CTS2 Development Framework Implementation

Technical Overview

Scott Bauer, Cory Endle

HL7 Working Group Meeting
May 9th, 2013

Pre-tutorial House Keeping

- 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
- Instructions are in the requirements document here: https://github.com/hsbauer/cts2-example-service/archive/master.zip

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

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 examples
- Command line Maven, Python and Unix/ Linux may be helpful as well

Overview

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

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

Terminology Services

- Awareness of 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 entities in service
  - For dealing with updates, versioning, history and other administrative functions

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

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

Modular Functionality
Implement / use what you need

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

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

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

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

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, …

CTS2 IMPLEMENTATION
PLANNING AN IMPLEMENTATION

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?

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.

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?

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

Tutorial Overview
Where to Start

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

CodeSystemCatalogEntry, Entity Description, Value Sets, Associations

- And use the same model elements to access code system entities, value sets, versions of code systems, and the relations tying entities together
- Create a service plugin based on the CodeSystemCatalog model element
- Note that these are a small part of the API

CTS2 REST CLIENT API

CTS2 REST Client API

CTS2 design is based on REST architecture:

- This shapes method calls and naming conventions
  - A directory groups summarized objects, such as code systems or entities represented as a catalog of these items
  - Read services generally are looking for a unique identifier for input
  - Resource links further define many of the returned values

Interactive Demo: CTS2 REST Client API

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

CTS2 Rest Client API

There is a pattern for building some basic REST calls in CTS2

- <Base URL>/ + codesystems
- <Base URL>/ + codesystemversions
- <Base URL>/ + valuesets

Returns a catalog contained in a directory of entries that have individual summaries for each element

CTS2 Rest Client API

With some preparation to the base URL, similar directories of more granular elements in the terminology service can be retrieved.

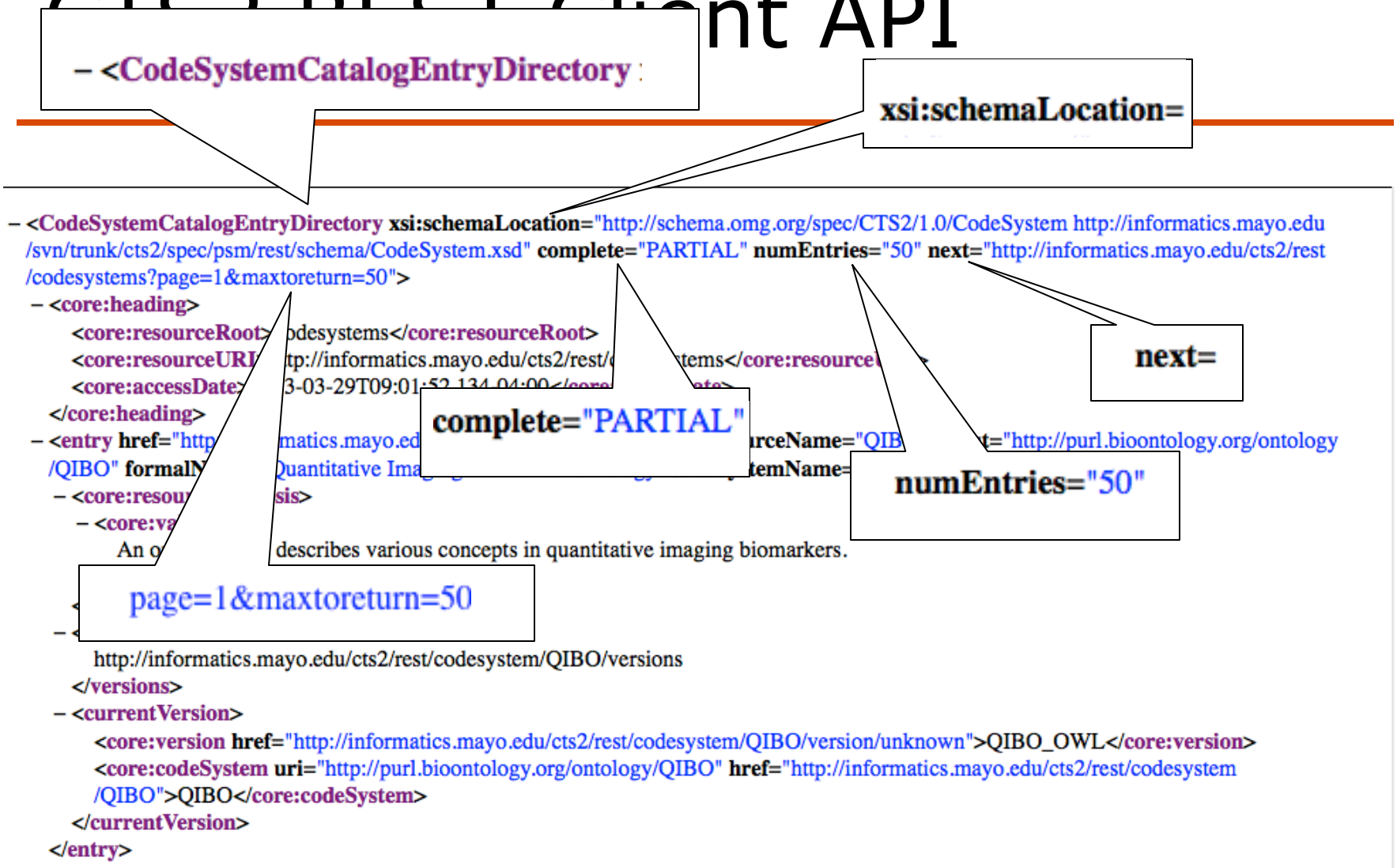
- <Base URL>/ + codesystem/ + <code system name>/ + version/ + <version>/ + entities

