/>
    <context:component-scan base-package="org.openmrs.module.basicmodule.web.controller" />

</beans>
```

- **src/main/resources/config.xml**

```xml
<module configVersion="1.2">

    <!-- Base Module Properties -->
    <id>${project.parent.artifactId}</id>
    <name>${project.parent.name}</name>
    <version>${project.parent.version}</version>
    <package>${MODULE_PACKAGE}</package>
    <author>Ben Wolfe</author>
    <description>
            ${project.parent.description}
    </description>

    <activator>${MODULE_PACKAGE}.BasicModuleActivator</activator>

    <!-- updateURL>https://modules.openmrs.org/modules/download/@MODULE_ID@/update.rdf</updateURL> -->
    <!-- /Base Module Properties -->

    <require_version>${openmrs.platform.version}</require_version>

    <!-- Extensions -->
    <extension>
        <point>org.openmrs.admin.list</point>
        <class>${MODULE_PACKAGE}.extension.html.AdminList</class>
    </extension>
```

- **src/main/webapp/basicmoduleForm.jsp**

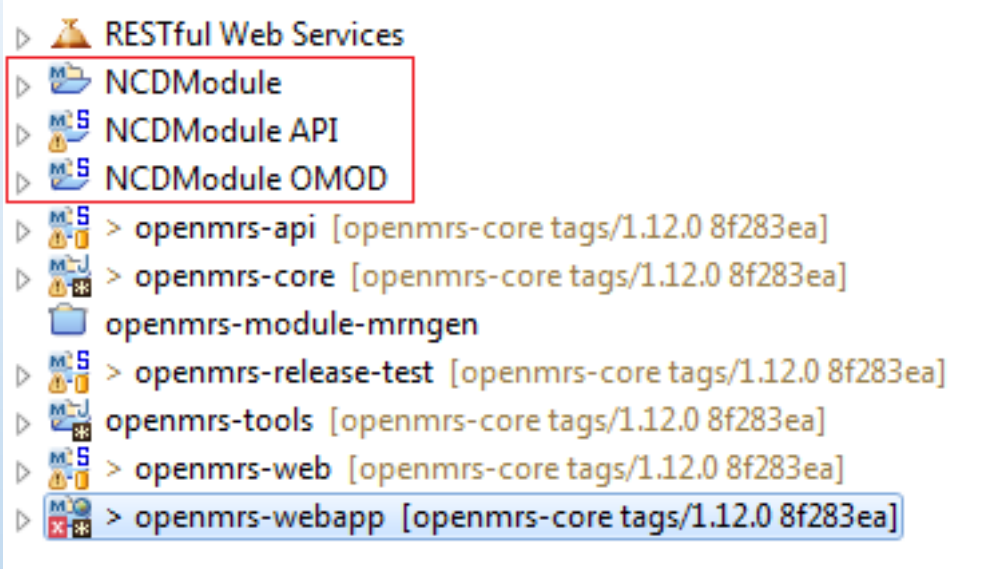
```jsp
<%@ include file="/WEB-INF/template/include.jsp"%>
<%@ include file="/WEB-INF/template/header.jsp"%>
<h2><spring:message code="basicmodule.replace.this.link.name" /></h2>

<br/>
<table>
  <tr>
    <th>Patient Id</th>
    <th>Name</th>
    <th>Identifier</th>
  </tr>
  <c:forEach var="patient" items="${thePatientList}">
      <tr>
          <td>${patient.patientId}</td>
          <td>${patient.personName}</td>
          <td>${patient.patientIdentifier}</td>
      </tr>
  </c:forEach>
</table>

