CTS2 Development Framework Implementation

Technical Overview

Scott Bauer, Cory Endle

HL7 Working Group Meeting
January 13th -18th, 2013

Pre-tutorial House Keeping

- This tutorial is not in a computer lab
- If you wish to follow along with the interactive demos you'll need a laptop and the tech stack and set up noted in the requirements document
- This will need to be done before the tutorial starts if you want to follow along
- Instructions are in the requirements document here: https://github.com/hsbauer/cts2-example-service/archive/master.zip

MAYO CLINIC

Who is this Tutorial for?

- Software developers, technical programmers, Information architects, health informaticists, technical analysts
- Should know how to implement the CTS2 technical framework specification at the end of this tutorial

MAYO CLINIC

What do I Need to Know as a Developer

- Introduction to CTS2 recommended
- Some knowledge of UML, CTS2 Specification, XML, JSON, REST required for full implementation
- Java and JavaScript will be helpful for some of the example
- Command line Maven, Python and Unix/Linux may be helpful as well

MAYO CLINIC

Overview

- CTS2 is an OMG standard for the query, interchange and update of terminological resources.
- This tutorial will provide technical implementers with guidance for the implementation of the standard.

MAYO CLINIC

Terminology Services

Terminologies:

- A list of unique terms and a set of properties helps define our common context for a term.
  - Like a dictionary.
- The way we know how to use terms is more complex:
  - Classifications
  - Synonyms
  - Unstated relationships

MAYO CLINIC

Terminology Services

- Knowing what is not in a dictionary is a way of knowing (This is sometimes called ontology).
- This is often built in to complex terminologies
- A terminology service provides the structure
  - For using this way of knowing about the service
  - For dealing with updates, versioning, history and other administrative functions

MAYO CLINIC

How to Approach the Specification

- The specification is designed for modular implementation. You do not need to implement the entire specification!
- Choose your implementation strategy based on local requirements, not to include a broad spectrum of the specification

MAYO CLINIC

Modular Specification
Implement / use what you need

- Code System Catalog Entry and Code System Version Catalog Entry
- Entity Description and Association
- Value Set Catalog, Value Set Definition and Resolved Value Set
- Map Catalog Entry, Map Version
- Concept Domain, Concept Domain Binding
- Statement

MAYO CLINIC

Modular Functionality
Implement / use what you need

- Read
- Query
- Import
- Export
- Update
- Maintenance
- History
- Temporal

MAYO CLINIC

Every Module Has:

- Formal Information Model in UML
  - Defines the resource, how it is identified and what information it contains
- Formal Computational Model in UML
  - Defines operations that are available for each functional (read, query, import, …) profile

MAYO CLINIC

Every Module Also Has

- XML Schema representation of the Information Model
- WADL rendering of Computational Model
  - Defines the URLs + operations (GET, PUT, POST, REMOVE) that implement the computations
- WSDL rendering of Computational Model

MAYO CLINIC

Combined they provide a set of rules that:

- Describe (some of) what can be said about various aspects of terminologies
- Provide a straight-forward format for structuring the information
- Provide rules for publishing, querying, exchanging and updating information in a distributed, federated environment

MAYO CLINIC

CTS2 in a nutshell

A set of XML Schemas for:

- Code Systems, Maps, Value Sets and Concept Domain Bindings

A set of URL formation rules for:

- Reading, Querying, Importing and Updating the XML Documents

A set of rules for representing XML in JSON, RDF, Java, Python, …

MAYO CLINIC

CTS2 Implementation
Planning an Implementation

MAYO CLINIC

How do I plan an implementation?

Ask the following questions:

- Do I need to implement a terminology service of my own?
- Do I need unique content?
- Do I need unique services from the content?
- Can I adequately map the content into the CTS2 Model or do I need expert help?

MAYO CLINIC

How do I plan an implementation?
CTS2 Service

A persisted Terminology

- CTS2 doesn't specify how this is done

- It may require wrapping an existing terminology service
  - Providing a mapping and a bridge for legacy technology

- Or creating a direct implementation of CTS2 over a database
  - Starting from scratch against a (set of) persisted terminologies.

MAYO CLINIC

How do I plan an implementation?
Are You Consuming Terminology Content?

If so what services do I need from the terminology service:

- Do I want to search for terminology entities?
- Do I need value sets for research forms?
- Do I need to map from one terminology to another?
- Do I need to traverse from a specific term to a more general one. Or establish term classification siblings? Semantic types?

MAYO CLINIC

Ways to Implement

- Service Implementers may want to implement a CTS2 Development Framework (DF) service plugin (This isn't required.)
- CTS2 consumers will want to implement a REST/JSON, REST/XML or SOAP client

MAYO CLINIC

CTS2 Plugin Tutorial Overview

MAYO CLINIC

Plugin Tutorial Overview
Where to Start

We'll choose a CTS2 module that can serve a fairly wide set of use cases:

CodeSystemCatalogEntry

- Create a service plugin based on this model element
- And use the same model element to access code system entities, their associated children, value sets, and versions of code systems.
- Note that this is a very small part of the API

MAYO CLINIC

CTS2 Development Framework
Providing a Toolset

- REST HTTP URL bindings
  - CTS2 model as Java Beans
  - Various Builder pattern query building objects
- CTS2 REST client
  - Manages JSON and XML parsing and Marshaling of either into CTS2 model objects
- Plugin admin structure
    - A quick start on your own service

MAYO CLINIC

CTS2 Implementation
Service Plugin Tutorial

MAYO CLINIC

CTS2 Plugins

How are Development Framework Plugins Used?

A plugin can be used to add a service to the CTS2 Development Framework base.

- Given a source terminology with it's own data model and corresponding database schema
- Map it into a set of CTS2 Model Objects and attach it to the framework where it can be served up in CTS2 REST

MAYO CLINIC

CTS2 plugins
How are Plugins Used?

How can a plugin be enabled within the CTS2 Development Framework?

- Every plugin must implement the ServiceProvider Interface*
- The jarred plugin will be uploaded to the stand alone CTS2 Development Framework server.

*edu.mayo.cts2.framework.service.provider.ServiceProvider

MAYO CLINIC

CTS2 Development Framework Service Plugin Example

Requirements

- Maven
- Eclipse with the Maven plugin
- Example Code
- Stand alone CTS2 Framework Server
- MySQL loaded with small Gene Ontology database

MAYO CLINIC

CTS2 Development Framework Service Plugin Example

Goals for this part of tutorial

- Connect to a service
- Integrate 3rd party software into the build
- Map source data to a CTS2 model element
- Get the standalone development framework service up and running
- Integrate the plugin into the service

MAYO CLINIC

Interactive Demo
Service Plugin

If you want to follow along, you should at this point have:

- Maven installed
- MySQL server started with the Gene Ontology loaded into a database named “godata”
- The example project pulled into Eclipse
- The code snippet folder available
- Initial clean and install of the maven build

Instructions here: https://github.com/hsbauer/cts2-example-service/archive/master.zip

MAYO CLINIC

Interactive Demo
Service Plugin

After running the clean install command your output should look something like this:

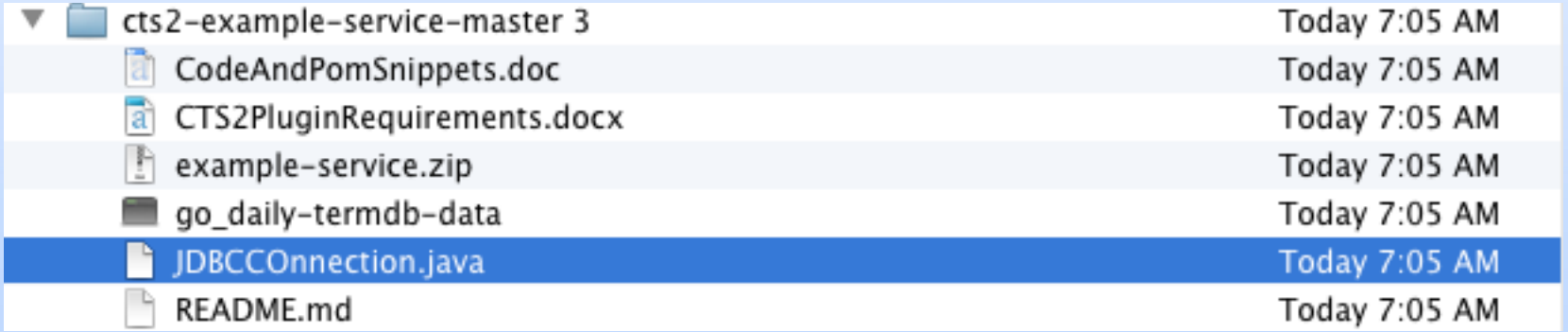
```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
2012-11-26 11:38:20.619:INFO:/webapp-rest:Destroying Spring FrameworkServlet 'appServlet'
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Closing WebApplicationContext for namespace
[INFO] Total time: 24.123s
[INFO] Finished at: Mon Nov 26 11:38:20 CST 2012
[INFO] Final Memory: 48M/99M
[INFO] ------------------------------------------------------------------------
2012-11-26 11:38:21.082:INFO::Shutdown hook executing
INFO : org.springframework.beans.factory.support.DefaultListableBeanFactory - Destroying singletons in org.springfram
2012-11-26 11:38:21.114:INFO:/webapp-rest:Closing Spring root WebApplicationContext
INFO : org.springframework.web.context.support.XmlWebApplicationContext - Closing Root WebApplicationContext: startup
INFO : org.springframework.beans.factory.support.DefaultListableBeanFactory - Destroying singletons in org.springfram
2012-11-26 11:38:21.218:INFO::Shutdown hook complete
```

MAYO CLINIC

Interactive Demo
Service Plugin

Connecting to a unique source.