CTS2 REST Client API

How do we know what code 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

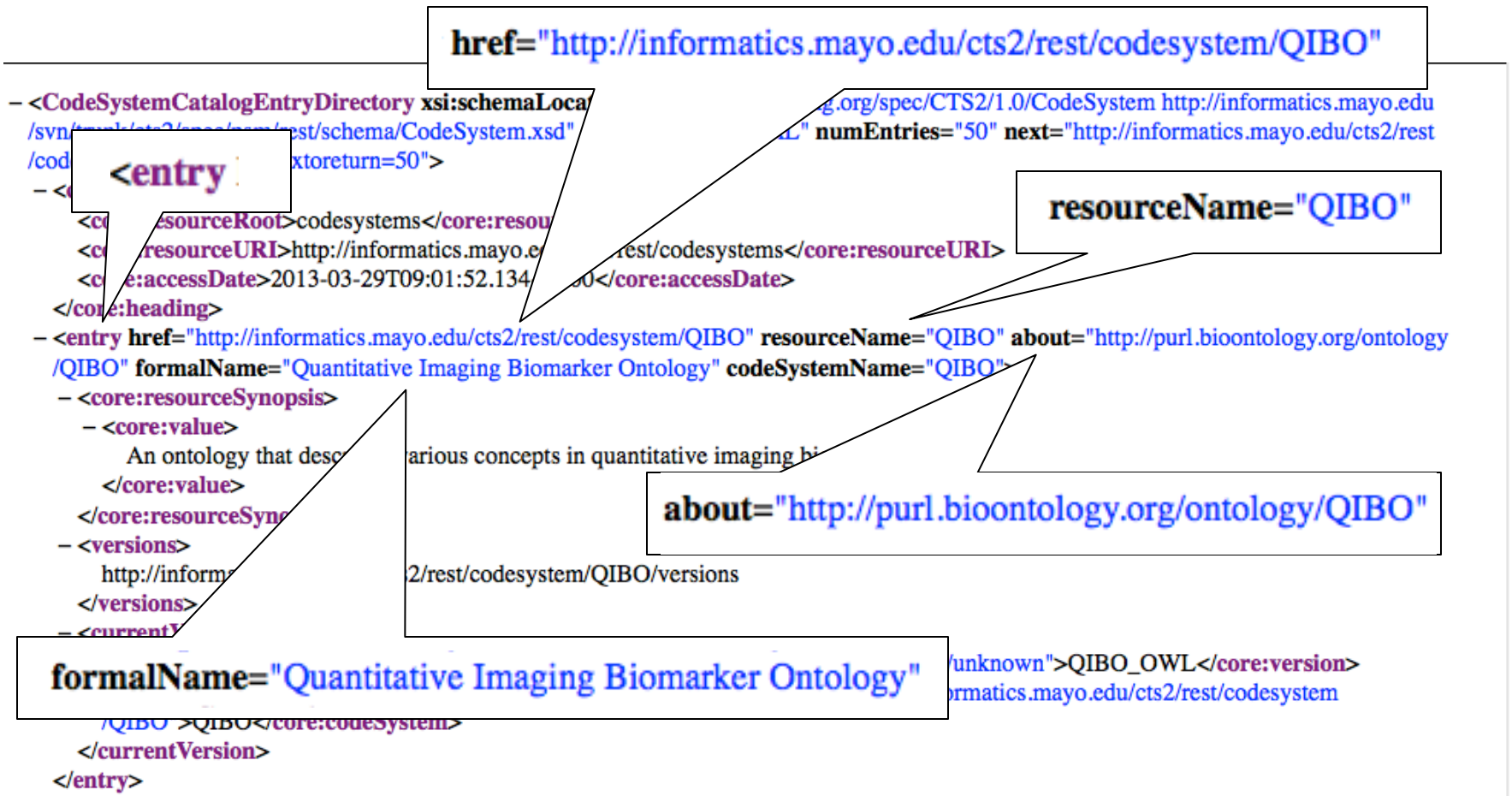
CTS2 REST Client API



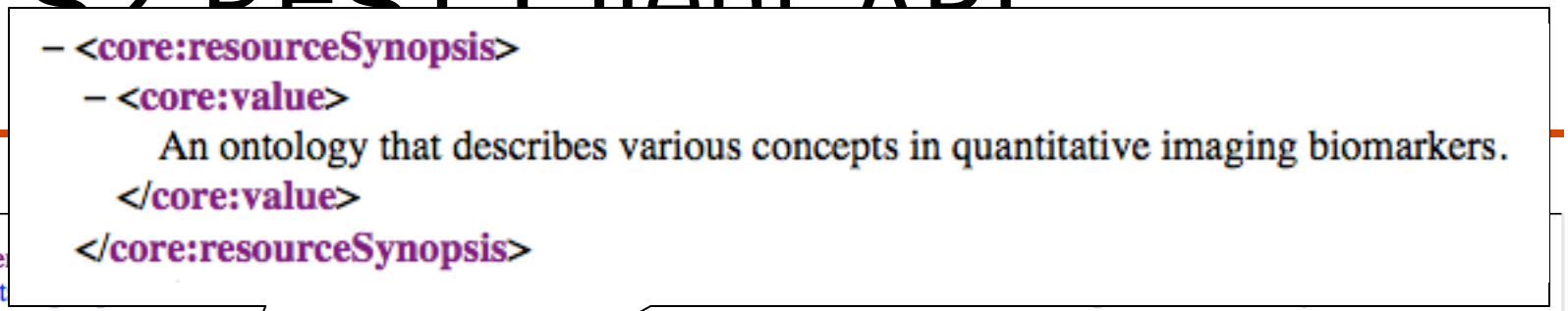
– <core:heading>
<core:resourceRoot>codesystems</core:resourceRoot>
<core:resourceURI>http://informatics.mayo.edu/cts2/rest/codesystems</core:resourceURI>
<core:accessDate>2013-03-29T09:01:52.134-04:00</core:accessDate>
</core:heading>

– <CodeSystemCata xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/CodeSystem http://informatics.mayo.edu/svn/trunk/cts2/sp chema/CodeSystem.xsd" complete="PARTIAL" numEntries="50" next="http://informatics.mayo.edu/cts2/rest/codesystems?pa maxtoreturn=50">
– <core:heading>
<core:resourceRoot>codesystems</core:resourceRoot>
<core:resourceURI>http://informatics.mayo.edu/cts2/rest/codesystems</core:resourceURI>
<core:accessDate>2013-03-29T09:01:52.134-04:00</core:accessDate>
</core:heading>
– <entry href="http://informatics.mayo.edu/cts2/rest/codesystem/QIBO" resourceName="QIBO" about="http://purl.bioontology.org/ontology/QIBO" formalName="Quantitative Imaging Biomarker Ontology" codeSystemName="QIBO">
– <core:resourceSynopsis>
– <core:value>
An ontology that describes various concepts in quantitative imaging biomarkers.
</core:value>
</core:resourceSynopsis>
– <versions>
http://informatics.mayo.edu/cts2/rest/codesystem/QIBO/versions
</versions>
– <currentVersion>
<core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/QIBO/version/unknown">QIBO_OWL</core:version>
<core:codeSystem uri="http://purl.bioontology.org/ontology/QIBO" href="http://informatics.mayo.edu/cts2/rest/codesystem/QIBO">QIBO</core:codeSystem>
</currentVersion>
</entry>

CTS2 REST Client API



CTS2 REST Client API



– <CodeSyste
/svn/trunk/ct
/codesystems?page=1&maxtoretur
– <core:heading>
<core:resourceRoot>code eRoot>
<core:resourceURI>http: .edu/cts2/rest/codesystems</core:resourceURI>
<core:accessDate>2013 52.134-04:00</core:accessDate>
</core:heading>
– <entry href="http://infor s.mayo.edu/cts2/rest/codesystem/QIBO" resourceName="QIBO" about="http://purl.bioontology.org/ontology
/QIBO" formalName="Quantitative Imaging Biomarker Ontology" codeSystemName="QIBO">
– <core:resourceSynopsis>
– <core:value>
An ontology that describes various concepts in quantitative imaging biomarkers.
</core:value>
</core:resourceSynopsis>
– <versions>
http://informatics.mayo.edu/cts2/rest/codesystem/QIBO/versions
</versions>
– <currentVersion>
<core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/QIBO/version/unknown">QIBO_OWL</core:version>
<core:codeSystem uri="http://purl.bioontology.org/ontology/QIBO" href="http://informatics.mayo.edu/cts2/rest/codesystem
/QIBO">QIBO</core:codeSystem>
</currentVersion>
</entry>

CTS2 REST Client API

```
<versions>
  http://informatics.mayo.edu/cts2/rest/codesystem/QIBO/versions
</versions>
<currentVersion>
  <core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/QIBO/version/unknown">QIBO_OWL</core:version>
  <core:codeSystem uri="http://purl.bioontology.org/ontology/QIBO" href="http://informatics.mayo.edu/cts2/rest/codesystem
  /QIBO">QIBO</core:codeSystem>
</currentVersion>
```

```
    <core:resourc                                    ot>
    <core:resour                                     cts2/rest/codesystems</core:resourceURI>
    <core:access                                     04:00</core:accessDate>
  </core:headin
 – <entry href="                                     u/cts2/rest/codesystem/QIBO" resourceName="QIBO" about="http://purl.bioontology.org/ontology
  /QIBO" form                                        Imaging Biomarker Ontology" codeSystemName="QIBO">
   – <core:res
     – <core:
         An                                          describes various concepts in quantitative imaging biomarkers.
       </co
     </core                                          ceSynopsis>
   – <vers                                           >
       http://informatics.mayo.edu/cts2/rest/codesystem/QIBO/versions
     </versions>
   – <currentVersion>
       <core:version href="http://informatics.mayo.edu/cts2/rest/codesystem/QIBO/version/unknown">QIBO_OWL</core:version>
       <core:codeSystem uri="http://purl.bioontology.org/ontology/QIBO" href="http://informatics.mayo.edu/cts2/rest/codesystem
       /QIBO">QIBO</core:codeSystem>
     </currentVersion>
  </entry>
```

CTS2 REST Client API

Review:

- There is a similar structure for value sets and code system versions returned for CTS2 REST calls
- You can reconstruct your last call from the heading
- Depending on the Service, some XML tag content and attributes can be used to filter queries

CTS2 REST Client API

Review continued:

- URLs exist that link other resources
- Parsed values and links can be used to determine the number of values on the page and whether you can page for another set of values
- Links and values exist to either help you build a REST call or provide you with a link to further define the resource

CTS2 REST Client API

What does this mean?