<%@ include file="/WEB-INF/template/footer.jsp"%>
```

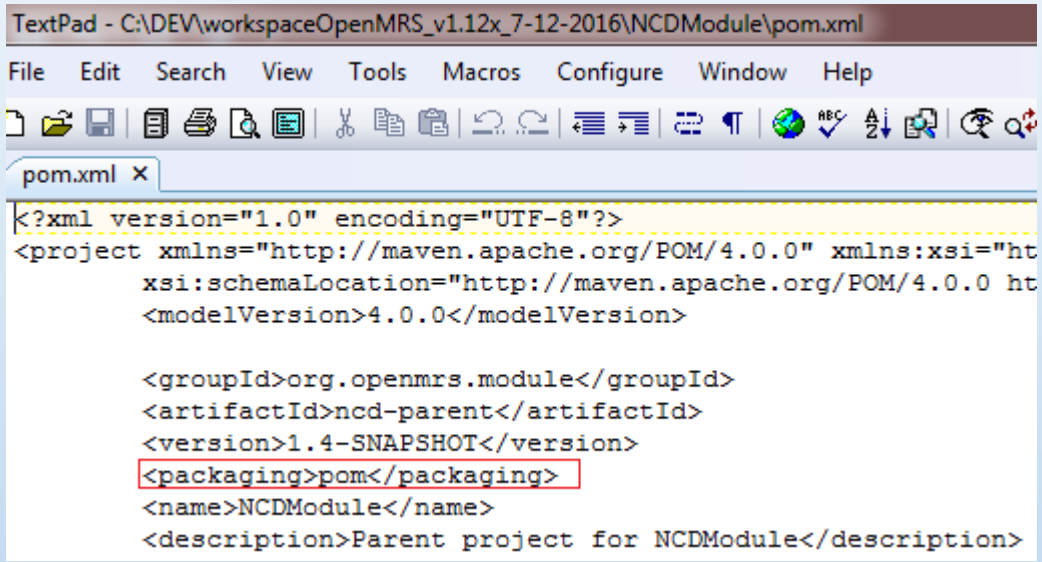
- **pom.xml**

Original: **J:\SolutionEngineering\DS&A NCD\NCD 1.4 - OpenMRS 1.12.0\Convert NCD to Maven Notes\NCD-1.4.x_Project_Notes.vsd**

Module layout: API, OMOD

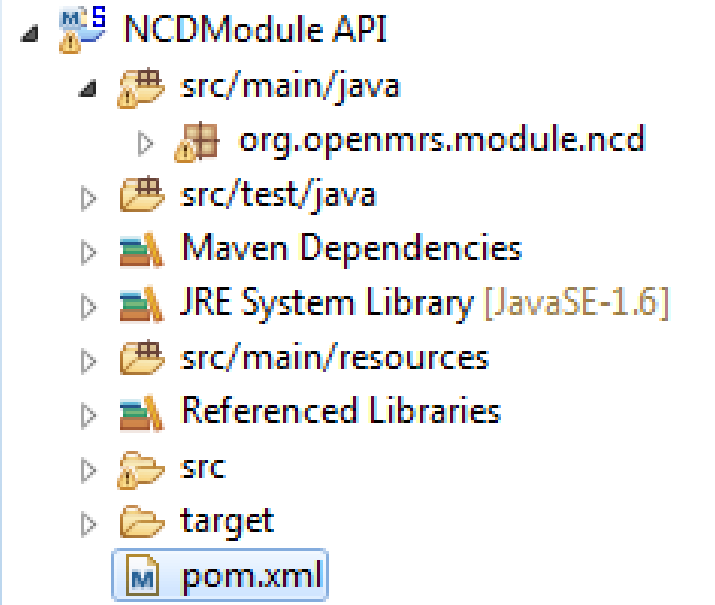
- This is the way NCD looks in Eclipse:

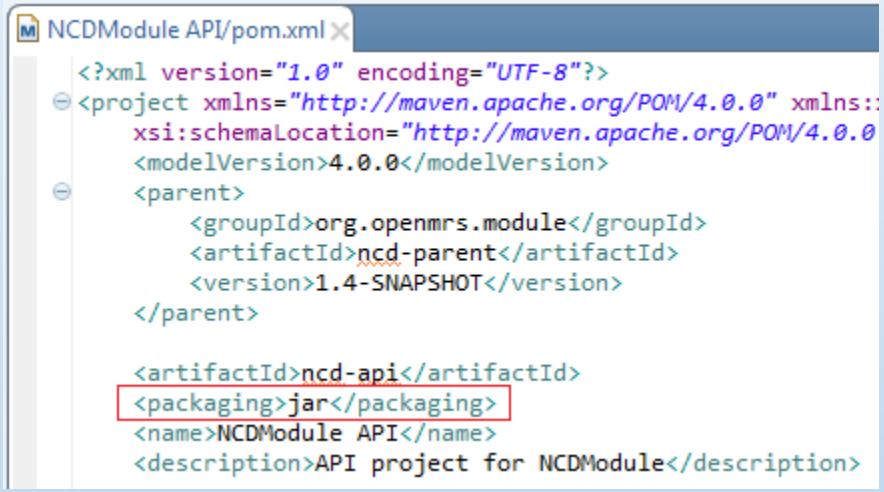


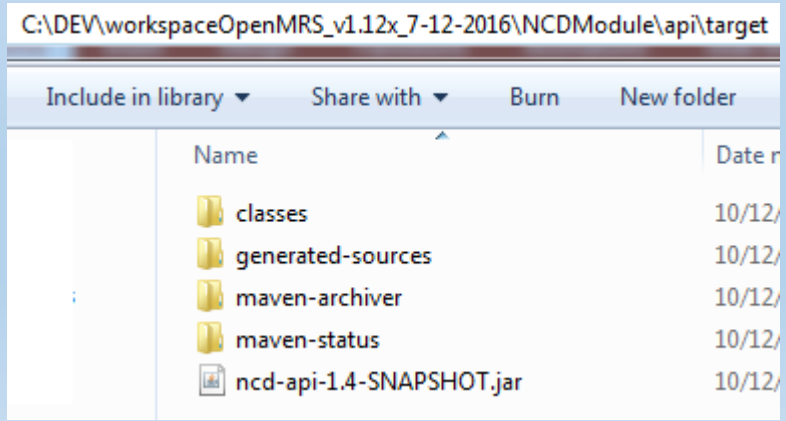


- NCDModule is the parent.

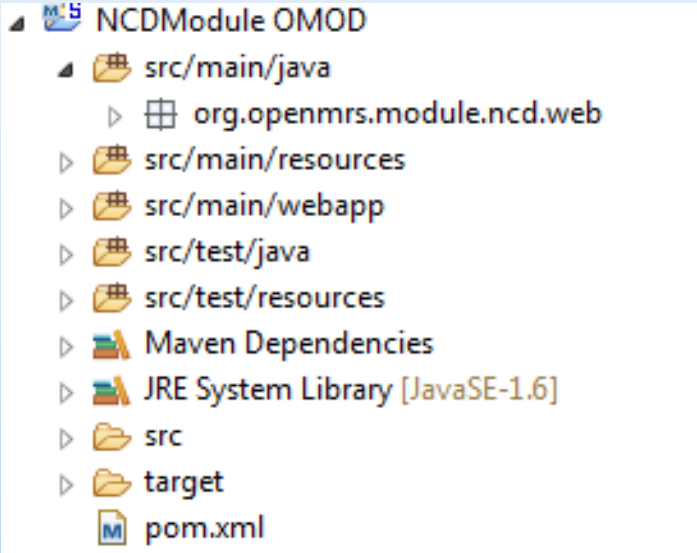
- The NCDModule API is a child project and it produces a JAR.

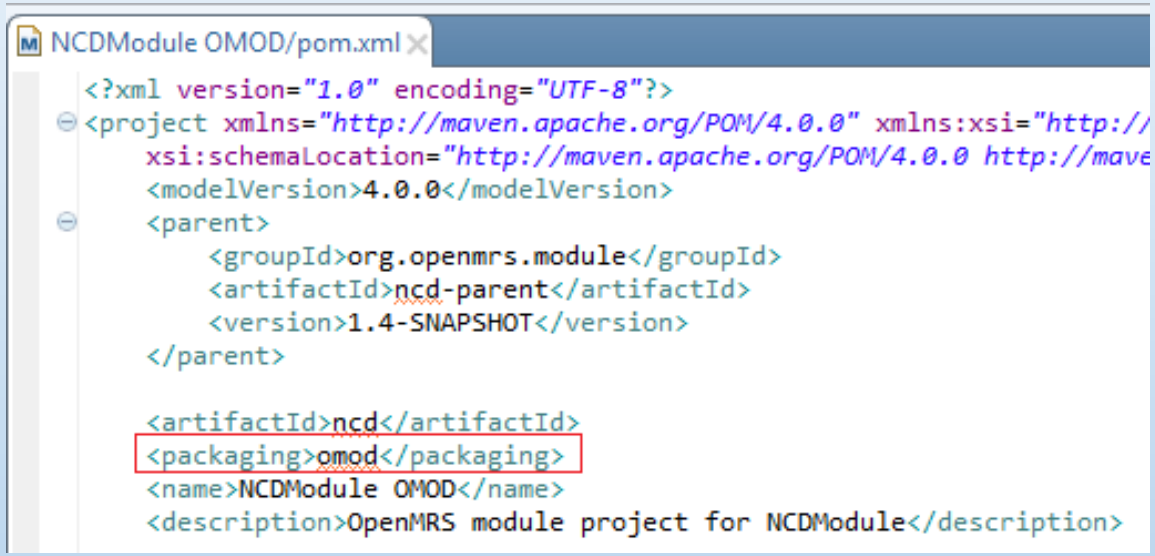


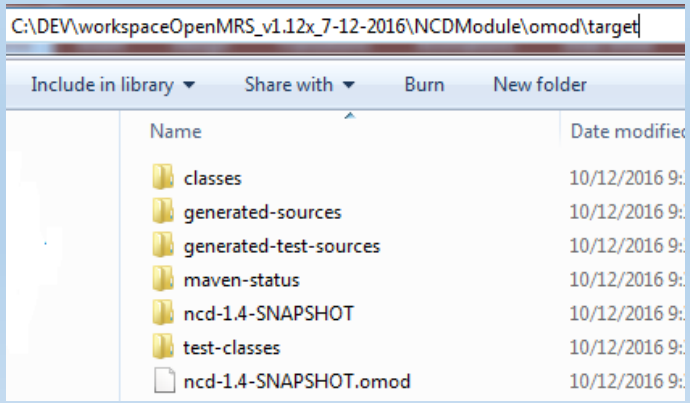




- The NCDModule OMOD is a child project and it produces an **.omod**.



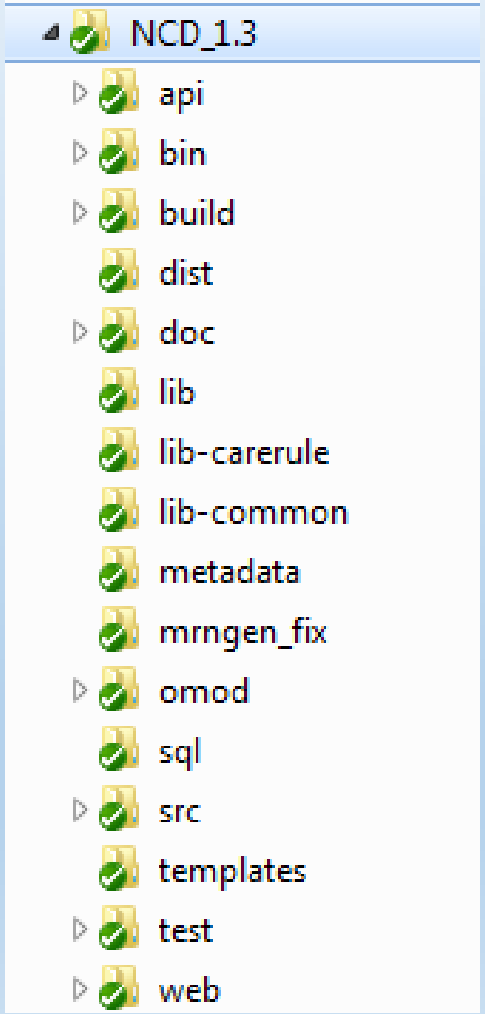


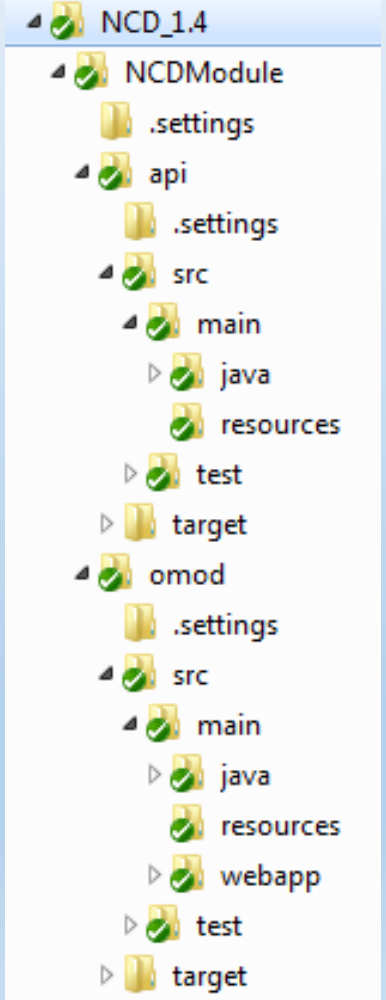


- When I migrated from NCD 1.3 to NCD 1.4, that involved leaving ANT and embracing Maven.