- Copy the JDBConnnector.java file to the edu.mayo.cts2.framework.plugin.service.example folder of the example-service



MAYO CLINIC

Interactive Demo
Service Plugin

Adjust the hard coded connection properties to your local MySQL installation.

```
public Connection createConnection() {
    Connection con = null;
    if (connection != null) {
        System.out.println("Connection Failed");
    } else {
        try {
            con = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/godata", "root",
                    "lexgrid");
            System.out.println("Connected");
        } catch (SQLException e) {
            System.out.println(e.toString());
        }
    }
    return con;
```

MAYO CLINIC

Interactive Demo
Service Plugin

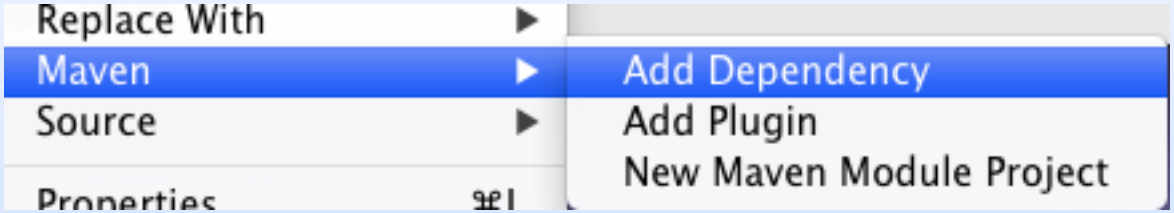
Running the main method we see we need a JDBC driver class.

```
java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost:3306/godata
Exception in thread "main" java.lang.NullPointerException
        at edu.mayo.cts2.framework.plugin.service.example.JDBCCOnnection.getResult(JDBCCOnnection.java:
        at edu.mayo.cts2.framework.plugin.service.example.JDBCCOnnection.main(JDBCCOnnection.java:70)
```

MAYO CLINIC

Interactive Demo
Service Plugin

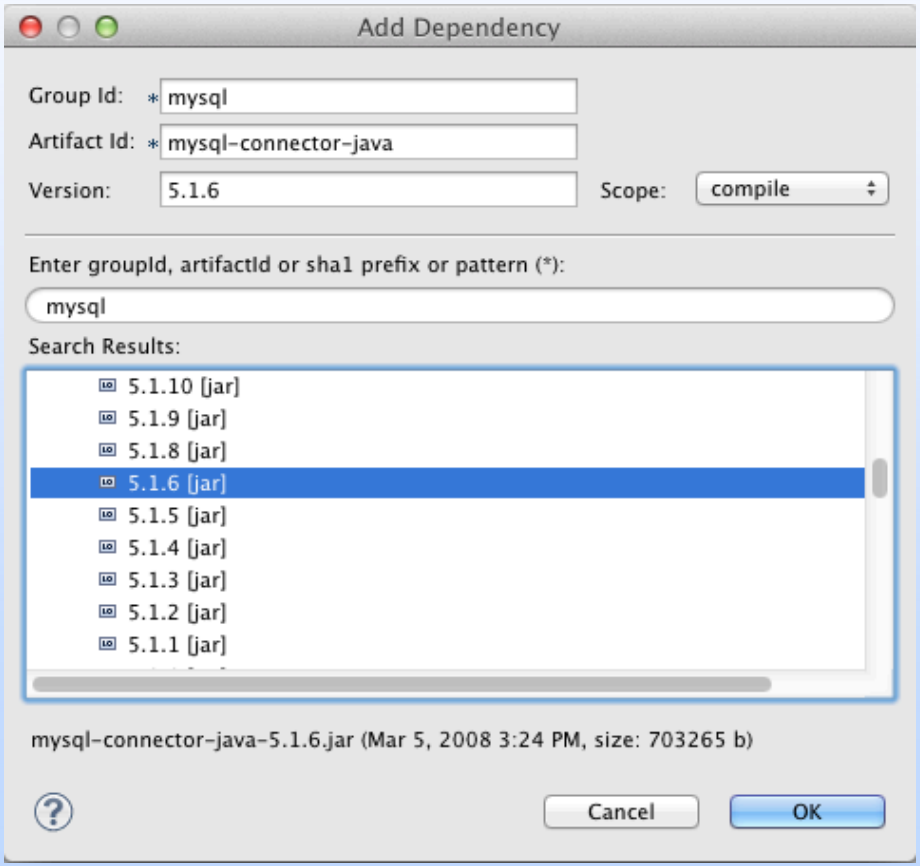
Using Eclipse as our example we'll add a dependency to the pom.xml file:



MAYO CLINIC

Interactive Demo
Service Plugin

Add the MySQL connector version 5.1.6 to the pom file:



Interactive Demo
Service Plugin

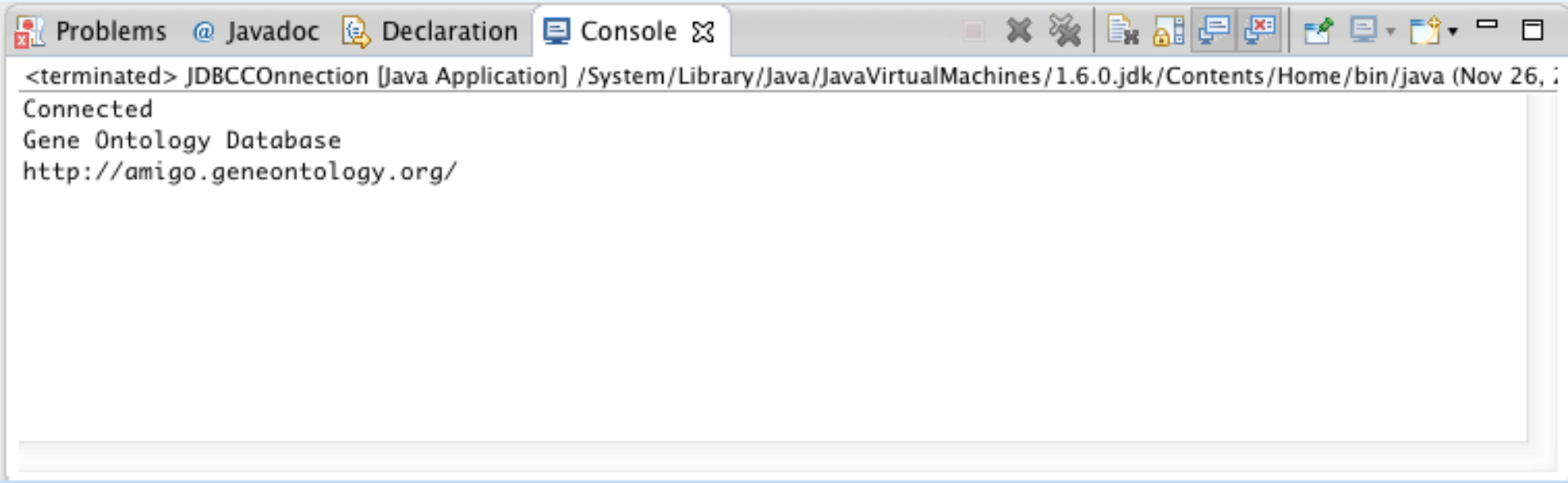
Alternatively you can cut and paste from the CodeAndPomSnippets file:

```
<!--This adds the third party jar to the plugin -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.6</version>
        </dependency>
```

MAYO CLINIC

Interactive Demo
Service Plugin

Running the main method against the “godata” database should give you output like this:



MAYO CLINIC

Interactive Demo
Service Plugin

This gives us local runtime access to the connector – Let's add it to the build using the OSGI bundling plugin (maven-bundle-plugin) and leave the other dependencies unresolved. We do this by adding it to the pom file:

```
<Import-Package>
    edu.mayo.cts2.framework.service.provider,
    *;resolution:=optional
</Import-Package>
<Embed-Dependency>
  mysql-connector-java
</Embed-Dependency>
```

MAYO CLINIC

Interactive Demo
Service Plugin

- The current mapping of the CTS2 CodeSystemCatalogEntry object is hardcoded
- Change that to a dynamic method:
  - Copy and paste over the entire read method with the snippet from the CodeAndPomSnippet's file
  - This runs a query against the database depending upon the abbreviated name of a code system
- Run the Maven clean and install methods

MAYO CLINIC

Interactive Demo
Service Plugin

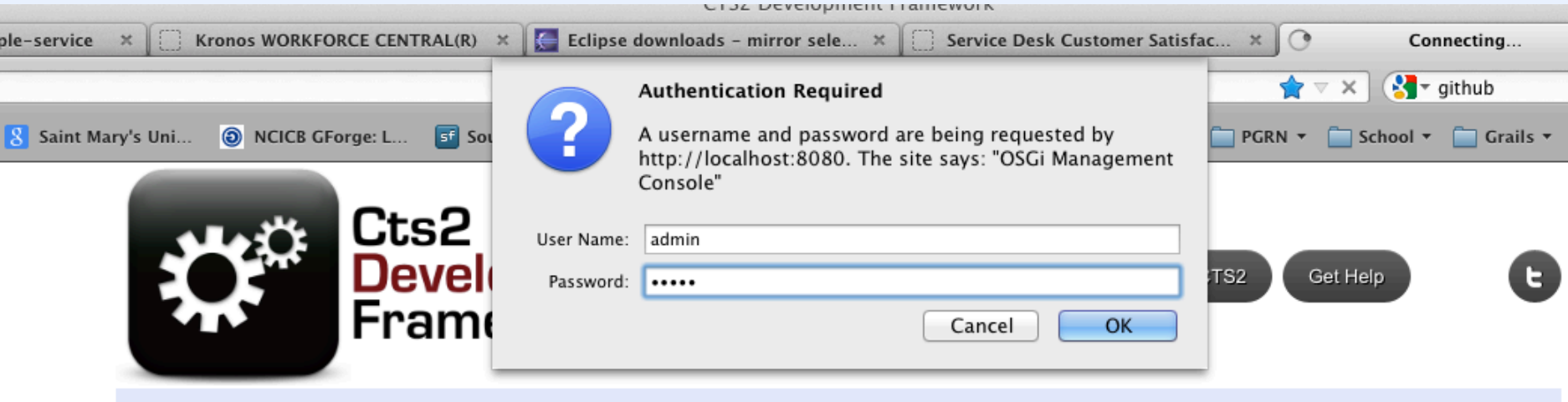
Start the standalone server:

```
r0223758:software m029206$ java -XX:PermSize=128m  -XX:MaxPermSize=512m -jar
 cts2framework-standalone.jar
```

MAYO CLINIC

Interactive Demo
Service Plugin

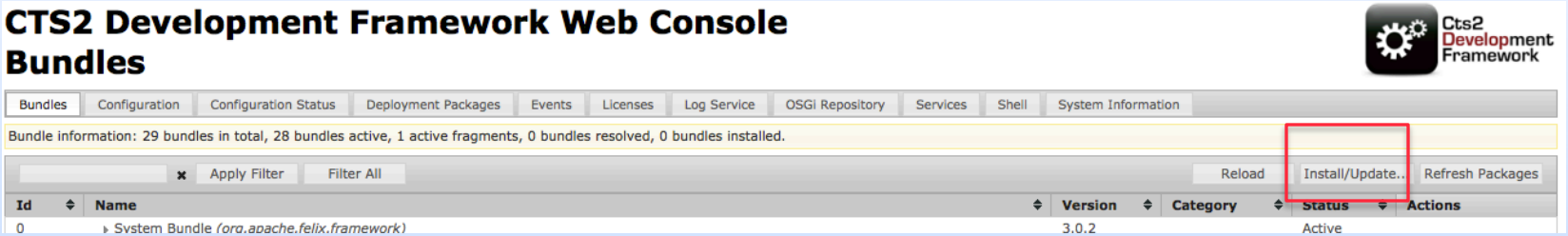
Click on the Admin Console button on the resulting web page:



Login using admin/admin.