- Users can both “Read” and “Query” a code system or any other entry that is cataloged in a directory
- The listing of entries can each be read by their “resourceName” or queried by their global identifier or “URI” using a REST call

CTS2 REST Client API

And that queries can be done using filters:

- By any attribute or tag supported as a filter in the service. This may include a terminology name, ontology Id, or description
- And may use matching algorithms such as “contains” or “exact match”

CTS2 REST Client API

Building the REST:

<Base URL>/codesystem/<resourceName>

<Base URL>/codesystembyuri?<uri>

<Base URL>/codesystems? +

matchvalue=<value>& +

filtercomponent=<element to filter>

…. more

CTS2 REST CLIENT API

<Base URL>/codesystems?page=<page no>

<Base URL>/codesystems? +

page=<page no>& +

maxtoreturn=<number of entries>

CTS2 REST Client API

Walking through the set of entries in a catalog a user can cache all resourceNames and read with certainty the terminology associated with that resource name. Where MedDRA is MDR we call:

http://informatics.mayo.edu/cts2/rest/codesystem/MDR

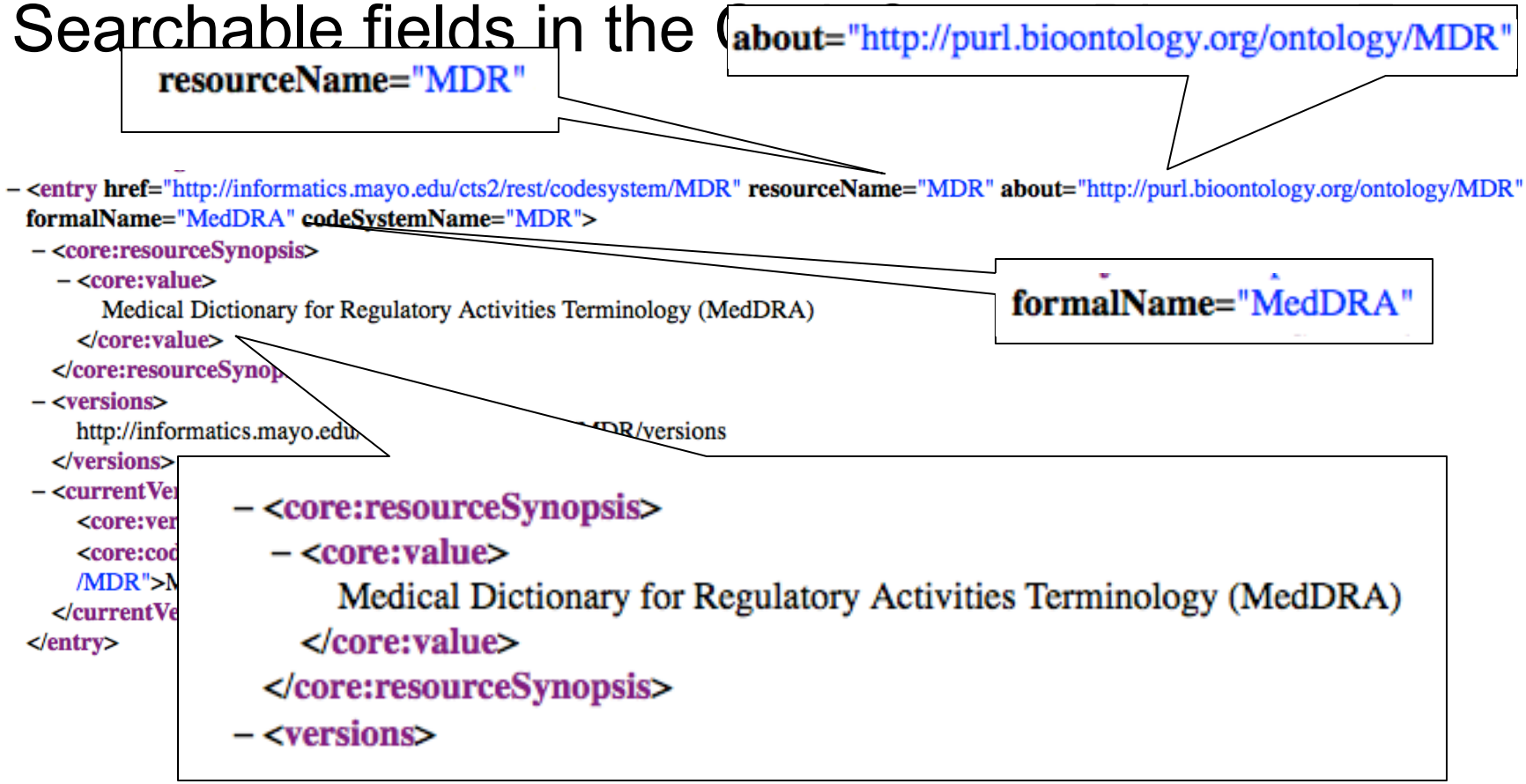
CTS2 REST Client API

Similarly a name is matched against the abbreviated ontology designation in this query of the code system

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

CTS2 REST CLIENT API

Searchable fields in the C



CTS2 REST Client API

Review:

<Base URL>/codesystems? +

matchvalue=<value to match>& +

filtercomponent=<element to filter>

CTS2 REST Client API

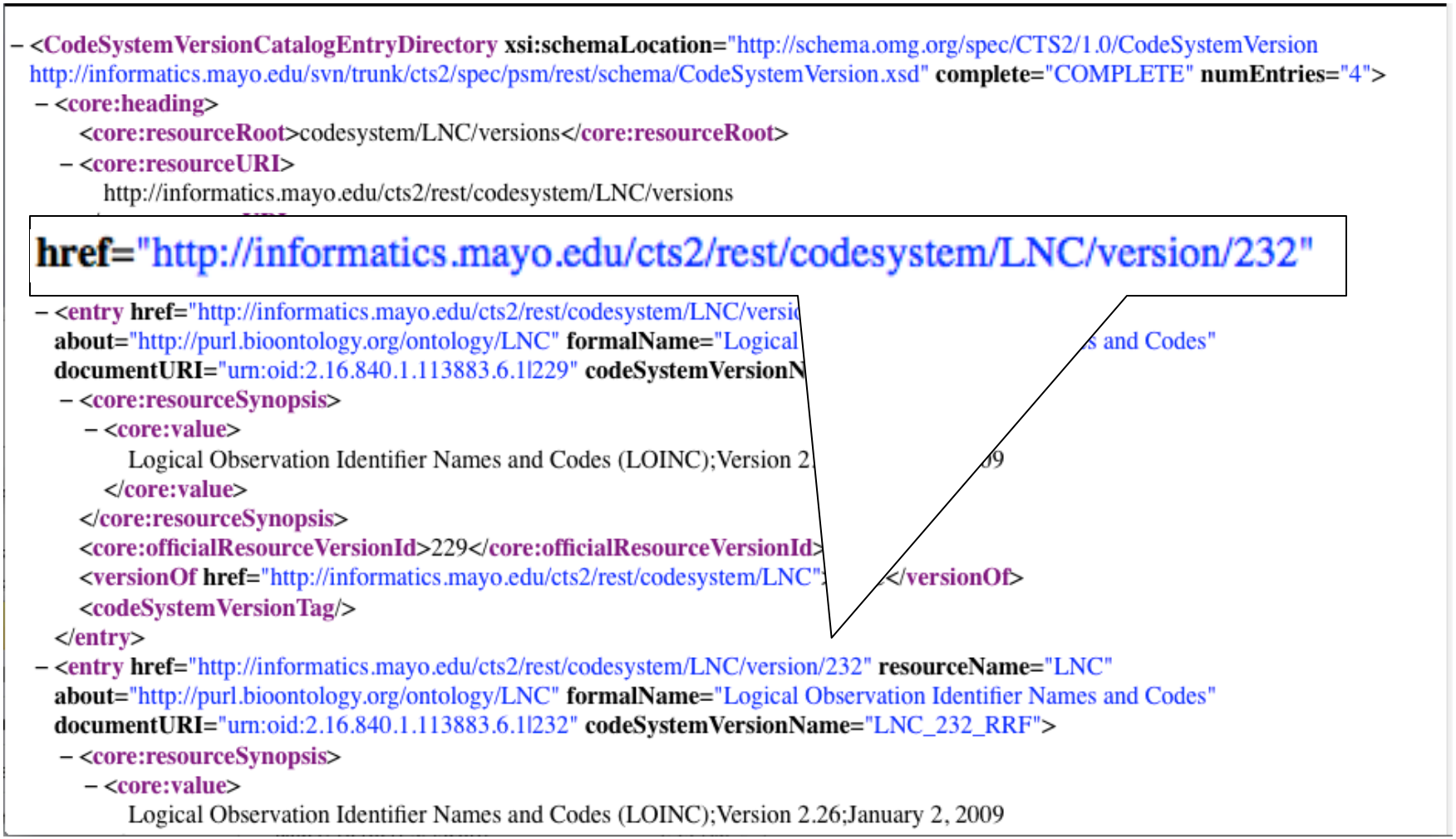
A codesystemversions query has a similar query interface for getting all the versions of all the code systems on a given service

- 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

CTS2 REST Client API



CTS2 REST Client API

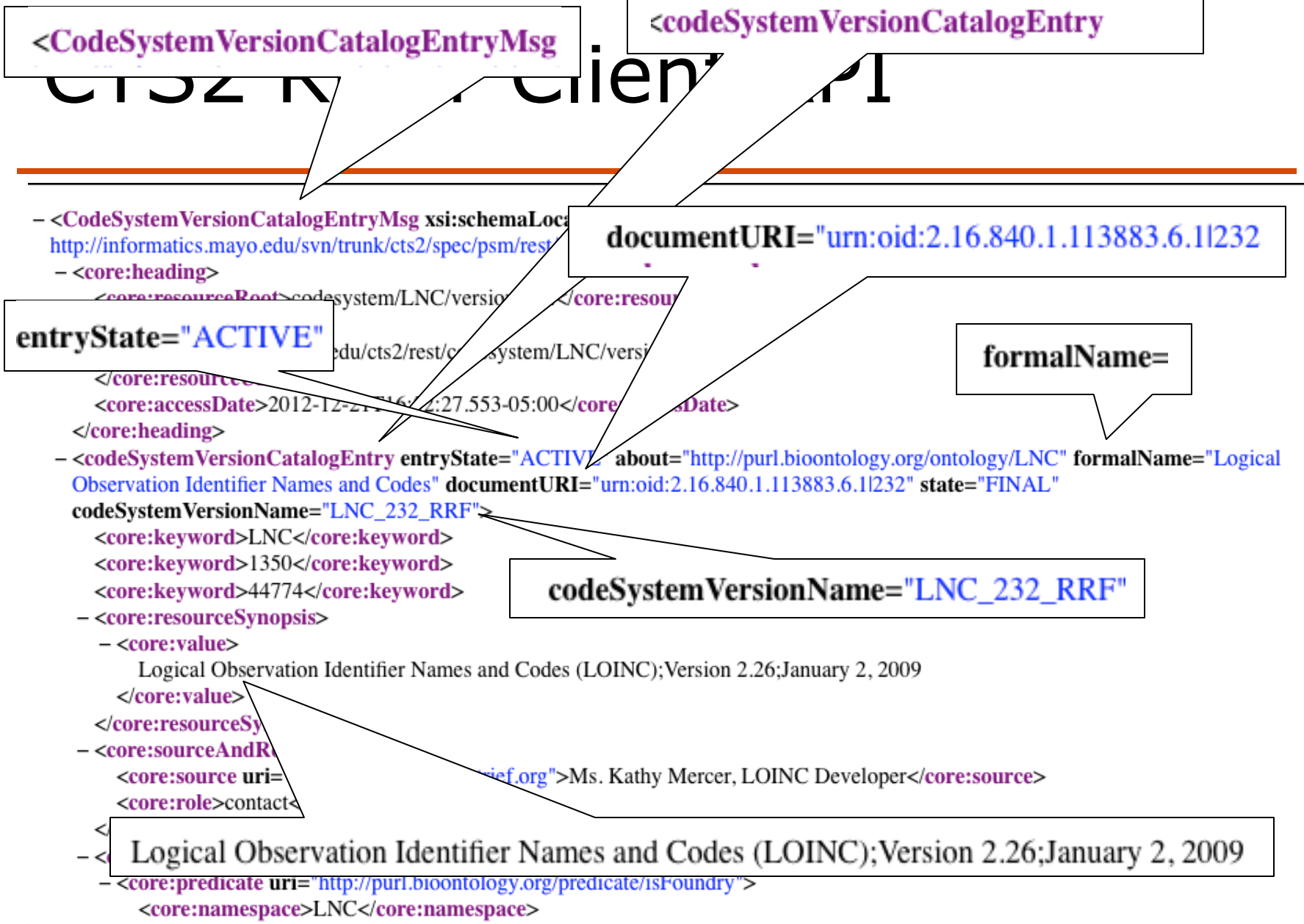
Review:

- Retrieve the smaller set of versions
  - <Base URL>/codesystem/<code system id>/versions
- Use the href of the version designation to return the desired version