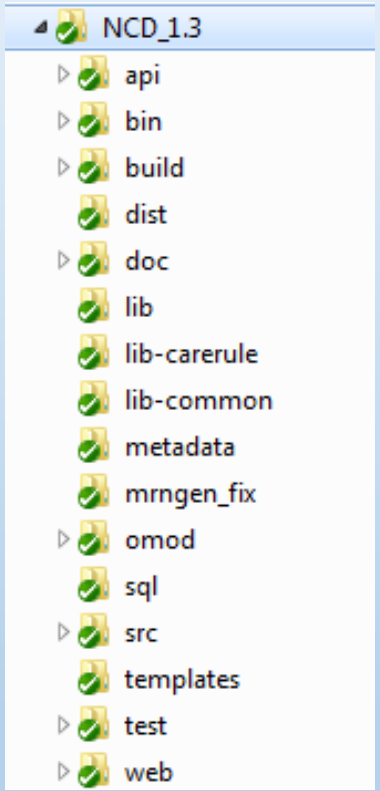
- Further, after I analyzed how an OpenMRS module was supposed to work, I moved the majority of the code to NCDModule API.

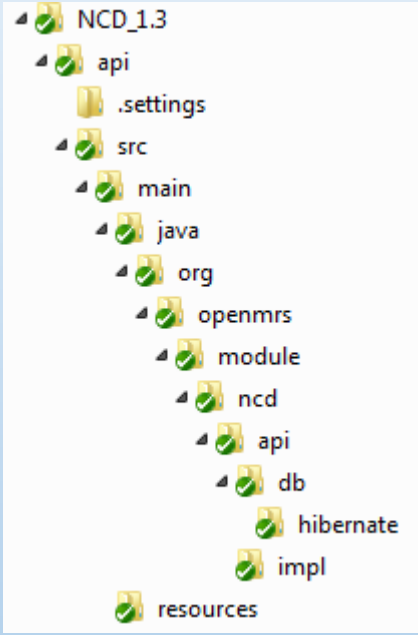
- On the left is the project structure for NCD 1.3
- On the right is the equivalent that I created for NCD 1.4

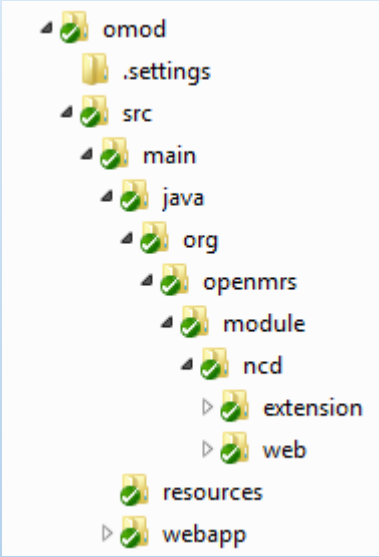


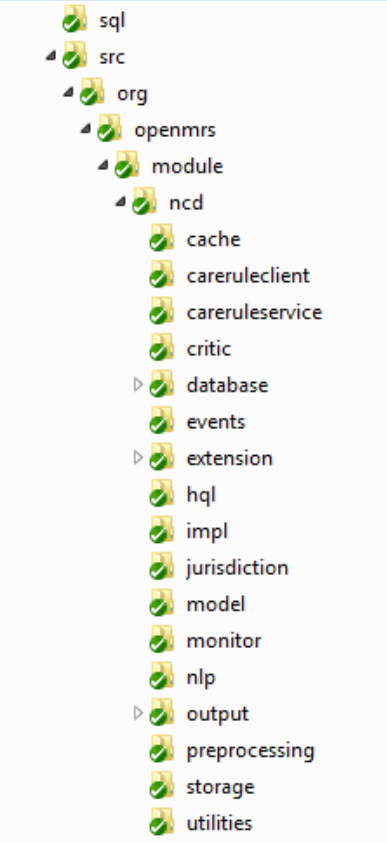


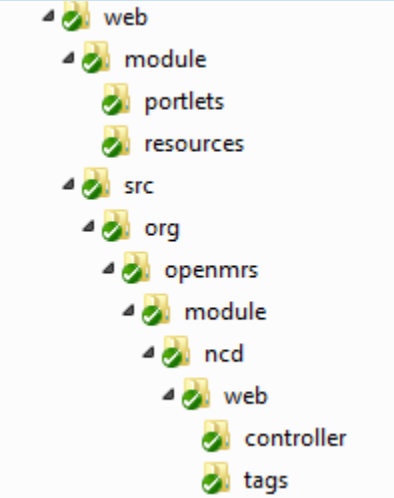
- NCD 1.3 has Java files scattered over several locations with no real rhyme or reason.





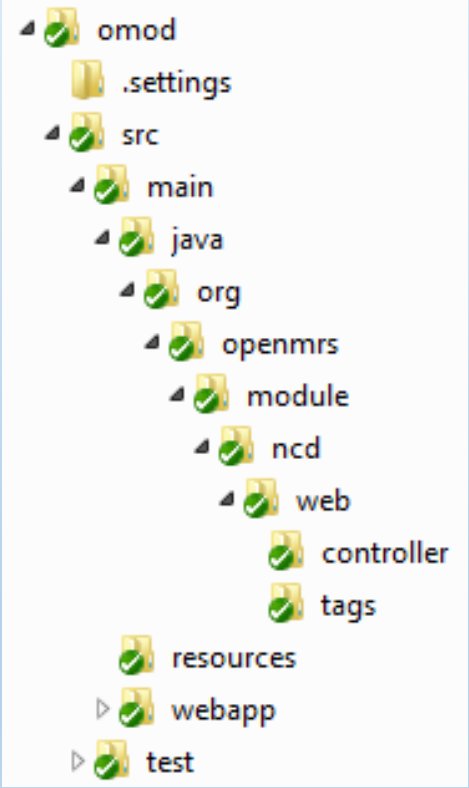


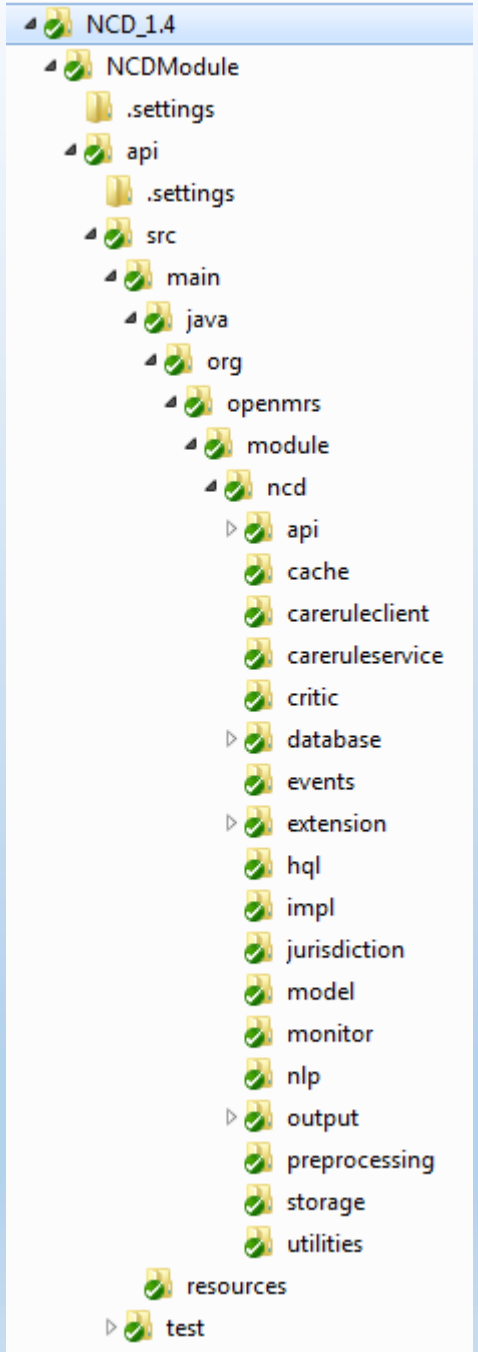




- For NCD 1.4 I made the decision to restructure the module, so that most of the Java code is in NCDModule API, and with only UI-related code in the NCDModule OMOD.
- NCD 1.4 builds in this order:

```
--- maven-install-plugin:2.5.2:install (default-install) @ ncd ---
Installing C:\DEV\workspaceOpenMRS_v1.12x_7-12-2016\NCDModule\omod\target\ncd-1.4-SNAPSHOT.omod
Installing C:\DEV\workspaceOpenMRS_v1.12x_7-12-2016\NCDModule\omod\pom.xml to C:\Users\thunter\.
------------------------------------------------------------------------
Reactor Summary:

NCDModule ........................................... SUCCESS [  3.338 s]
NCDModule API ....................................... SUCCESS [ 16.130 s]
NCDModule OMOD ...................................... SUCCESS [ 29.776 s]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 49.902 s
Finished at: 2016-10-12T09:32:42-04:00
Final Memory: 48M/747M
------------------------------------------------------------------------
```



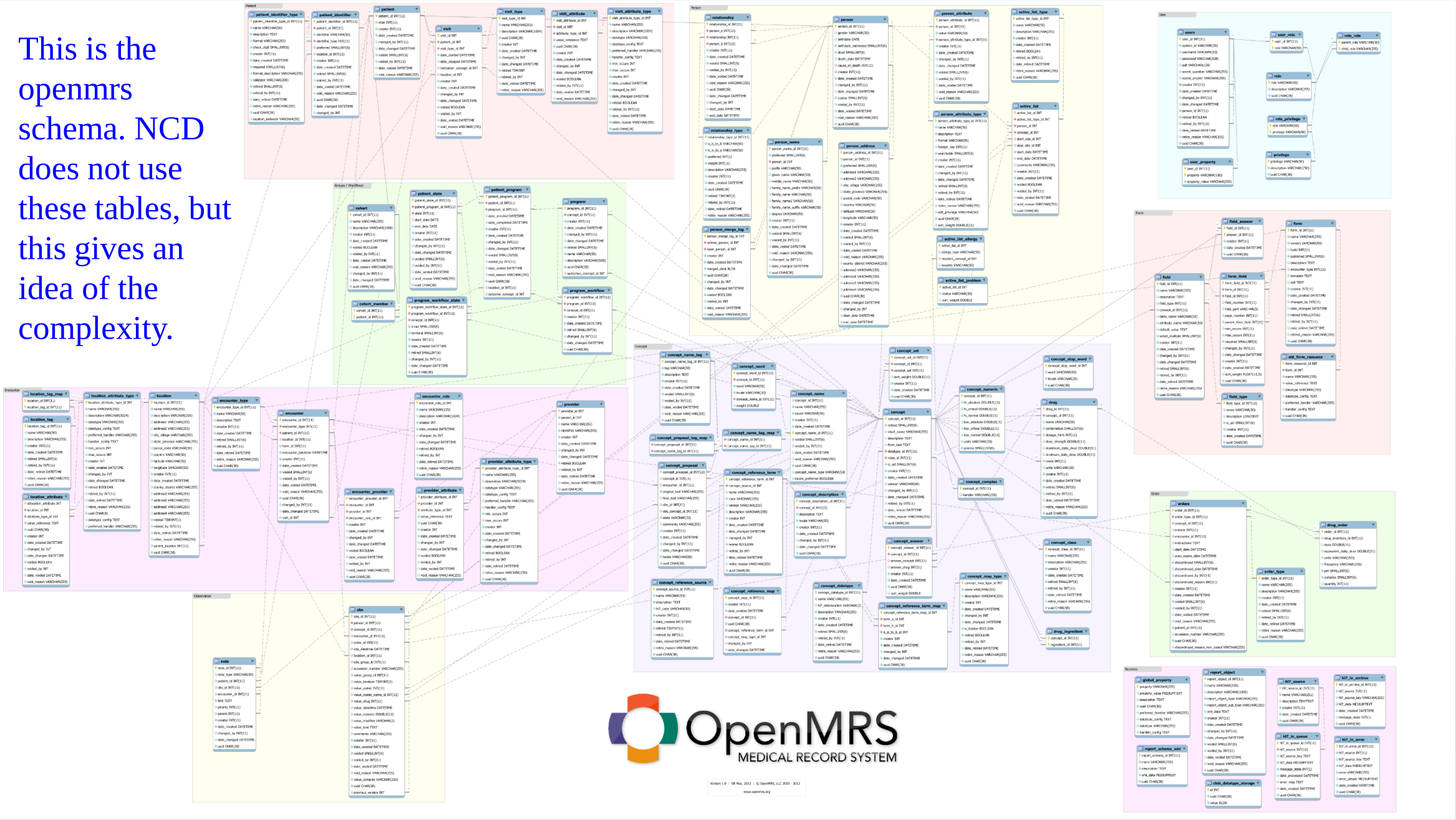


- API is just a JAR dependency in OMOD

OpenMRS Database

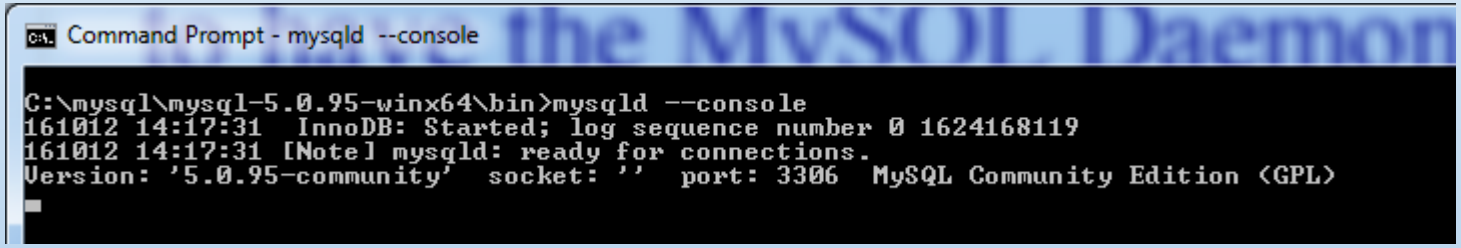
Understanding NCD 1.4

This is the openmrs schema. NCD does not use these tables, but this gives an idea of the complexity.



- OpenMRS uses as its database MySQL.
- Additionally, it uses a technology called Liquibase—one that facilitates automated changes to the database.
- Normally you can ignore Liquibase.

- If you have MySQL installed locally on your PC, have created an empty **`openmrs`** *database*, an **`openmrs`** *user* and given the grants to that user, then anytime you want to run OpenMRS, you *must* have the MySQL Daemon running, as follows:



- The MySQL command **`mysqld`** runs the daemon.
- To run in interactive mode, you run this command: **`mysql -u root -p`**
- Better yet: download MySQL Workbench

Efficient Development of NCD 1.4

- To develop NCD 1.4 efficiently, I have found the following setup works best:

1. Build OpenMRS using the Maven command to the right.