Interactive Demo: Service Plugin
Import Plugin to Framework

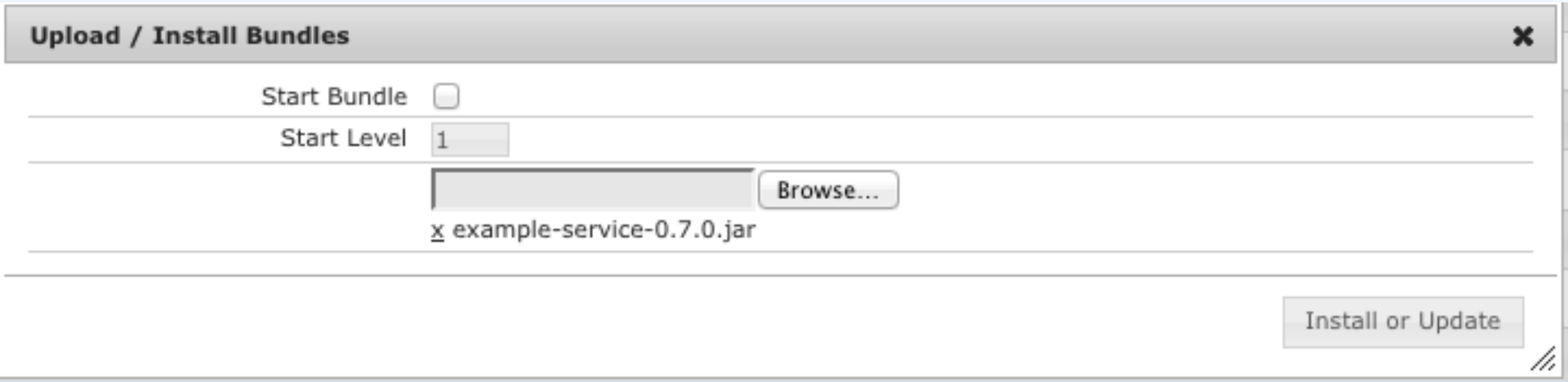
Click on install/update



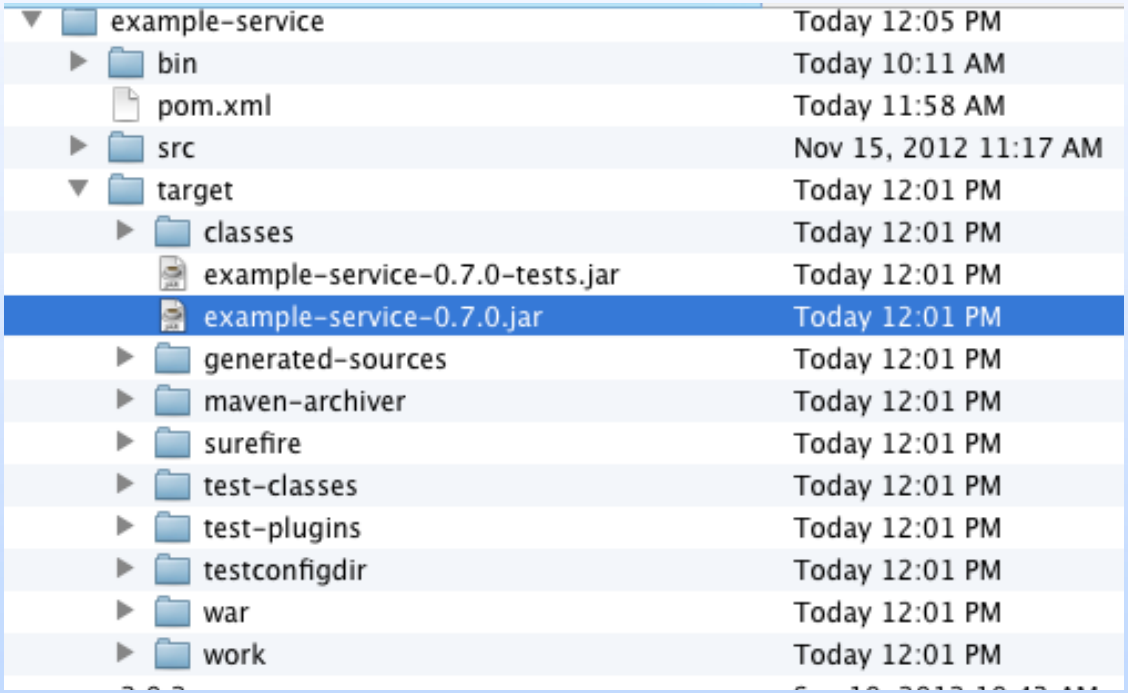
MAYO CLINIC

Interactive Demo: Service Plugin
Import Plugin to Framework

Upload the jar



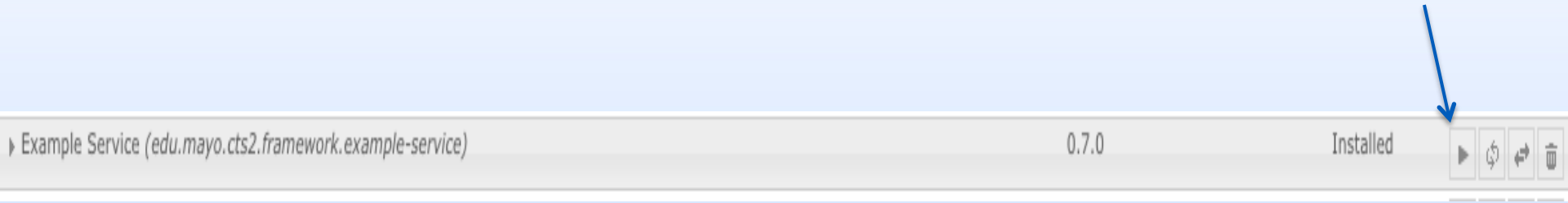
Interactive Demo: Service Plugin
Import Plugin to Framework



MAYO CLINIC

Interactive Demo: Service Plugin
Import Plugin to Framework

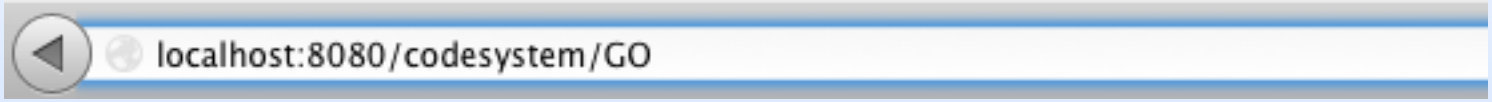
If necessary start the plugin:



MAYO CLINIC

Interactive Demo: Service Plugin
Import Plugin to Framework

Create a REST call in a browser address bar:



MAYO CLINIC

Interactive Demo
Service Plugin

Expected Results

```
– <CodeSystemCatalogEntryMsg xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/CodeSystem http://informatics.ma
  – <core:heading>
      <core:resourceRoot>codesystem/GO</core:resourceRoot>
      <core:resourceURI>http://localhost:8080/codesystem/GO</core:resourceURI>
      <core:accessDate>2012-11-26T12:23:37.669-06:00</core:accessDate>
    </core:heading>
  – <codeSystemCatalogEntry entryState="ACTIVE" about="Gene Ontology Database GO" codeSystemName="GO">
    – <core:resourceSynopsis>
        <core:value>http://amigo.geneontology.org/ GO</core:value>
      </core:resourceSynopsis>
    </codeSystemCatalogEntry>
  </CodeSystemCatalogEntryMsg>
```

MAYO CLINIC

Interactive Demo
Service Plugin

What have we learned:

- What class needs to be extended to create a plugin class
- How to create a mapping from a source to a CTS2 compatible element
- How to import third party software into the OSGI bundling system using maven
- How to run the CTS2 Development Framework Standalone server and upload a plugin to it

MAYO CLINIC

CTS2 REST API

MAYO CLINIC

CTS2 REST API

CTS2 is designed on a REST architecture:

- This shapes method calls and naming conventions
  - A directory is a summary of a group of objects, such as code systems or entities. A catalog (list) will provide complete objects
  - Read services generally are looking for a unique identifier for input
  - Query services offer filters on a matching value

MAYO CLINIC

Interactive Demo: CTS2 REST API

- The focus will be on read and query module functions
- Entity, code system(s), code system versions and value sets to comprise the examples
- REST calls will be to the NCBO Bioportal CTS2 REST wrapper

MAYO CLINIC

CTS2 REST API

How do we know what systems are at a service?

- Start by querying the code systems
- http://informatics.mayo.edu/cts2/rest/codesystems
- This returns a directory, or list of summaries of the code systems on that service

MAYO CLINIC

```
CodeSystemCatalogEntryDirectory xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/CodeSystem http://informatics.mayo.
vn/trunk/cts2/spec/psm/rest/schema/CodeSystem.xsd" complete="PARTIAL" numEntries="50" next="http://informatics.mayo.edu/cts2/
odesystems?page=1&maxtoreturn=50">
 <core:heading>
    <core:resourceRoot>codesystems</core:resourceRoot>
    <core:resourceURI>http://informatics.mayo.edu/cts2/rest/codesystems</core:resourceURI>
    <core:accessDate>2012-12-26T14:01:47.114-05:00</core:accessDate>
 </core:heading>
 <entry href="http://informatics.mayo.edu/cts2/rest/codesystem/EMAP" resourceName="EMAP" about="http://purl.bioontology.org
 /ontology/EMAP" formalName="Mouse gross anatomy and development" codeSystemName="EMAP">
  - <core:resourceSynopsis>
     - <core:value>
          A structured controlled vocabulary of stage-specific anatomical structures of the mouse (Mus).
       </core:value>
    </core:resourceSynopsis>
  - <versions>
       http://informatics.mayo.edu/cts2/rest/codesystem/EMAP/versions
    </versions>
  - <currentVersion>
       <core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/EMAP/version/45267">EMAP_45267_OBO</core:version
       <core:codeSystem uri="http://purl.bioontology.org/ontology/EMAP" href="http://informatics.mayo.edu/cts2/rest/codesystem
       /EMAP">EMAP</core:codeSystem>
    </currentVersion>
 </entry>
 <entry href="http://informatics.mayo.edu/cts2/rest/codesystem/EMO" resourceName="EMO" about="http://purl.bioontology.org
 /ontology/EMO" formalName="Enzyme Mechanism Ontology" codeSystemName="EMO">
  - <core:resourceSynopsis>
```

CTS2 REST API

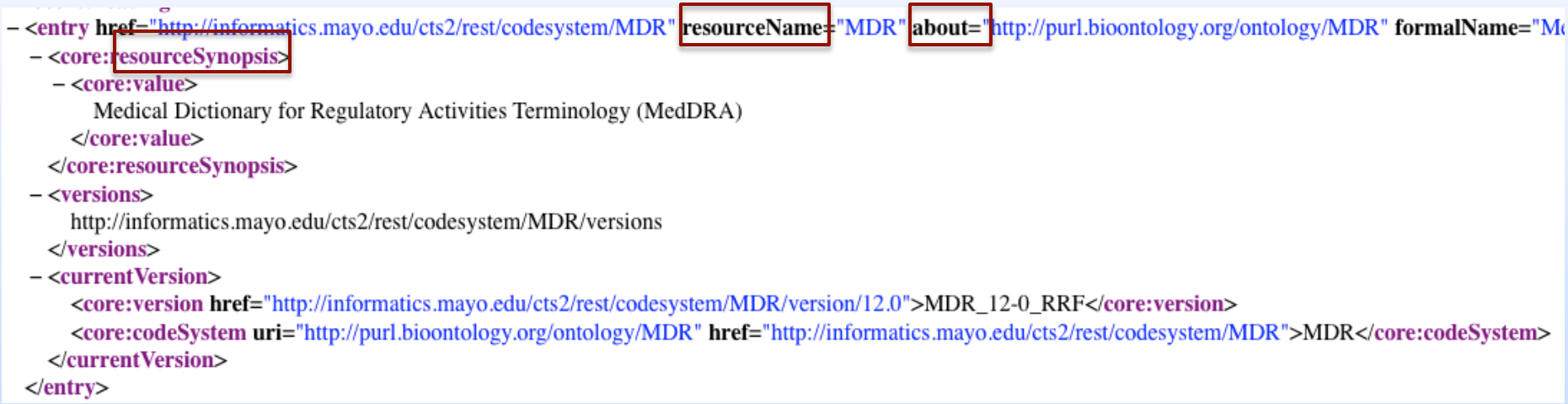
How can we query individual code systems from this service?