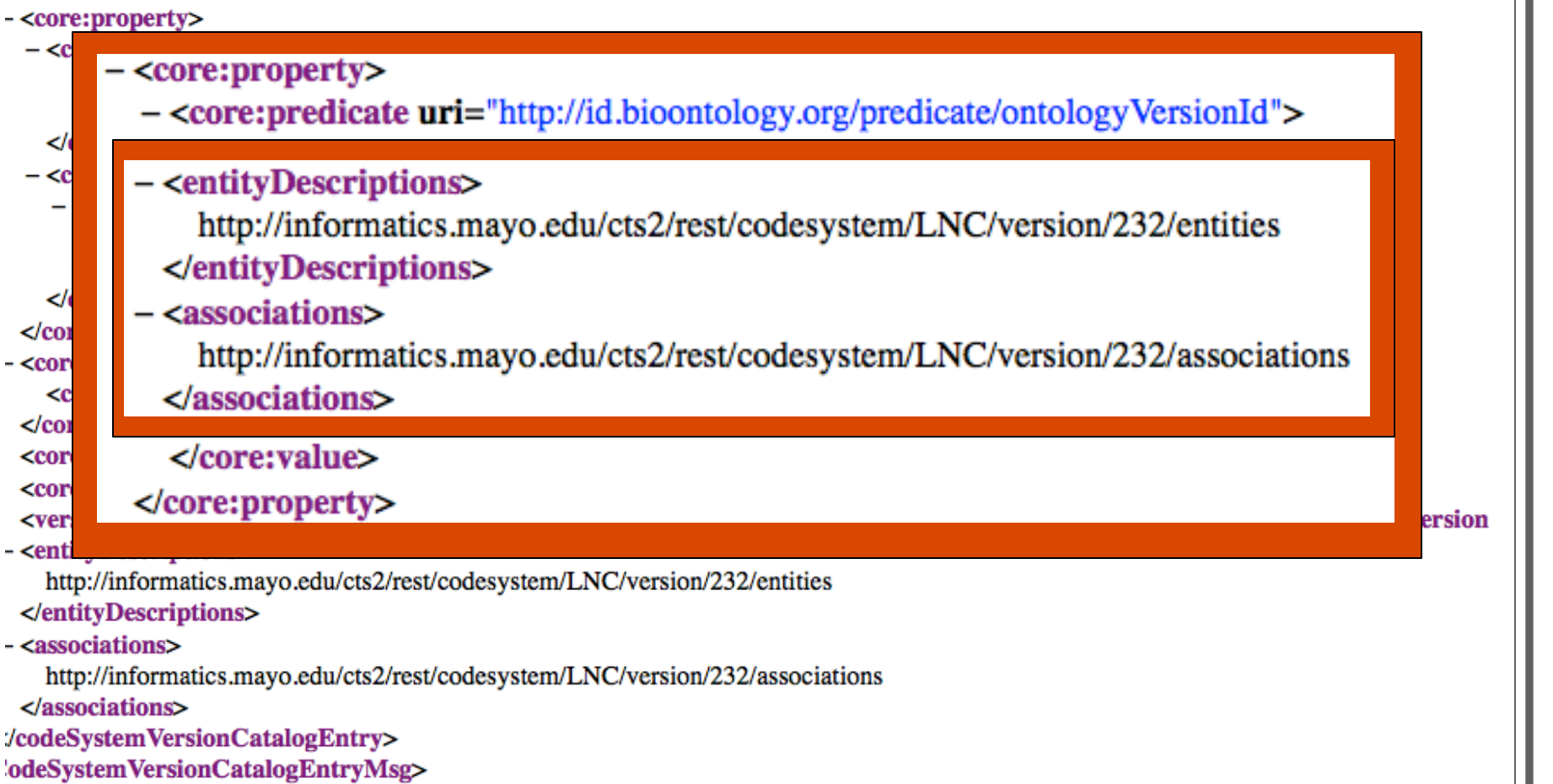
CTS2 REST Client API

- The href happens to be the same REST call otherwise used for the Code System Version Read function:
- http://informatics.mayo.edu/cts2/rest/codesystem/LNC/version/232
- This is a fairly consistent pattern. Entries contain URLs to another resource and in this case that resource is what is also returned by a similar read call

CTS2 REST Client API



CTS2 REST Client API



CTS2 REST Client API

Review -- Retrieve versions of code Systems:

<Base URL>/codesystemversions

<Base URL>/codesystem/<code system id>/versions

<Base URL>/codesystem/<code system id>/version/<version id>

CTS2 REST Client API

Most users may just 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

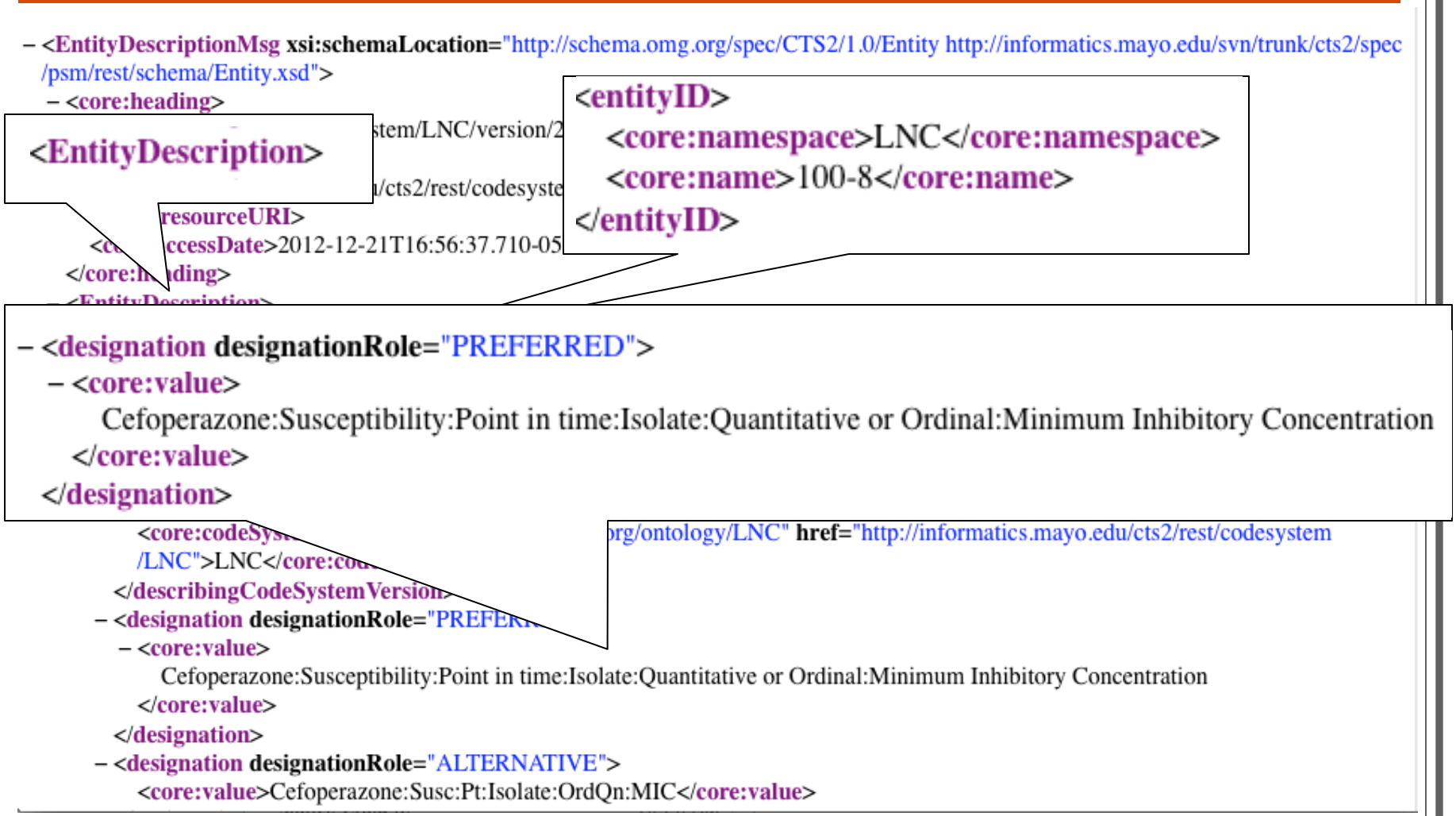
CTS2 REST Client API

CTS2 REST Client 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

CTS2 REST Client API



CTS2 REST Client API

Review:

<Base URL>/entities

<Base URL and code system version>/ +

entities?matchvalue=<text to match>

<Base URL and code system version>/ +

entity/<entity name tag contents>

CTS2 REST Client API

Review:

<Base URL>/entities

<Base URL and code system version>/ +

entities?matchvalue=<text to match>& +

matchalgorithm=<match algorithm name>

CTS2 REST Client API

This lays the foundation for a likely use case. Starting from the code systems:

- Display available code systems to user
  - User chooses code system(s)
- Display versions to user
  - User chooses code system version
- Display text match for concept search
  - User initiates text search and makes choice from matches (Final result is a readable entity display)