```
C:\DEV\workspaceOpenMRS_v1.12x_7-12-2016\openmrs-core>mvn clean install -Dmaven.test.skip=true -Dlicense.skip=true
```

Notice this script has 'maven.test.skip=true'. That skips both the *compiling* and running of unit tests.

2. After a successful build, deploy the war above here.

3. Build NCD 1.4 using the Maven command on the right.

```
C:\DEV\workspaceOpenMRS_v1.12x_7-12-2016\NCDModule>mvn clean install -DskipTests=true
```

4. Move the just-built `ncd-*.omod` to this directory, where it will be picked up automatically by OpenMRS.

5. This window represents the MySQL daemon process and it must remain running anytime you use OpenMRS.

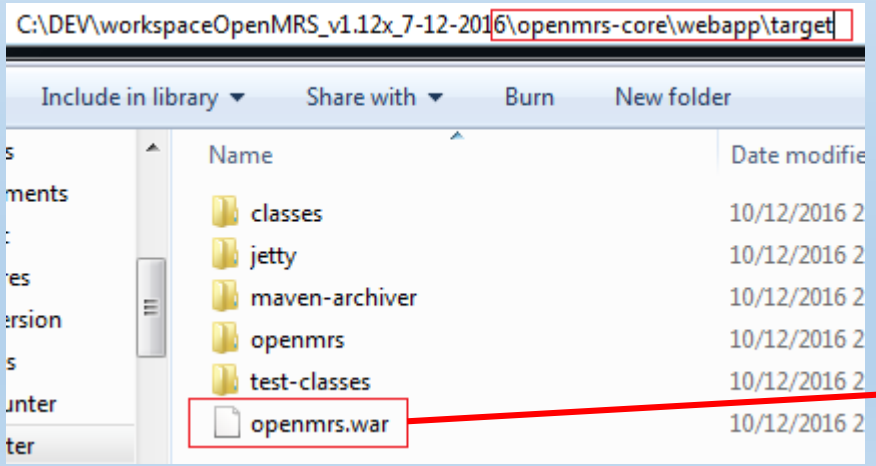
```
C:\mysql\mysql-5.0.95-winx64\bin>mysqld --console
160829 13:15:25  InnoDB: Started; log sequence number 0 723959821
160829 13:15:25 [Note] mysqld: ready for connections.
Version: '5.0.95-community'  socket: ''  port: 3306  MySQL Community Edition (GPL)
```

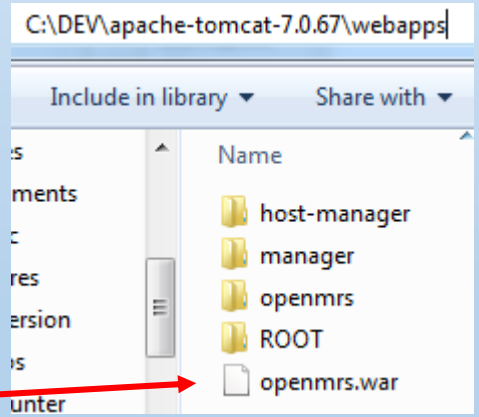
6. This MySQL database script should be run after every test, if you want to avoid repeat-patient errors and such. The text version of this script can be found here:
J:\SolutionEngineering\DS&A NCD\TSH_Notes_2016\OpenMRS_Delete_Users_to_ClearTestDB.txt