- Working through CodeSystem and CodeSystems REST APIs
  - By name, ontologyId, or description
  - By matching algorithms “contains” or “exact match”

MAYO CLINIC

CTS2 REST API

Searchable fields in the CodeSystemDirectoryEntry



MAYO CLINIC

CTS2 REST API

Working through codesystems, name is matched against the abbreviated ontology designation.

Example: MedDRA is MDR.

- http://informatics.mayo.edu/cts2/rest/codesystems?matchvalue=MDR&filtercomponent=resourceName
- Where matchvalue is the text to match and resourceName is the name field to match

MAYO CLINIC

```
- <CodeSystemCatalogEntryDirectory xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/CodeSystem http://informatics.mayo.edu
  /svn/trunk/cts2/spec/psm/rest/schema/CodeSystem.xsd" complete="COMPLETE" numEntries="1">
  - <core:heading>
      <core:resourceRoot>codesystems</core:resourceRoot>
      <core:resourceURI>http://informatics.mayo.edu/cts2/rest/codesystems</core:resourceURI>
    - <core:parameter arg="filtercomponent">
        <core:val>resourceName</core:val>
      </core:parameter>
    - <core:parameter arg="matchvalue">
        <core:val>MDR</core:val>
      </core:parameter>
      <core:accessDate>2012-12-21T16:39:44.557-05:00</core:accessDate>
    </core:heading>
  - <entry href="http://informatics.mayo.edu/cts2/rest/codesystem/MDR" resourceName="MDR" about="http://purl.bioontology.org
    /ontology/MDR" formalName="MedDRA" codeSystemName="MDR">
    - <core:resourceSynopsis>
      - <core:value>
          Medical Dictionary for Regulatory Activities Terminology (MedDRA)
        </core:value>
      </core:resourceSynopsis>
    - <versions>
        http://informatics.mayo.edu/cts2/rest/codesystem/MDR/versions
      </versions>
    - <currentVersion>
        <core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/MDR/version/12.0">MDR_12-0_RRF</core:version>
        <core:codeSystem uri="http://purl.bioontology.org/ontology/MDR" href="http://informatics.mayo.edu/cts2/rest/codesystem
        /MDR">MDR</core:codeSystem>
```

CTS2 REST API

Using the codesystems API, the other "filtercomponents" we can match are:

- resourceSynopsis: a description of the resource
- resourceName: the abbreviation of the code system
- about: the URI of the code system
- keyword: contents of the keyword tag
- ontologyId: unique ID of the ontology in the service

MAYO CLINIC

CTS2 REST API

Direct read calls to the code system API require a a similar knowledge of the abbreviated name of the code system:

- http://informatics.mayo.edu/cts2/rest/codesystem/VANDF
- You can get this information by first making a call to the codesystems and returning a directory of codesystems

MAYO CLINIC

```
- <CodeSystemCatalogEntryMsg xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/CodeSystem http://informatics.mayo.edu
  /svn/trunk/cts2/spec/psm/rest/schema/CodeSystem.xsd">
  - <core:heading>
      <core:resourceRoot>codesystem/VANDF</core:resourceRoot>
    - <core:resourceURI>
        http://informatics.mayo.edu/cts2/rest/codesystem/VANDF
      </core:resourceURI>
      <core:accessDate>2012-12-21T16:43:50.197-05:00</core:accessDate>
    </core:heading>
  - <codeSystemCatalogEntry entryState="ACTIVE" about="http://purl.bioontology.org/ontology/VANDF" formalName="VANDF"
    codeSystemName="VANDF">
      <core:keyword>VANDF</core:keyword>
      <core:keyword>1527</core:keyword>
    - <core:resourceSynopsis>
        <core:value>Veterans Health Administration National Drug File</core:value>
      </core:resourceSynopsis>
    - <core:sourceAndRole>
        <core:source uri="mailto:michael.lincoln@med.va.gov">Michael Lincoln</core:source>
        <core:role>contact</core:role>
      </core:sourceAndRole>
    - <core:property>
      - <core:predicate uri="http://purl.bioontology.org/predicate/isFlat">
          <core:namespace>VANDF</core:namespace>
          <core:name>isFlat</core:name>
        </core:predicate>
      - <core:value>
        - <core:literal>
```

CTS2 REST API

A codesystemversions query has a similar query interface for getting all the versions on a given service

- http://server root/codesystemversions

A more likely use case might be to get all versions of a given code system:

- http://informatics.mayo.edu/cts2/rest/codesystem/LNC/versions

MAYO CLINIC

```
- <CodeSystemVersionCatalogEntryDirectory xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/CodeSystemVersion
  http://informatics.mayo.edu/svn/trunk/cts2/spec/psm/rest/schema/CodeSystemVersion.xsd" complete="COMPLETE" numEntries="4">
   - <core:heading>
       <core:resourceRoot>codesystem/LNC/versions</core:resourceRoot>
     - <core:resourceURI>
         http://informatics.mayo.edu/cts2/rest/codesystem/LNC/versions
       </core:resourceURI>
       <core:accessDate>2012-12-21T16:45:47.514-05:00</core:accessDate>
     </core:heading>
   - <entry href="http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/229" resourceName="LNC"
     about="http://purl.bioontology.org/ontology/LNC" formalName="Logical Observation Identifier Names and Codes"
     documentURI="urn:oid:2.16.840.1.113883.6.1|229" codeSystemVersionName="LNC_229_RRF">
     - <core:resourceSynopsis>
       - <core:value>
           Logical Observation Identifier Names and Codes (LOINC);Version 2.26;January 2, 2009
         </core:value>
       </core:resourceSynopsis>
       <core:officialResourceVersionId>229</core:officialResourceVersionId>
       <versionOf href="http://informatics.mayo.edu/cts2/rest/codesystem/LNC">LNC</versionOf>
       <codeSystemVersionTag/>
     </entry>
   - <entry href="http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/232" resourceName="LNC"
     about="http://purl.bioontology.org/ontology/LNC" formalName="Logical Observation Identifier Names and Codes"
     documentURI="urn:oid:2.16.840.1.113883.6.1|232" codeSystemVersionName="LNC_232_RRF">
     - <core:resourceSynopsis>
       - <core:value>
           Logical Observation Identifier Names and Codes (LOINC);Version 2.26;January 2, 2009
```

CTS2 REST API

- Matchvalues options for codesystemversions include resourceName, about and resourceSynopsis
- Perhaps more useful is to get the versions of a given code system as above, then use the href of the version designation to return the desired version http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/232
- This also happens to be the syntax for the code system version read function

MAYO CLINIC

```
– <CodeSystemVersionCatalogEntryMsg xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/CodeSystemVersion
  http://informatics.mayo.edu/svn/trunk/cts2/spec/psm/rest/schema/CodeSystemVersion.xsd">
  – <core:heading>
      <core:resourceRoot>codesystem/LNC/version/232</core:resourceRoot>
    – <core:resourceURI>
        http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/232
      </core:resourceURI>
      <core:accessDate>2012-12-21T16:52:27.553-05:00</core:accessDate>
    </core:heading>
  – <codeSystemVersionCatalogEntry entryState="ACTIVE" about="http://purl.bioontology.org/ontology/LNC" formalName="Logical
    Observation Identifier Names and Codes" documentURI="urn:oid:2.16.840.1.113883.6.1|232" state="FINAL"
    codeSystemVersionName="LNC_232_RRF">
      <core:keyword>LNC</core:keyword>
      <core:keyword>1350</core:keyword>
      <core:keyword>44774</core:keyword>
    – <core:resourceSynopsis>
      – <core:value>
          Logical Observation Identifier Names and Codes (LOINC);Version 2.26;January 2, 2009
        </core:value>
      </core:resourceSynopsis>
    – <core:sourceAndRole>
        <core:source uri="mailto:loinc@regenstrief.org">Ms. Kathy Mercer, LOINC Developer</core:source>
        <core:role>contact</core:role>
      </core:sourceAndRole>
    – <core:property>
      – <core:predicate uri="http://purl.bioontology.org/predicate/isFoundry">
          <core:namespace>LNC</core:namespace>
```

CTS2 REST API

Most users may simply either want to get an entity (a concept) by text match from a vocabulary service.

- http://informatics.mayo.edu/cts2/rest/entities?matchvalue=swelling

Or they may want to get an entity from a particular version of a particular code system:

http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/232/entities?matchvalue=Cefoperazone

MAYO CLINIC

```
– <EntityDirectory xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/Entity http://informatics.mayo.edu/svn/trunk/cts2/spec
  /psm/rest/schema/Entity.xsd" complete="COMPLETE" numEntries="19">
  – <core:heading>
      <core:resourceRoot>codesystem/LNC/version/232/entities</core:resourceRoot>
    – <core:resourceURI>
        http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/232/entities
      </core:resourceURI>
    – <core:parameter arg="matchvalue">
        <core:val>Cefoperazone</core:val>
      </core:parameter>
      <core:accessDate>2012-12-21T16:54:49.073-05:00</core:accessDate>
    </core:heading>
  – <entry about="http://purl.bioontology.org/ontology/LNC/LP14524-0" href="http://informatics.mayo.edu/cts2/rest/codesystem
    /LNC/version/236/entity/LP14524-0">
    – <core:name>
        <core:namespace>LNC</core:namespace>
        <core:name>LP14524-0</core:name>
      </core:name>
    – <core:knownEntityDescription>
      – <core:describingCodeSystemVersion>
          <core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/236">LNC_236_UMLS-
          RELA</core:version>
          <core:codeSystem uri="http://purl.bioontology.org/ontology/LNC" href="http://informatics.mayo.edu/cts2/rest/codesystem
          /LNC">LNC</core:codeSystem>
        </core:describingCodeSystemVersion>
        <core:designation>Cefoperazone</core:designation>
      </core:knownEntityDescription>
```

CTS2 REST API

Having chosen a desired entity from that list we can resolve the full entity with an entity read:

- http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/232/entity/100-8

MAYO CLINIC

```
– <EntityDescriptionMsg xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/Entity http://informatics.mayo.edu/svn/trunk/cts2/spec
  /psm/rest/schema/Entity.xsd">
   – <core:heading>
       <core:resourceRoot>codesystem/LNC/version/232/entity/100-8</core:resourceRoot>
     – <core:resourceURI>
         http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/232/entity/100-8
       </core:resourceURI>
       <core:accessDate>2012-12-21T16:56:37.710-05:00</core:accessDate>
     </core:heading>
   – <EntityDescription>
     – <namedEntity about="http://purl.bioontology.org/ontology/LNC/100-8" entryState="ACTIVE">
       – <entityID>
           <core:namespace>LNC</core:namespace>
           <core:name>100-8</core:name>
         </entityID>
       – <describingCodeSystemVersion>
           <core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/232">LNC_232_RRF</core:version>
           <core:codeSystem uri="http://purl.bioontology.org/ontology/LNC" href="http://informatics.mayo.edu/cts2/rest/codesystem
           /LNC">LNC</core:codeSystem>
         </describingCodeSystemVersion>
       – <designation designationRole="PREFERRED">
         – <core:value>
             Cefoperazone:Susceptibility:Point in time:Isolate:Quantitative or Ordinal:Minimum Inhibitory Concentration
           </core:value>
         </designation>
       – <designation designationRole="ALTERNATIVE">
           <core:value>Cefoperazone:Susc:Pt:Isolate:OrdQn:MIC</core:value>
```

CTS2 REST API

This lays the foundation for a couple of likely scenarios. Starting from the code system:

- Display available coding schemes to user
  - User chooses coding schemes
- Display versions to user
  - User chooses code system version
- Display text match for concept search
  - User initiates text search and makes choice from matches

MAYO CLINIC

CTS2 REST API

Alternatively:

- Display text match field to user
  - User enters text match
- Display list of entity matches
  - User chooses match
- Display selected user friendly entity attributes and properties to user

MAYO CLINIC

CTS2 REST API

Developer Strategies:

- Parse initial XML for resourceName and construct REST queries
- Parse the href from each set of queries and use the resulting XML to resolve further. (Remember each result has href(s) pointing to new resource options.)