CTS2 REST Client API

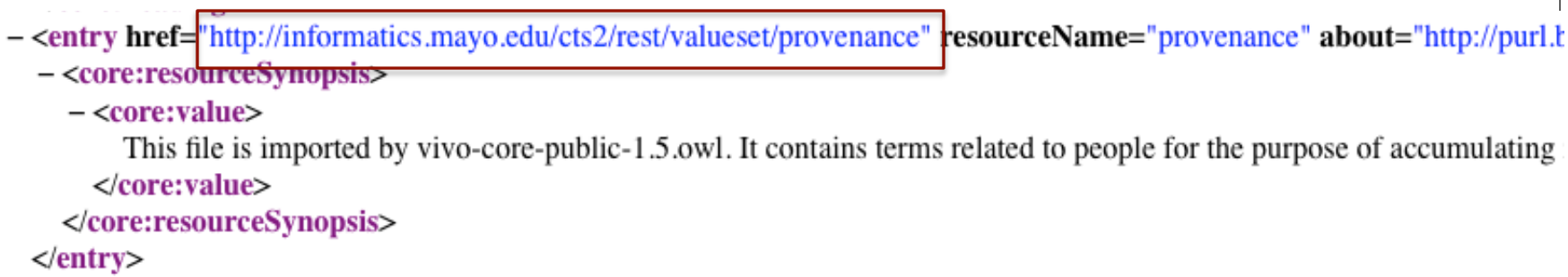
Alternatively:

- 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

CTS2 REST Client 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



- <Base URL>/valuesets
- <Base URL>/valueset/<resourceName>

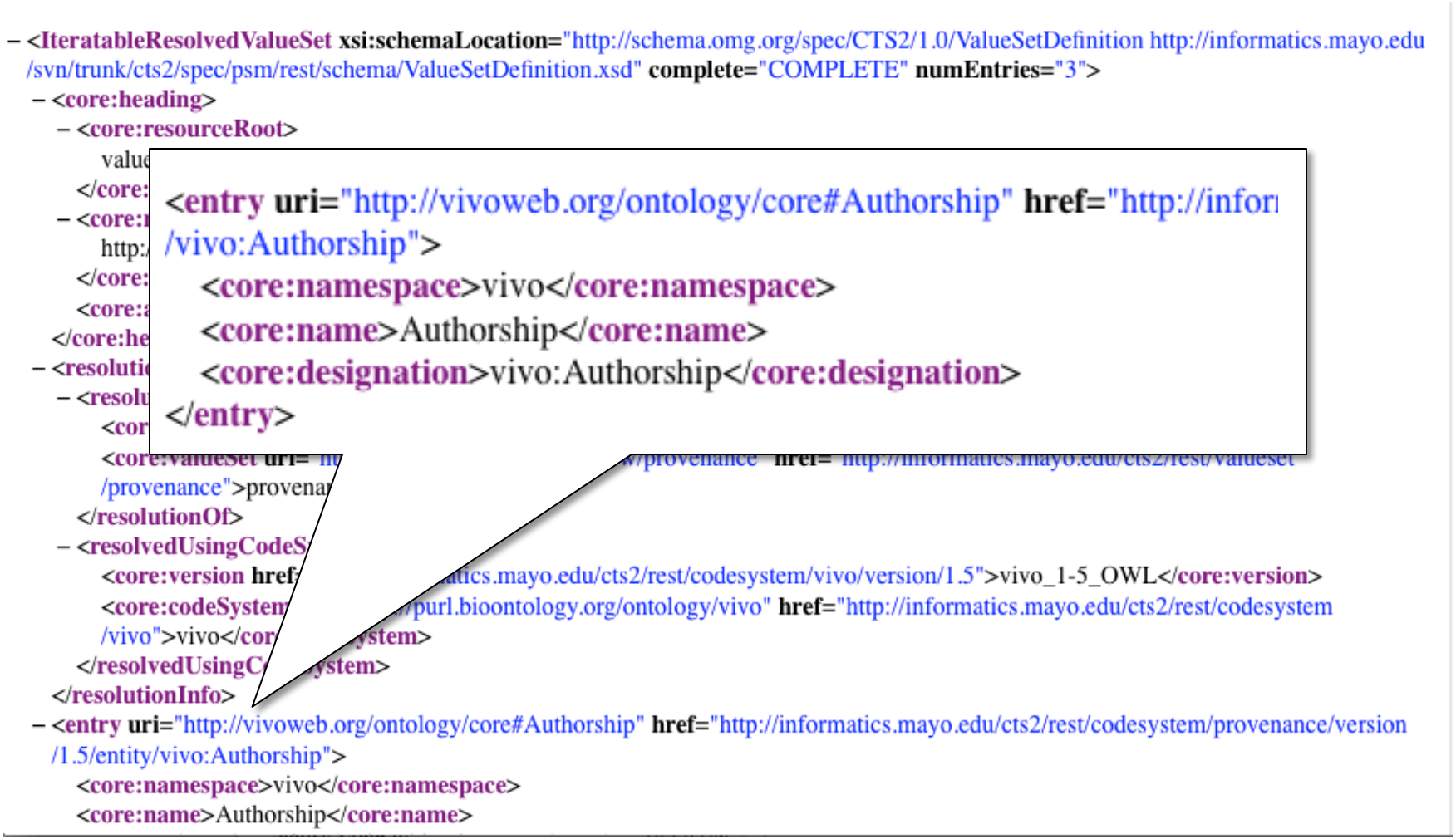
CTS2 REST Client API
Value Sets

Resolving a value set (adding the resolution parameter):

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

<Base URL>/valueset/<resourceName>/resolution

CTS2 REST Client API
Value Sets



CTS2 REST Client 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.

<Base URL>/resolvedvaluesets

CTS2 REST Client API
Value Sets

Review:

<Base URL>/valuesets

<Base URL>/valueset/<resourceName>

<Base URL>/resolvedvaluesets

CTS2 REST Client API
Associations

For many terminologies you don't want to page through the entire set of associations but you can if you want:

http://informatics.mayo.edu/cts2/services/bioportal-rdf/codesystem/ICD9CM/version/ICD9CM-47178/associations

Normally graph traversal starts at some focus node and progresses from there.

CTS2 REST Client API
Associations

Traversing from a focus node can start here:

http://informatics.mayo.edu/cts2/services/bioportal-rdf/codesystem/ICD9CM/version/ICD9CM-47178/entity/E008/subjectof

Where we find all the triples where entity "E008" is the subject

CTS2 REST Client API
Associations


```
– <AssociationDirectory xsi:schemaLocation="http://schema.omg.org/spec/CTS2/1.0/Association http://informatics.mayo.edu/svn/trunk/cts2/spec/psm/rest
  /schema/Association.xsd" complete="COMPLETE" numEntries="23">
  – <core:heading>
    – <core:resourceRoot>
        codesystem/ICD9CM/version/ICD9CM-47178/entity/E008/subjectof
      </core:resourceRoot>
    – <core:resourceURI>
        http://informatics.mayo.edu/cts2/services/bioportal-rdf/codesystem/ICD9CM/version/ICD9CM-47178/entity/E008/subjectof
      </core:resourceURI>
      <core:accessDate>2013-04-02T14:03:02.233-04:00</core:accessDate>
    </core:heading>
  – <entry>
    – <subject uri="http://purl.bioontology.org/ontology/ICD9CM/E008" href="http://informatics.mayo.edu/cts2/services/bioportal-rdf/codesystem/ICD9CM
      /version/ICD9CM-47178/entity/E008">
        <core:namespace>ICD9CM</core:namespace>
        <core:name>E008</core:name>
      </subject>
    – <predicate uri="http://www.w3.org/1999/02/22-rdf-syntax-ns#type" href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">
        <core:namespace>rdfs1999</core:namespace>
        <core:name>type</core:name>
      </predicate>
    – <target>
      – <core:entity uri="http://www.w3.org/2002/07/owl#Class" href="http://informatics.mayo.edu/cts2/services/bioportal-rdf/codesystem/ICD9CM
        /version/ICD9CM-47178/entity/Class">
          <core:namespace>owl</core:namespace>
          <core:name>Class</core:name>
        </core:entity>
```