```
delete from openmrs.person_name where person_name_id > 0;
delete from openmrs.person_attribute where person_attribute_id >
delete from openmrs.person_address where person_address_id > 0;
delete FROM openmrs.encounter_provider where encounter_provider_i
delete from openmrs.provider where provider_id > 0;
update openmrs.obs set obs_group_id = null where obs_id > 0;
delete from openmrs.obs where obs_id > 0;
delete FROM openmrs.encounter where encounter_id > 0;
delete from openmrs.patient where patient_id > 38;
delete from openmrs.person where person_id > 38;
```

Better yet, run the **`mvnCleanInstall.bat`** file saved here with the project—because it handles our security better.

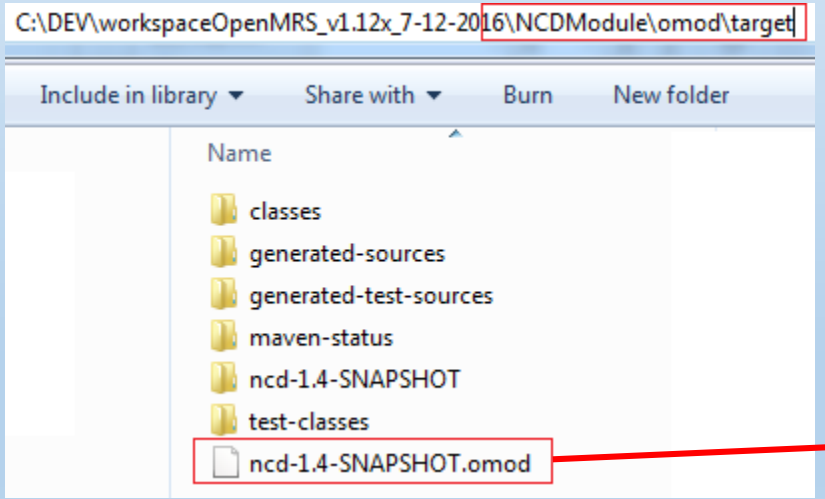
- Here's how I arrived at this approach:
- OpenMRS does not deploy well from within Eclipse because of the weird `.omod` modules.
- So, I build OpenMRS from the command line and deploy the WAR manually into Tomcat. That still permits debugging.



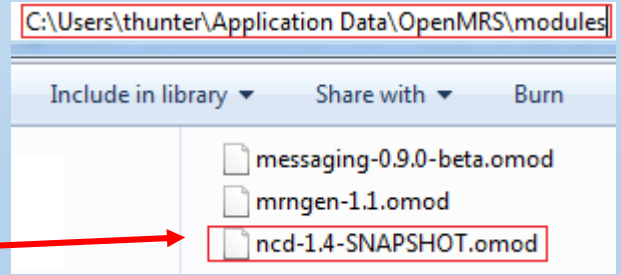



```
mvn clean install -Dmaven.test.skip=true -Dlicense.skip=true -Djavax.net.ssl.trustStore=C:/DEV/JDK6_45/jre/lib/security/trust.jks -
```

- Next, you—because of the `.omod` file—face the same problem with NCD 1.4.
- So, I build NCD with Maven on the command line and then manually deploy it to the OpenMRS modules directory.

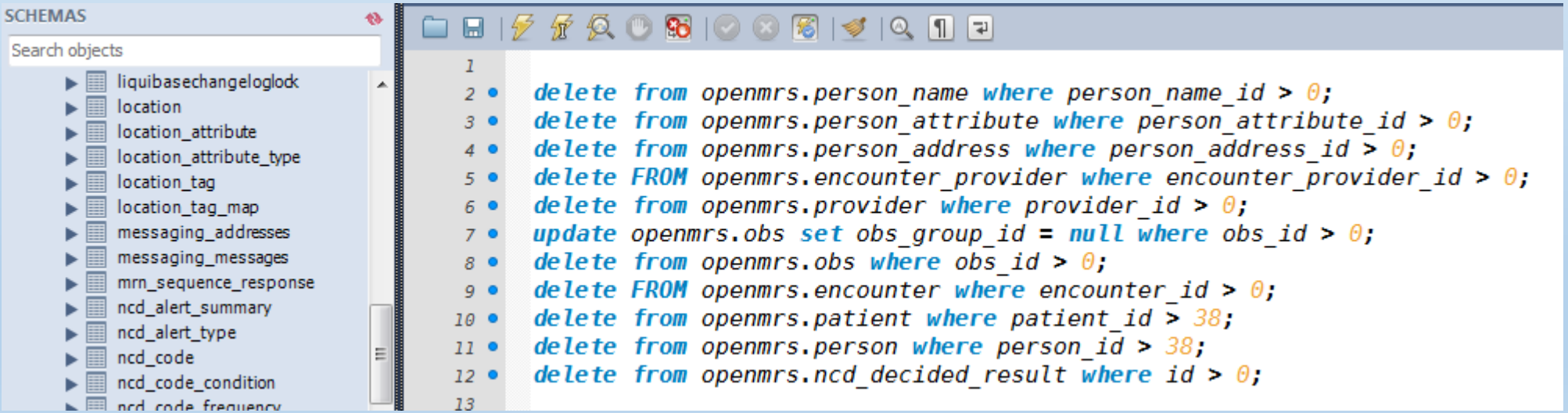


Every time you change NCD 1.4, you need to deploy it like this to have it pick up your changes.