MAYO CLINIC

CTS2 REST API
Value Sets

Using valueset and valuesets and resolvedvalueset API's:

- What value sets are on the service:
- http://informatics.mayo.edu/cts2/rest/valuesets

```
– <entry href="http://informatics.mayo.edu/cts2/rest/valueset/provenance" resourceName="provenance" about="http://purl.b
  – <core:resourceSynopsis>
    – <core:value>
        This file is imported by vivo-core-public-1.5.owl. It contains terms related to people for the purpose of accumulating
      </core:value>
    </core:resourceSynopsis>
  </entry>
```

- href value can be resolved to an individual value set

MAYO CLINIC

CTS2 REST API
Value Sets: Pagination

Parsing values for “complete” and “next” tag will allow developers to page through results:

```
- <ResolvedValueSetDirectory xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/ValueSetDefinition http://informatics.mayo
  complete="PARTIAL" numEntries="50" next="http://informatics.mayo.edu/cts2/rest/resolvedvaluesets?page=1&maxtoreturn=50">
  - <core:heading>
      <core:resourceRoot>resolvedvaluesets</core:resourceRoot>
    - <core:resourceURI>
        http://informatics.mayo.edu/cts2/rest/resolvedvaluesets
      </core:resourceURI>
      <core:accessDate>2012-11-30T09:31:15.092-05:00</core:accessDate>
    </core:heading>
```

If complete value is “PARTIAL” then you can parse and resolve next value.

MAYO CLINIC

CTS2 REST API
Value Sets

Getting a value set (using the href value):

http://informatics.mayo.edu/cts2/rest/valueset/provenance


```
– <ValueSetCatalogEntryMsg xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/ValueSet http://informatics.mayo.edu/svn/trunk
  /cts2/spec/psm/rest/schema/ValueSet.xsd">
  – <core:heading>
      <core:resourceRoot>valueset/provenance</core:resourceRoot>
    – <core:resourceURI>
        http://informatics.mayo.edu/cts2/rest/valueset/provenance
      </core:resourceURI>
      <core:accessDate>2012-12-21T16:59:14.292-05:00</core:accessDate>
    </core:heading>
  – <valueSetCatalogEntry entryState="ACTIVE" about="http://purl.bioontology.org/view/provenance" formalName="Provenance"
    valueSetName="provenance">
```


MAYO CLINIC

CTS2 REST API
Value Sets

Resolving a value set (adding the resolution parameter):

http://informatics.mayo.edu/cts2/rest/valueset/provenance/resolution

MAYO CLINIC

```
- <IteratableResolvedValueSet xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/ValueSetDefinition http://informatics.mayo.edu
  /svn/trunk/cts2/spec/psm/rest/schema/ValueSetDefinition.xsd" complete="COMPLETE" numEntries="3">
  - <core:heading>
    - <core:resourceRoot>
        valueset/provenance/definition/provenance_1-5_OWL/resolution
      </core:resourceRoot>
    - <core:resourceURI>
        http://informatics.mayo.edu/cts2/rest/valueset/provenance/definition/provenance_1-5_OWL/resolution
      </core:resourceURI>
      <core:accessDate>2012-12-21T17:04:05.481-05:00</core:accessDate>
    </core:heading>
  - <resolutionInfo>
    - <resolutionOf>
        <core:valueSetDefinition>provenance_1-5_OWL</core:valueSetDefinition>
        <core:valueSet uri="http://purl.bioontology.org/view/provenance" href="http://informatics.mayo.edu/cts2/rest/valueset
        /provenance">provenance</core:valueSet>
      </resolutionOf>
    - <resolvedUsingCodeSystem>
        <core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/vivo/version/1.5">vivo_1-5_OWL</core:version>
        <core:codeSystem uri="http://purl.bioontology.org/ontology/vivo" href="http://informatics.mayo.edu/cts2/rest/codesystem
        /vivo">vivo</core:codeSystem>
      </resolvedUsingCodeSystem>
    </resolutionInfo>
  - <entry uri="http://vivoweb.org/ontology/core#Authorship" href="http://informatics.mayo.edu/cts2/rest/codesystem/provenance/version
    /1.5/entity/vivo:Authorship">
      <core:namespace>vivo</core:namespace>
      <core:name>Authorship</core:name>
```

CTS2 REST API
Value Sets

Getting the resolved value set objects:

http://informatics.mayo.edu/cts2/rest/resolvedvaluesets

Resolution using the href value from these results gives direct access to the resolved value set.

MAYO CLINIC

```
- <ResolvedValueSetDirectory xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/ValueSetDefinition http://informatics.mayo.edu
  /svn/trunk/cts2/spec/psm/rest/schema/ValueSetDefinition.xsd" complete="PARTIAL" numEntries="50" next="http://informatics.mayo.edu
  /cts2/rest/resolvedvaluesets?page=1&maxtoreturn=50">
  - <core:heading>
      <core:resourceRoot>resolvedvaluesets</core:resourceRoot>
    - <core:resourceURI>
        http://informatics.mayo.edu/cts2/rest/resolvedvaluesets
      </core:resourceURI>
      <core:accessDate>2012-12-21T17:21:24.014-05:00</core:accessDate>
    </core:heading>
  - <entry resolvedValueSetURI="http://purl.bioontology.org/view/BRO-Chinese/resolution/41005" href="http://informatics.mayo.edu
    /cts2/rest/valueset/BRO-Chinese/definition/BRO-Chinese_2-5_OWL/resolution/41005">
    - <resolvedHeader>
      - <resolutionOf>
          <core:valueSetDefinition>BRO-Chinese_2-5_OWL</core:valueSetDefinition>
          <core:valueSet uri="http://purl.bioontology.org/view/BRO-Chinese" href="http://informatics.mayo.edu/cts2/rest/valueset
          /BRO-Chinese">BRO-Chinese</core:valueSet>
        </resolutionOf>
      - <resolvedUsingCodeSystem>
          <core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/BRO/version/2.7.1">BRO_2-7-1_OWL-
          FULL</core:version>
          <core:codeSystem uri="http://purl.bioontology.org/ontology/BRO" href="http://informatics.mayo.edu/cts2/rest/codesystem
          /BRO">BRO</core:codeSystem>
        </resolvedUsingCodeSystem>
      </resolvedHeader>
    </entry>
  - <entry resolvedValueSetURI="http://purl.bioontology.org/view/OGMD_OWL/resolution/43092" href="http://informatics.mayo.edu
```

CTS2 REST API
Summary

- Developers should now be able to text match search on CTS2 module attributes
- Retrieve coding schemes, entities, and value sets
- Page through large result sets
- Follow href values to other module definitions as they are available

MAYO CLINIC

CTS2 Java REST Client

MAYO CLINIC

CTS2 REST Client
Java

- Create a new Maven project in Eclipse using cts2.rest as the group and artifact id
- Check the box that allows you to make a simple Maven project (No Archetypes)
- Create a class in src/main/java with a main method

(Code is found in the cts2-example-service zip ValueSetClient.java)

CTS2 REST Client
Java

Create a method called getValueSets with a return value of void.

```
public void getValueSets()
```

You'll need some kind of REST URL to resolve:

```
String uri = "
http://informatics.mayo.edu/cts2/rest/valuesets";
```

And convert to a Java URL

```
URL url;

try { url = new URL(uri);
```

CTS2 REST Client
Java

Create a connection:

```
HttpURLConnection connection =
(HttpURLConnection) url.openConnection();
```

Test it:

```
if (connection.getResponseCode() != 200) { throw new
RuntimeException("Failed : The HTTP error code is :
" + connection.getResponseCode()); }
```

MAYO CLINIC

CTS2 REST Client
Java

Stream it into a buffered reader:

```
BufferedReader br = new BufferedReader(new
InputStreamReader( (connection.getInputStream())));
```

Here the output can be printed:

```
String output;

System.out.println("Output from Server .... \n"); while
((output = br.readLine()) != null)
{ System.out.println(output); }
```

MAYO CLINIC

CTS2 REST Client
Java

The normal use case for XML consumption is to marshal to java objects.

Create a new method: *getValueSet()*

We'll make a REST call that returns JSON and consume it using some CTS2 specific code.

```
String uri = "
http://informatics.mayo.edu/cts2/rest/valuesets?
matchvalue=Sequence&format=json";
```

And add a request parameter to the connection:

```
connection.setRequestProperty("Accept", "text/json");
```

MAYO CLINIC

CTS2 REST Client
Java

- We'll handle the buffered output differently:

```
JsonConverter converter = new JsonConverter();

        ValueSetCatalogEntryDirectory valuesetcat =
converter.fromJson(builder.toString(),
ValueSetCatalogEntryDirectory.class);
```

- Uses CTS2 Development Framework classes to handle parsing and marshaling duties

MAYO CLINIC

CTS2 REST Client
Java

This will cause some compiling errors, so we'll add some values to the maven project. First add a repository:

```
<repositories>
    <repository>
        <id>edu.informatics.maven.release</id>
        <name>Informatics Maven Release Repository</name>
        <url>http://informatics.mayo.edu/maven/content/repositories/releases</url>
    </repository>
</repositories>
```

(From the CodeandPomSnippets file)

MAYO CLINIC

CTS2 REST Client
Java

Then add a single dependency:

```
<dependency>
    <groupId>edu.mayo.cts2.framework</groupId>
    <artifactId>core</artifactId>
    <version>0.8.0</version>
</dependency>
```

The CTS2 Development Framework solves a number of the parsing and marshaling problems so long as you marshal the XML or JSON into a CTS2 object.