CTS2 REST Client API

```
– <subject uri="http://purl.bioontology.org/ontology/ICD9CM/E008" href=
  /version/ICD9CM-47178/entity/E008">
     <core:namespace>ICD9CM</core:namespace>
     <core:name>E008</core:name>
  </subject>
– <entry>
<predicate uri="http://www.w3.org/2000/01/rdf-schema#subClassOf" href=
   <core:namespace>rdfs</core:namespace>
   <core:name>subClassOf</core:name>
</predicate>
   </predicate>
<target>
 – <core:entity uri="http://purl.bioontology.org/ontology/ICD9CM/E001-E030.9" href="
   rdf/codesystem/ICD9CM/version/ICD9CM-47178/entity/E001-E030.9">
      <core:namespace>ICD9CM</core:namespace>
      <core:name>E001-E030.9</core:name>
   </core:entity>
</target>
</entry>
```

CTS2 REST Client API
Associations

Similarly we can find all the triples where entity "E008" is the target

http://informatics.mayo.edu/cts2/services/bioportal-rdf/codesystem/ICD9CM/version/ICD9CM-47178/entity/E008/targetof

CTS2 REST Client API
Associations

Using a designation of children or parents we can merely list the entities which have that designation without any of the metadata describing the relationship

http://informatics.mayo.edu/cts2/services/bioportal-rdf/codesystem/ICD9CM/version/ICD9CM-47178/entity/E008/children

CTS2 REST Client API
Associations

knownEntityDescription href=

<core:designation>Frisbee</core:designation>

```
– <entry about="http://purl.bioontolo                    tology/ICD9CM/E008.3" href="http://informatics.mayo.edu/cts2/services/bioportal-rdf/entity
/ICD9CM:E008.3" resourceName=                    .3">
  – <core:name>
      <core:namespace>ICD9CM                    ore:namespace>
      <core:name>E008.3</core                    ame>
    </core:name>
  – <core:knownEntityDescription href="http://informatics.mayo.edu/cts2/services/bioportal-rdf/codesystem/ICD9CM/version/ICD9CM-47178/entity
    /ICD9CM:E008.3">
    – <core:describingCodeSystemVers
        <core:version uri="http://biopor                    mayo.edu/cts2/services/bioportal-
        rdf/codesystem/ICD9CM/version
        <core:codeSystem uri="http://bioportal.bio                    ntologies/ICD9CM" href="http://informatics.mayo.edu/cts2/services/bioportal-
        rdf/codesystem/ICD9CM">ICD9CM</                    aSystem>
      </core:describingCodeSystemVersi                    n>
      <core:designation>Frisbee</core:designation>
    </core:knownEntityDescription>
</entry>
        /ICD9CM:E008.2">
    – <core:describingCodeSystemVersion>
        <core:version uri="http://bioportal.bioontology.org/ontologies/ICD9CM/version/47178" href="http://informatics.mayo.edu/cts2/services/bioportal-
        rdf/codesystem/ICD9CM/version/ICD9CM-47178">ICD9CM-47178</core:version>
        <core:codeSystem uri="http://bioportal.bioontology.org/ontologies/ICD9CM" href="http://informatics.mayo.edu/cts2/services/bioportal-
```

CTS2 REST Client API
Associations

- Ancestors and descendants of a given focus code provide some support for a transitive reasoning of what codes are related to each other a over a hierarchy of relations:

<Base URL code system version>/ +

entity/<entity id>/descendent

- Resulting xml looks the same as children/ parents REST call results – a list of entities

CTS2 REST Client API
Associations

- While relationships are described in subjectof and targetof calls, graph relationships can be shown by appending “graph” to any of the previous association type calls

<Base URL code system version>/ +

entity/<entity id>/subjectof/graph

CTS2 REST Client API
Associations

- Association Graph defines a flattened representation of the corresponding graph that is designed for manipulation and reassembly
- every entry that occurs in the set of associations is assigned a unique number. These numbers are assigned in depth first order and, when both forward and reverse association traversal is supported, in the order of forward first followed by reverse