```
mvn -e clean install -DskipTests=true -Djavax.net.ssl.trustStore=C:/DEV/JDK6_45/jre/lib/security/trust.jks -Djavax.net.ssl.trustStorePassword=changeit
```

- If you are working on a particular NCD 1.4 feature—and want to be able to hit that same patch of code over and over—then you have one more step. You need to clean out a few tables in the MySQL DB or else NCD will not give you idempotent behavior.



Soft Copy: J:\SolutionEngineering\DS&A NCD\NCD 1.4 - OpenMRS 1.12.0\OpenMRS_Delete_Users_to_ClearTestDB_Script.txt

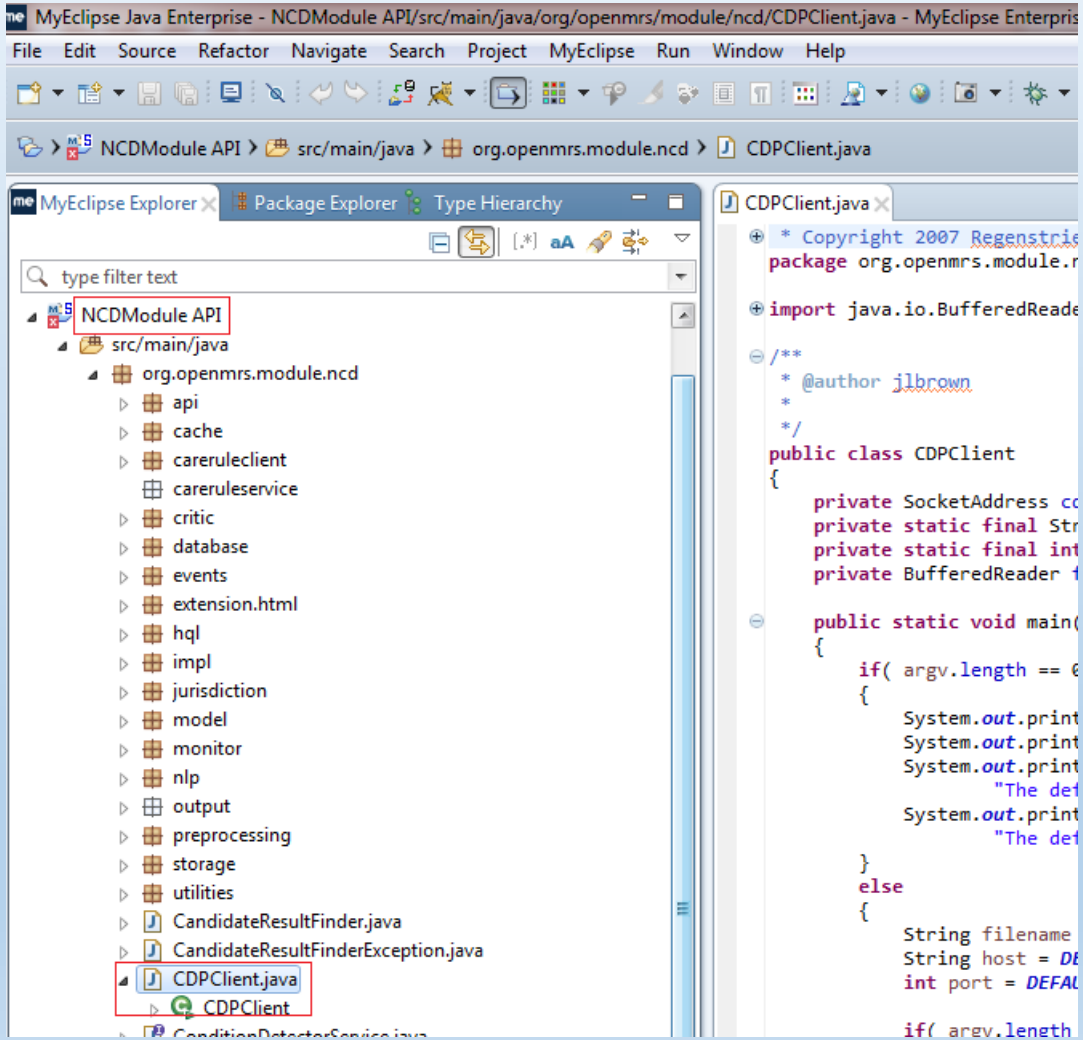
- “I want to work on NCD. What do I do?”
  - Open these windows:
    - `C:\DEV\workspaceOpenMRS_v1.12x_7-12-2016\openmrs-core\webapp`
    - `C:\DEV\apache-tomcat-7.0.67\webapps`

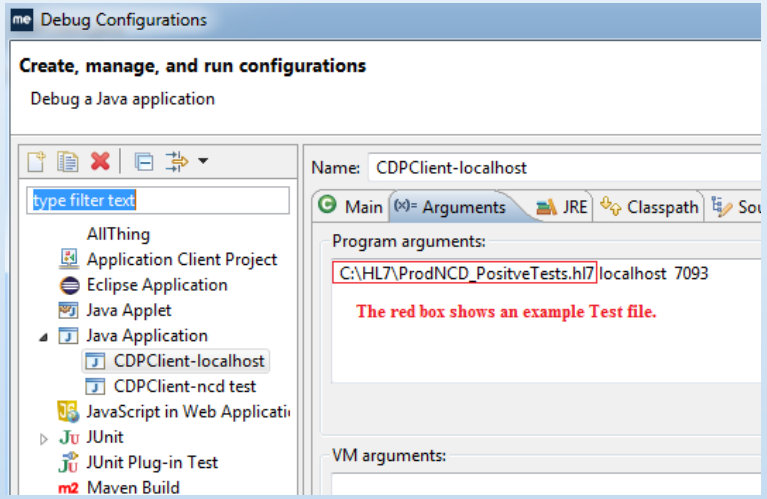
    - `C:\DEV\workspaceOpenMRS_v1.12x_7-12-2016\NCDModule\omod`
    - `C:\Users\thunter\Application Data\OpenMRS\modules`
  - Open three command prompts:
    - `C:\mysql\mysql-5.0.95-winx64\bin> mysqld -console` ***(leave open and running)***

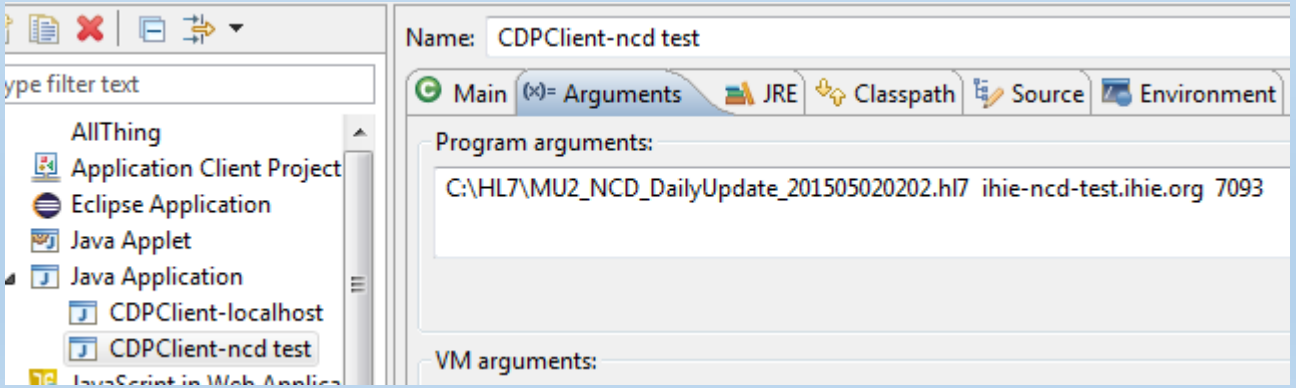
    - `C:\DEV\workspaceOpenMRS_v1.12x_7-12-2016\openmrs-core>mvnCleanInstall.bat`
    - `C:\DEV\workspaceOpenMRS_v1.12x_7-12-2016\NCDModule>mvnCleanInstall.bat`
  - Open MySQL Workbench
    - Connect to your local MySQL instance

How To Test NCD 1.4

- Testing NCD 1.4 consists of writing HL7s over port 7093.







Use of Terser

- One of the challenges you face when using NCD is finding creative ways to get the data you need such as the Provider or Person contained within an HL7.
- NCD 1.3 relies on XPath to get that sort of data.

- The XPath-based approach led to NCD 1.3 code like below:

XmlUtilities.java