MAYO CLINIC

CTS2 REST Client
Java

Add code to print some output:

```
List<ValueSetCatalogEntrySummary> sum =
valuesetcat.getEntryAsReference();

        for(ValueSetCatalogEntrySummary s : sum){

        System.out.println(s.getFormalName());

        System.out.println(s.getCurrentDefinition());

        System.out.println(s.getResourceName());

        System.out.println(s.getValueSetName());

                System.out.println(s.getHref());

        }
```

CTS2 REST Client
Java

And disconnect:

```
connection.disconnect();
```

MAYO CLINIC

CTS2 REST Client
Java

Create a main method and run one of the methods:

```
public static void main(String[] args){
        new ValueSetClient().getValueSet();
}
```

MAYO CLINIC

CTS2 REST Client
Java

Simple REST clients can be duplicated in any number of implementation environments.

See details of more Java, Python, Scala and Unix based curl command implementations here:

http://informatics.mayo.edu/CTS2_V3/index.php/Value_Set_REST_API_and_Implementation_Examples

MAYO CLINIC

CTS2 Implementation
Command Line and XSLT Transforms

MAYO CLINIC

CTS2 REST Client
Command Line

Windows, Linux and Unix can pull in CTS2 XML via command line REST calls.

On Linux and Unix the curl command can be used to pull xml into a file where an application can process it and serve it up to a user.

MAYO CLINIC

CTS2 REST Client
Command Line

- For example:
- curl http://informatics.mayo.edu/cts2/rest/valuesets > valuesets.xml
- Pipes a CTS2 value set into an xml document where it could be served up on an eXist XML database or processed into a more user friendly file using a XSLT style sheet

MAYO CLINIC

CTS2 REST Client
Command Line and XSLT

Exercise:

Use curl or a browser to get some CTS2 object in XML from a service

Use an XSLT transform to transform the values into an HTML representation.

MAYO CLINIC

CTS2 REST Client
Command Line and XSLT

Using Firefox open the following in tabs:

- http://informatics.mayo.edu/cts2/rest/valuesets
- http://markbucayan.appspot.com/xslt/index.html
- And the two XSLT files in the example folder
- Use the ValueSetCatalogEntryDirectory.xsl to transform the first XML page. Use the link to click through to an IterableResolvedValueSet and try the second XSL, IterableResVS

MAYO CLINIC

CTS2 Optional Tutorial Section

Python and Scala

MAYO CLINIC

Optional Tutorial
Platform Independence

- REST means platform independent clients
- REST can be consumed by any client that can make http or https connections and parse XML or JSON
- Python and Scala can both achieve this
- In the example folders there is sample code for Python and Scala REST consuming clients

Note: Not an interactive demo

MAYO CLINIC

Optional
Platform Independence: Python

Python, a C++ based scripting language, can create a simple REST client in a few lines of code

- This example depends on a library not included in the regular Python distribution:

from restful_lib import Connection

MAYO CLINIC

Optional
Platform Independence: Python

- Create the connection:

```
conn = Connection("http://informatics.mayo.edu/cts2/rest")

reply = conn.request_get("/valuesets?format=json",
    headers={'Accept':'application/json;q=1.0'})
```

MAYO CLINIC

Optional
Platform Independence: Python

Test the response and print it:

```
if reply['headers']['status'] == '200':
    print reply['body']
    print eval(reply['body'])
```

MAYO CLINIC

Optional
Platform Independence: Python

Here is a short, self-contained script that does a search on value sets matching the word “Sequence”:

```
conn = Connection("http://informatics.mayo.edu/cts2/rest")

print conn.request_get("/valuesets?matchvalue='Sequence'")['body']
```

MAYO CLINIC

Optional
Platform Independence: Python

Using a couple of integrated Python libraries you can also do authentication using https.

```
import urllib2, base64

request = urllib2.Request("https://informatics.mayo.edu/cts2/services/mat/valueset/2.16.840.1.113883.3.526.02.99/resolution")
base64string = base64.encodestring('%s:%s' % ("name", "password")).replace('\n', '')
request.add_header("Authorization", "Basic %s" % base64string)
result = urllib2.urlopen(request)
data = result.read()
print data
```

(Requires valid username and password)

MAYO CLINIC

Optional
Platform Independence: Scala

- Scala is a functional programming language that compiles to Java bytecode
- As such it can reuse some of the same Java libraries used in the Java example

MAYO CLINIC

Optional
Platform Independence: Scala

Import Java http and Scala input/output libraries:

```
import java.net.{HttpURLConnection, URL}
import io.Source
```

Create an executable Scala object:

```
object CTS2RestClient extends App{
}
```

MAYO CLINIC

Optional
Platform Independence: Scala

Add some code to connect to the service:

```
val connection =

    new URL("http://informatics.mayo.edu/cts2/rest/
valuesets").openConnection().asInstanceOf[HttpURL
Connection]
```

MAYO CLINIC

Optional
Platform Independence: Scala

Get the input Stream:

```
val inputStream = connection.getInputStream
```

```
val src = Source.fromInputStream(inputStream)
```

Print it out:

```
src.getLines().foreach(println)
```

MAYO CLINIC

Optional
Platform Independence: Next Steps

- Since output can be formatted to JSON, JavaScript is an obvious example
- This will be the subject of the next tutorial

MAYO CLINIC

CTS2 Development Framework Implementations, Web Applications & UI Widgets

Cory Endle

MAYO CLINIC

CTS2 Development Framework and Implementations

MAYO CLINIC

CTS2 Development Framework
Implementations

There are several existing CTS2 Development Framework Implementations:

- eXist database
- Meaningful Use Quality Measure Value Set Service
- NCBO BioPortal REST
- NCBO BioPortal RDF

MAYO CLINIC

CTS2 Development Framework Implementations

eXist Database

eXist XML Database Plugin:

- A CTS2 implementation based on the eXist Database (an Open Source Native XML Database)
- Supports:
  - READ
  - QUERY
  - MAINTENANCE

MAYO CLINIC

CTS2 Development Framework Implementations

MU Quality Measure Value Set Service

- A CTS2 Query implementation of Value Sets based on Meaningful Use 2 Standards for NQF eMeasures
- Value Sets provided by US National Library of Medicine (NLM)
- Supports:
    - READ
    - QUERY
- https://informatics.mayo.edu/cts2/services/mat/valuesets

MAYO CLINIC

CTS2 Development Framework Implementations

NCBO BioPortal

Component of the National Center for Biomedical Ontologies (NCBO)

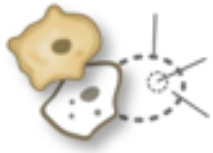
Mission:

- Create software and support services for the application of principled ontologies in biomedical science and clinical care, ranging from tools for application developers to software for end-users

MAYO CLINIC

CTS2 Development Framework Implementations

NCBO BioPortal



- Web-based application for accessing and sharing biomedical ontologies:
  - Search for terms
  - Submit a new ontology to BioPortal library
  - Views on large ontologies
  - Explore mappings between ontologies
- REST based API
- Many useful viewing, browsing and access widgets

MAYO CLINIC

CTS2 Development Framework Implementations

NCBO BioPortal REST

- NCBO BioPortal Wrapper
  - A CTS2 Implementation based on the NCBO BioPortal REST service
  - Supports:
    - READ
    - QUERY
- http://informatics.mayo.edu/cts2/rest/valuesets

MAYO CLINIC

CTS2 Development Framework Implementations
NCBO BioPortal RDF

- RDF Triplestore Wrapper
  - A CTS2 implementation based on the NCBO BioPortal SPARQL endpoint
  - Supports:
    - READ
    - QUERY
- http://informatics.mayo.edu/cts2/services/bioportal-rdf/valuesets

MAYO CLINIC

CTS2 Implementations

- SNOMED CT RF2 (py4cts2)
  - A Python implementation of CTS2 to expose the the contents of the SNOMED CT RF2 distribution
  - This service provides REST based access to the SNOMED CT RF2 tables and their equivalent CTS2 rendering
  - Supports READ and QUERY

MAYO CLINIC

Web Applications using the CTS2 Service

MAYO CLINIC

Web Applications using the CTS2 Service

- Now that we have an implementation of CTS2, we will want to display its content in a user friendly manner
- Developed 2 different web applications that display CTS2 content
  - CTS2 Value Set Viewer
  - High-Throughput Phenotyping (HTP)

MAYO CLINIC

UIs created for CTS2 Implementations
CTS2 Value Set Viewer

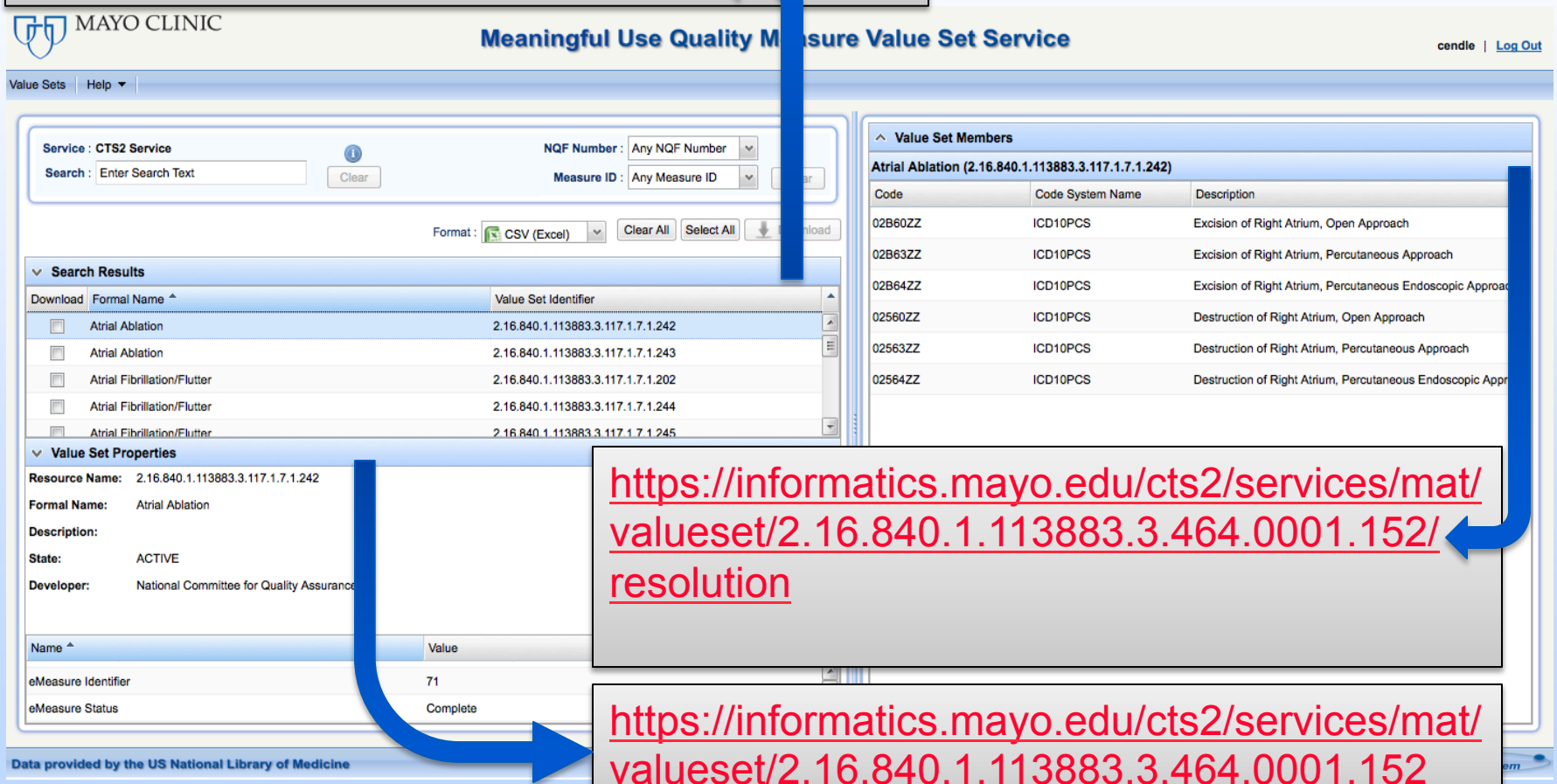
- Generic Value Set Browser/export
- Initially created for Meaningful Use (MU) Quality Measure Value Set Service to serve NLM Value Sets
  - Can view any CTS2 compliant Value Set service (NCBO,PHINVADS CDC,VSAC)
- https://informatics.mayo.edu/vsmc

MAYO CLINIC

UIs created for CTS2 Implementations

CTS2 Value Set Viewer

https://informatics.mayo.edu/cts2/services/mat/valuesets?&maxtoreturn=100

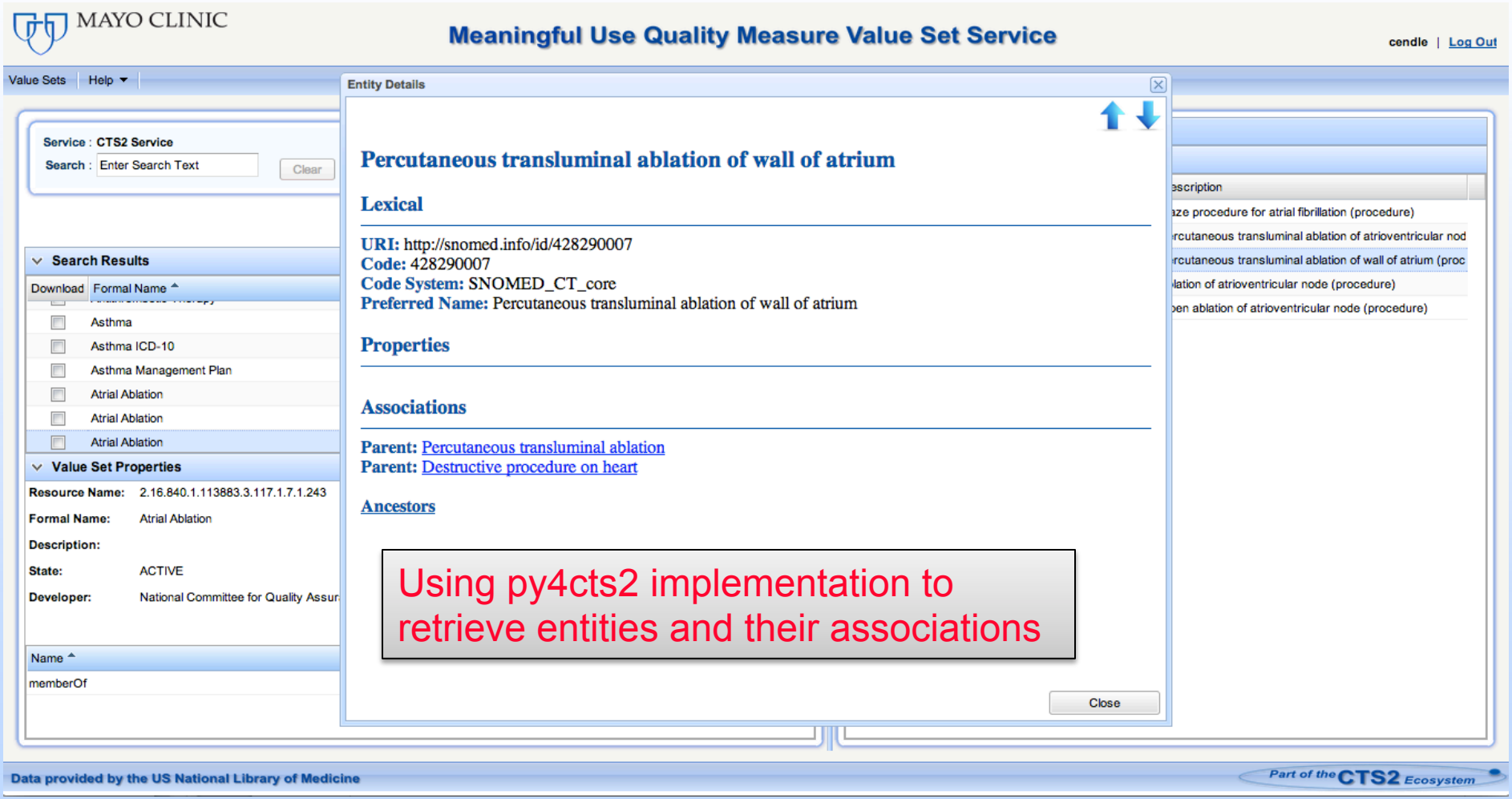


https://informatics.mayo.edu/cts2/services/mat/valueset/2.16.840.1.113883.3.464.0001.152/resolution

https://informatics.mayo.edu/cts2/services/mat/valueset/2.16.840.1.113883.3.464.0001.152

UIs created for CTS2 Implementations

CTS2 Value Set Viewer



UIs created for CTS2 Implementations
High-Throughput Phenotyping (HTP) Overview

- A tool to help identify patient cohorts using electronic health record (EHR) data by leveraging informatics-based phenotyping processes
- Rules from the National Quality Forum (NQF) Quality Data Model (QDM) files can be uploaded to the Phenotype Portal

MAYO CLINIC

UIs created for CTS2 Implementations
High-Throughput Phenotyping (HTP) Overview

- Convert structured phenotype criteria into executable queries using JBoss Drools Rules
- Runs against Clinical Element Model (CEM) DB and EHRs to find cohorts that match the given criteria

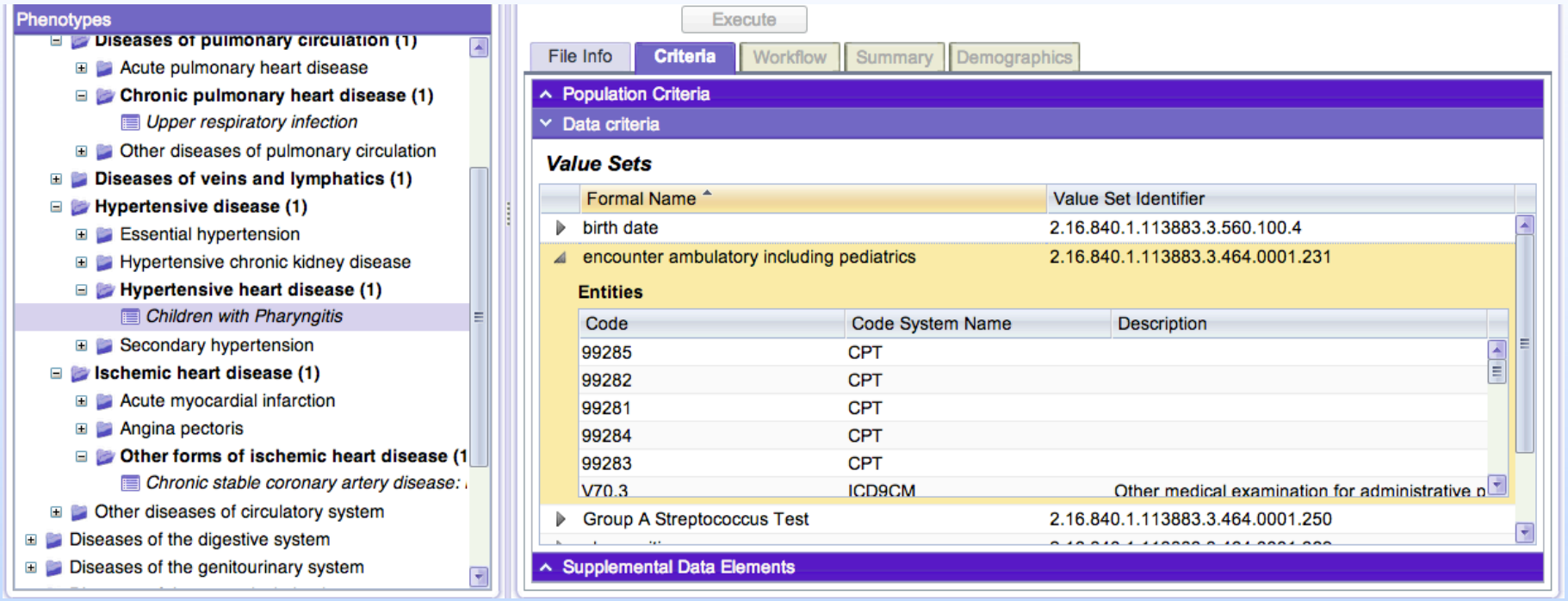
MAYO CLINIC

UIs created for CTS2 Implementations
High-Throughput Phenotyping (HTP) Overview

- Users can currently view Value Sets and their entities via the CTS2 service
- Planned availability to add/remove entries from Value Sets and save them. This will allow the user to execute the phenotypes with modified criteria to see how it affects the results

MAYO CLINIC

UIs created for CTS2 Implementations
High-Throughput Phenotyping (HTP)



CTS2 Enabled UI Widgets

MAYO CLINIC

Displaying CTS2 Content

- We have developed several customizable widgets that can be added to a web page to present CTS2 content
- Open source and available in GitHub:
  - https://github.com/cts2/

MAYO CLINIC

CTS2 Enabled UI Widgets
Overview

- Tooltips
  - jQuery tooltip widget used to display value sets and code systems when you mouse over text
- Autocomplete
  - jQuery autocomplete widget used to suggest value sets as you type
- Dropdown
  - jQuery example to fill a select (dropdown) with specific resolved value sets

MAYO CLINIC

CTS2 Enabled UI Widgets
Demonstration

- http://informatics.mayo.edu/cts2widgets/widgets.html
- Features and customization points of each widget:
  - Ability to set the value set ID/code system ID to retrieve different values
  - Cascading Style Sheets (CSS) to change the look and feel
  - CTS2 parameterized values to customize the query

MAYO CLINIC

UI Widget: Tooltip

MAYO CLINIC

CTS2 Enabled UI Widgets
Lab Exercise - Tooltip

- Retrieve a list of available value sets:
  - http://informatics.mayo.edu/cts2/rest/valuesets
- Select any value set – **resourceName**
  - Example: **skos**

MAYO CLINIC

CTS2 Enabled UI Widgets
Lab Exercise - Tooltip

- Look at the result of value set call:
  - http://informatics.mayo.edu/cts2/rest/valueset/skos
- We will use the data returned here to populate our CTS2 tooltip widget

MAYO CLINIC

CTS2 Enabled UI Widgets
Lab Exercise – Tooltip in JSFiddle

- We will use JSFiddle to work with the web related files
  - JSFiddle allows us to see all the files (CSS, JavaScript, and HTML) in one page and edit them
  - Changes are seen immediately
  - http://jsfiddle.net/coryendle/n9bWF/

CTS2 Enabled UI Widgets
Lab Exercise – Tooltip: Add new Tooltip

- Create a new tooltip for your value set ID
  - Add the HTML to display the tooltip with your value set ID

```
<p><a href="#" class="hint_vs" data-id="skos">SKOS</a> Value Set Demo.</p>
```

  - Select the “Run” button to test your changes.