```
private static Element findElement(String xpath, Element elem)
{
    if (elem == null)
    {
        return null;
    }

    xpath                       = xpath.replace("//", "**");
    String[] xpathElements      = xpath.split("/");
    int lastXPathElementIndex   = xpathElements.length - 1;

    if (xpathElements[lastXPathElementIndex].equals("text()"))
    {
        lastXPathElementIndex --;
    }

    Element curElem = elem;

    for( int curElementIndex = 0; curElementIndex <= lastXPathElementIndex; curElementIndex++ )
    {
        String curXPathElement = xpathElements[curElementIndex];

        if( curXPathElement.equals(".") )
        {
            // do nothing since this means the current node
            continue;
        }
        else if( curXPathElement.equals("..") )
        {
            // get the parent node
            curElem = (Element)curElem.getParentNode();
        }
        else if ( curXPathElement.startsWith("**"))
        {
            curXPathElement = curXPathElement.replace("**", "");
            curElem = (Element)curElem.getOwnerDocument().getElementsByTagName(curXPathElement).item(0);
        }
        else if ( curXPathElement.startsWith("*"))
        {
            curXPathElement = curXPathElement.replace("*", "");
            curElem = (Element)curElem.getElementsByTagName(curXPathElement).item(0);
        }
        else
        {
            curElem = (Element)curElem.getElementsByTagName(curXPathElement).item(0);
        }
```

- When I adapted NCD 1.3 into NCD 1.4, I generally allowed the existing approach to remain—but backstopped by a modern approach—using:

ca.uhn.hl7v2.util.Terser

- A Terser parses the HL7 message and allows clean access:

ResultStorageHelper.java

```
private static void performHeroicsToGetTheProviderFromTheOriginalMessage( String messageText, ProviderInfo providerInfo )
{
    Message parsedMessage   = null;

    try
    {
        PipeParser parser               = new PipeParser();
        parsedMessage                   = parser.parse( messageText );
        Terser parsedTerser             = new Terser( parsedMessage );

        String attendingPhysicianLast   = parsedTerser.get( "/.PV1-7-2" );
        String attendingPhysicianFirst  = parsedTerser.get( "/.PV1-7-3" );
        String attendingPhysicianMiddle = parsedTerser.get( "/.PV1-7-4" );
        String attendingPhysicianSuffix = null;
```

- So, in this example, you see many XPaths tried and then my backstop of the **Terser**, after all else has failed.

ResultStorageHelper.java

```
        }
        else if (providerIdSource.equals("OBR.28.1"))
        {
            firstName  = obr.getResultCopyToFirstName();
            lastName   = obr.getResultCopyToLastName();
            middleName = obr.getResultCopyToMiddleName();
            suffixName = obr.getResultCopyToSuffixName();
            nameSource = "OBR.28.2/OBR.28.3/OBR.28.4/OBR.28.5";
        }
        else if (providerIdSource.equals("PV1.8.1"))
        {
            firstName  = provider.getReferringDoctorFirstName();
            lastName   = provider.getReferringDoctorLastName();
            middleName = provider.getReferringDoctorMiddleName();
            suffixName = provider.getReferringDoctorSuffixName();
            nameSource = "PV1.8.2/PV1.8.3/PV1.8.4/PV1.8.5";
        }
        else if (providerIdSource.equals("PV1.9.1"))
        {
            firstName  = provider.getConsultingDoctorFirstName();
            lastName   = provider.getConsultingDoctorLastName();
            middleName = provider.getConsultingDoctorMiddleName();
            suffixName = provider.getConsultingDoctorSuffixName();
            nameSource = "PV1.9.2/PV1.9.3/PV1.9.4/PV1.9.5";
        }
        else if (providerIdSource.equals("PV1.7.1"))
        {
            firstName  = provider.getAttendingDoctorFirstName();
            lastName   = provider.getAttendingDoctorLastName();
            middleName = provider.getAttendingDoctorMiddleName();
            suffixName = provider.getAttendingDoctorSuffixName();
            nameSource = "PV1.7.2/PV1.7.3/PV1.7.4/PV1.7.5";
        }
        else if (providerIdSource.equals("PV1.17.1"))
        {
            firstName  = provider.getAdmittingDoctorFirstName();
            lastName   = provider.getAdmittingDoctorLastName();
            middleName = provider.getAdmittingDoctorMiddleName();
            suffixName = provider.getAdmittingDoctorSuffixName();
            nameSource = "PV1.17.2/PV1.17.3/PV1.17.4/PV1.17.5";
        }

        if( firstName == null || firstName.trim().length() == 0 )
        {
            performHeroicsToGetTheProviderFromTheOriginalMessage( messageText, providerInfo );
        }
```

- Successful use of the **Terser** requires a message that is fairly compliant.
- Often, to use the **Terser**, it's necessary to clean up a few fields, such as phone numbers, that would otherwise cause the **Terser** to fail.
- Being able to use the **Terser** is worth the trouble as you can even find specific repeating segment values:

```
for( int i = 0; keepGoing; i++ )
{
    String obxValue = "";

    try
    {
        obxValue = parsedTerser.get( "/.OBX(" + i + ")-25" );
    }
    catch( Exception e )
    {
        // Not an error
    }

    if( obxValue == null || obxValue.trim().length() == 0 )
    {
        keepGoing = false;
    }
    else
    {
```