CTS2 Enabled UI Widgets
Lab Exercise – Tooltip: Change Attributes

- Update the Tooltip fadeIn/fadeOut speed
  - “slow”
  - “fast”
  - Milliseconds
- Find the following JavaScript function:
  - *function showTooltipValueSets()*
  - Replace the *fadeIn("slow")* to *fadeIn("slow")* This will make the tooltip appear over a 2 second duration

MAYO CLINIC

CTS2 Enabled UI Widgets
Lab Exercise – Tooltip: Add a row to the tooltip

- Update the tooltip to display another row of data. We will add the entryState status:
- Find the following function:
  - *function getValueSetTableInfo(valueSetJson)*
  - Add another variable:

```
var entryState =
valueSetJson.valueSetCatalogEntryMsg.valueSetC
atalogEntry.entryState;
```

MAYO CLINIC

CTS2 Enabled UI Widgets
Lab Exercise – Tooltip: Add a row to the tooltip

- Add another HTML row:

```
"<tr><td class=\"noWrap\">Entry State:</
td><td>" + entryState + "</td></tr>" +
```

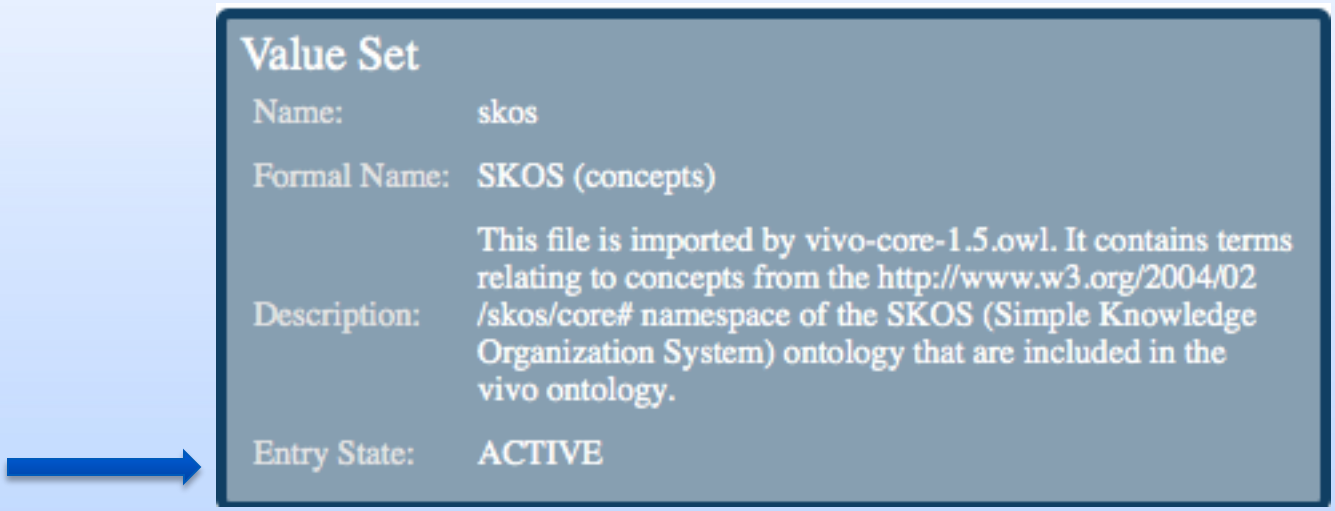
- Select the “Run” button to test your changes

MAYO CLINIC

CTS2 Enabled UI Widgets

Lab Exercise – Tooltip: Add a row to the tooltip

- Now when you mouse over a value set ID, you will see an additional row called “Entry State” in the tooltip



MAYO CLINIC

CTS2 Enabled UI Widgets

Lab Exercise – Tooltip: Change the Background Color

- We will update the CSS to change the background color
- In the CSS file, find the following section:

  div#tooltip

  - Replace this line:

    background-color: #879fb1;

  - With this:

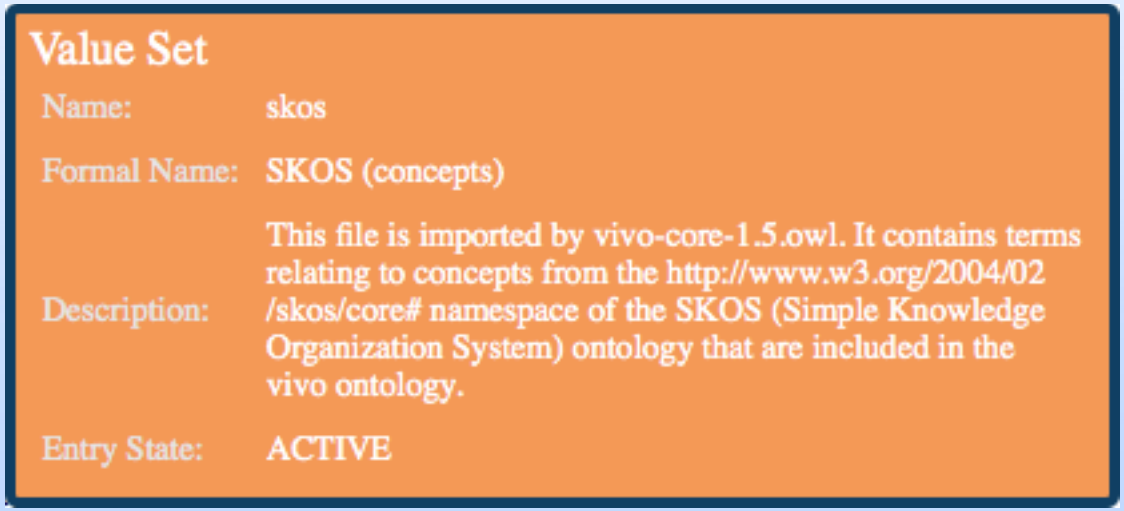
    background-color: #f49956;

MAYO CLINIC

CTS2 Enabled UI Widgets

Lab Exercise – Tooltip: Change the Background Color

- Select the “Run” button to test your changes
- Now when you mouse over a value set ID, you will see the background of the Tooltip has changed



MAYO CLINIC

UI Widget: Dropdown

MAYO CLINIC

CTS2 Enabled UI Widgets
Lab Exercise – Dropdown: Country of Birth

- We will create a dropdown to select a person's country of birth
- Retrieve a list of available value sets from CDC PHINVADS CTS2 service:
  - http://informatics.mayo.edu/cts2/services/phinvads/valuesets?maxtoreturn=100
- Select the value set where:
  - valueSetName="PHVS_BirthCountry_CDC"

MAYO CLINIC

CTS2 Enabled UI Widgets
Lab Exercise – Dropdown: Country of Birth

- Look at the result of resolved value set call:
  - http://informatics.mayo.edu/cts2/services/phinvads/valueset/PHVS_BirthCountry_CDC/resolution
- We will use the data returned here to populate our CTS2 dropdown widget

MAYO CLINIC

CTS2 Enabled UI Widgets
Lab Exercise – Dropdown: Country of Birth

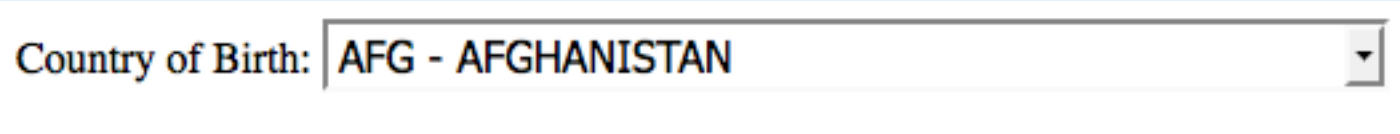
- JSFiddle code:
  - http://jsfiddle.net/coryendle/bhTaD/
- In the HTML code, we will add another dropdown to select the country of birth
- After the code for the “Severity” dropdown, add the following line of code:

```
<br/><br/> Country of Birth: <select class="cts2-valueset(name : PHVS_BirthCountry_CDC;max:500)"></select>
```

MAYO CLINIC

CTS2 Enabled UI Widgets
Lab Exercise – Dropdown: Country of Birth

- Select the “Run” button to test your changes
- You will now see the Country of Birth dropdown:



Summary

MAYO CLINIC

UI and Widget Review

- Examples of Web Applications that utilized CTS2 Service
- Customizable UI widgets available to quickly see the CTS2 content
- Open Source

MAYO CLINIC

References

- Sample REST calls for Module Specifications:
  - http://www.bioontology.org/wiki/index.php/CTS2_BioPortal_wrapper_summary
- Sample REST calls for Module Specifications against an RDF triple store:
  - http://www.bioontology.org/wiki/index.php/CTS2_RDF_Plugin
- REST interfaces for CTS2:
  - http://informatics.mayo.edu/cts2/index.php/REST
- Service Plugin resources:
  - https://github.com/hsbauer/cts2-example-service/archive/master.zip
  - http://informatics.mayo.edu/cts2/framework/downloads/cts2framework-standalone.jar
  - http://www.mysql.com/downloads/mysql/5.1.html
  - http://maven.apache.org/download.html (Maven 2.2.1)
  - http://www.eclipse.org/downloads/

MAYO CLINIC

References

- BioPortal REST Wrapper
  - http://informatics.mayo.edu/cts2/rest/valuesets
- BioPortal RDF Wrapper
  - http://informatics.mayo.edu/cts2/services/bioportal-rdf/valuesets
- BioPortal SPARQL endpoint:
  - http://sparql.bioontology.org/
- MU Quality Measure Value Set Service
  - https://informatics.mayo.edu/cts2/services/mat/valuesets
- GUIs that display CTS2 Content:
  - https://informatics.mayo.edu/vsmc
  - http://www.phenotypeportal.org/
- UI Widgets Example:
  - http://informatics.mayo.edu/cts2widgets/widgets.html

MAYO CLINIC

References

- UI Widgets in GitHub:
  - https://github.com/cts2/
  - https://github.com/cts2/cts2autocomplete
  - https://github.com/cts2/cts2tooltip
  - https://github.com/cts2/cts2dropdown
- jQuery UI Widget Reference:
  - http://jqueryui.com/
- JSFiddle UI Widgets Lab Exercises:
  - Tooltip: http://jsfiddle.net/coryendle/n9bWF/
  - Autocomplete: http://jsfiddle.net/coryendle/dKjXs/
  - Dropdown: http://jsfiddle.net/coryendle/bhTaD/

MAYO CLINIC

MAYO CLINIC

Questions & Discussion