CTS2 REST Client API

Associations

SPECIFIC_ENTITY
TOP_NODE
BOTTOM_NODE

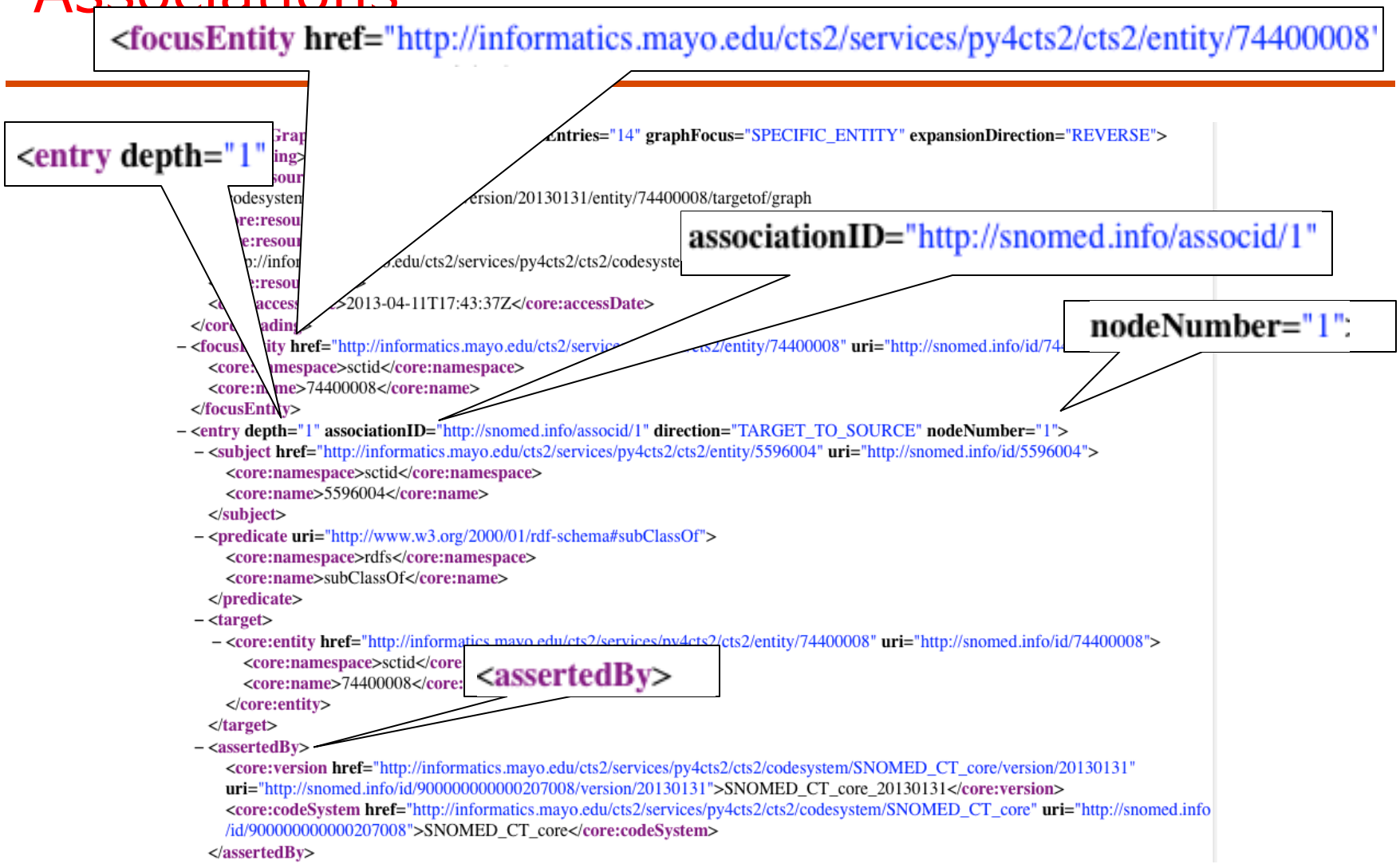
REVERSE
FORWARD
BOTH

TARGET_TO_SOURCE
SOURCE_TO_TARGET

```
<AssociationGraph complete="COMPLETE" numEntries="14" graphFocus="SPECIFIC_ENTITY" expansionDirection="REVERSE">
 - <core:heading>
   - <core:resourceRoot>
       codesystem/SNOMED_CT_core/version/20130131/entity/74400008/targetof/graph
     </core:resourceRoot>
   - <core:resourceURI>
       http://informatics.mayo.edu/cts2/services/py4cts2/cts2/codesystem/SNOMED_CT_core/v
     </core:resourceURI>
     <core:accessDate>2013-04-11T17:43:37Z</core:accessDate>
  </core:heading>
 - <focusEntity href="http://informatics.mayo.edu/cts2/services/py4cts2/cts2/entity/74400008" uri="http://snomed.info/id/74400008">
     <core:namespace>sctid</core:namespace>
     <core:name>74400008</core:name>
  </focusEntity>
 - <entry depth="1" associationID="http://snomed.info/associd/1" direction="TARGET_TO_SOURCE" nodeNumber="1">
   - <subject href="http://informatics.mayo.edu/cts2/services/py4cts2/cts2/entity/5596004" uri="http://snomed.info/id/5596004">
       <core:namespace>sctid</core:namespace>
       <core:name>5596004</core:name>
     </subject>
   - <predicate uri="http://www.w3.org/2000/01/rdf-schema#subClassOf">
       <core:namespace>rdfs</core:namespace>
       <core:name>subClassOf</core:name>
     </predicate>
   - <target>
     - <core:entity href="http://informatics.mayo.edu/cts2/services/py4cts2/cts2/entity/74400008" uri="http://snomed.info/id/74400008">
         <core:namespace>sctid</core:namespace>
         <core:name>74400008</core:name>
       </core:entity>
     </target>
   - <assertedBy>
       <core:version href="http://informatics.mayo.edu/cts2/services/py4cts2/cts2/codesystem/SNOMED_CT_core/version/20130131"
       uri="http://snomed.info/id/900000000000207008/version/20130131">SNOMED_CT_core_20130131</core:version>
       <core:codeSystem href="http://informatics.mayo.edu/cts2/services/py4cts2/cts2/codesystem/SNOMED_CT_core" uri="http://snomed.info
       /id/900000000000207008">SNOMED_CT_core</core:codeSystem>
     </assertedBy>
```

CTS2 REST Client API

Associations




```
Entries="14" graphFocus="SPECIFIC_ENTITY" expansionDirection="REVERSE">
ersion/20130131/entity/74400008/targetof/graph
.edu/cts2/services/py4cts2/cts2/codesyste
>2013-04-11T17:43:37Z</core:accessDate>
ity href="http://informatics.mayo.edu/cts2/servic cts2/entity/74400008" uri="http://snomed.info/id/74
mespace>sctid</core:namespace>
me>74400008</core:name>
</focusEntity>
<entry depth="1" associationID="http://snomed.info/associd/1" direction="TARGET_TO_SOURCE" nodeNumber="1">
<subject href="http://informatics.mayo.edu/cts2/services/py4cts2/cts2/entity/5596004" uri="http://snomed.info/id/5596004">
<core:namespace>sctid</core:namespace>
<core:name>5596004</core:name>
</subject>
<predicate uri="http://www.w3.org/2000/01/rdf-schema#subClassOf">
<core:namespace>rdfs</core:namespace>
<core:name>subClassOf</core:name>
</predicate>
<target>
<core:entity href="http://informatics.mayo.edu/cts2/services/py4cts2/cts2/entity/74400008" uri="http://snomed.info/id/74400008">
<core:namespace>sctid</core
<core:name>74400008</core:
</core:entity>
</target>
<assertedBy>
<core:version href="http://informatics.mayo.edu/cts2/services/py4cts2/cts2/codesystem/SNOMED_CT_core/version/20130131"
uri="http://snomed.info/id/900000000000207008/version/20130131">SNOMED_CT_core_20130131</core:version>
<core:codeSystem href="http://informatics.mayo.edu/cts2/services/py4cts2/cts2/codesystem/SNOMED_CT_core" uri="http://snomed.info
/id/900000000000207008">SNOMED_CT_core</core:codeSystem>
</assertedBy>
```

CTS2 REST Client API
Associations

Review:

<Base URL and code system version>/ +

associations

<Base URL and codesystem version>/ +

entity/<entity code>/subjectof

<Base URL and codesystem version>/ +

entity/<entity code>/targetof

CTS2 REST Client API
Associations

Review:

<Base URL and codesystem version>/ +
entity/<entity code>/children

<Base URL and codesystem version>/ +
entity/<entity code>/descendants

<Base URL and codesystem version>/ +
entity/<entity code>/children/graph

.

CTS2 REST Client API
Associations

Scenario

- An end user wants to to find instances of a class of medications
- End user chooses a term in code system as a starting place.
- A set of associations are displayed
- User chooses an appropriate association and depth to resolve a hierarchy …

CTS2 REST Client API
Associations

Scenario continued

A hierarchy is displayed with a set of instances at the bottom.

Example:

Analgesics is the top node, leaves are displayed that are dosages of a brand name aspirin and other analgesics.

CTS2 REST Client API
Export Format

REST export formats:

- Where supported, a choice of formats can be applied to the results. Formats could include such things as OWL, pipe delimited text, or JSON

<Base URL>/codesystems&format=<format>

http://informatics.mayo.edu/cts2/services/py4cts2/cts2/codesystems/?format=json

CTS2 REST Client 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 either data and possibly href(s) pointing to new resource options.)

CTS2 REST Client API
Summary

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

CTS2 REST Client API
Summary of Technical Details

- Reads on code systems and value set are constructed from the resourceName attribute
- Reads on entity are constructed from the contents of the the name tag
- Default queries (No filtercomponent applied) defaults to resourceSynopsis where present and designation in entities.

CTS2 IMPLEMENTATION
COMMAND LINE AND XSLT TRANSFORMS

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.

CTS2 REST Client
Command Line

- For example:
- curl http://informatics.mayo.edu/cts2/rest/valuesets > valuesets.xml
- Redirects 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

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

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

CTS2 REST CLIENTS

PYTHON AND SCALA

REST Clients 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

REST Clients
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

REST Clients
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'})
```

REST Clients
Platform Independence: Python

Test the response and print it:

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

REST Clients
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']
```

REST Clients
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)

REST Clients
Platform Independence: Scala

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

REST Clients
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{
}
```

REST Clients
Platform Independence: Scala

Add some code to connect to the service:

```
val connection =
    new URL("http://informatics.mayo.edu/cts2/rest/valuesets").openConnection().asInstanceOf[HttpURLConnection]
```

REST Clients
Platform Independence: Scala

Get the input Stream:

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

Print it out:

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

CTS2 CLIENT IMPLEMENTATION BUILDING BLOCKS AND EXAMPLES

CTS2 UI BASICS

CTS2 JavaScript Examples
JSFiddle - Overview

- 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/

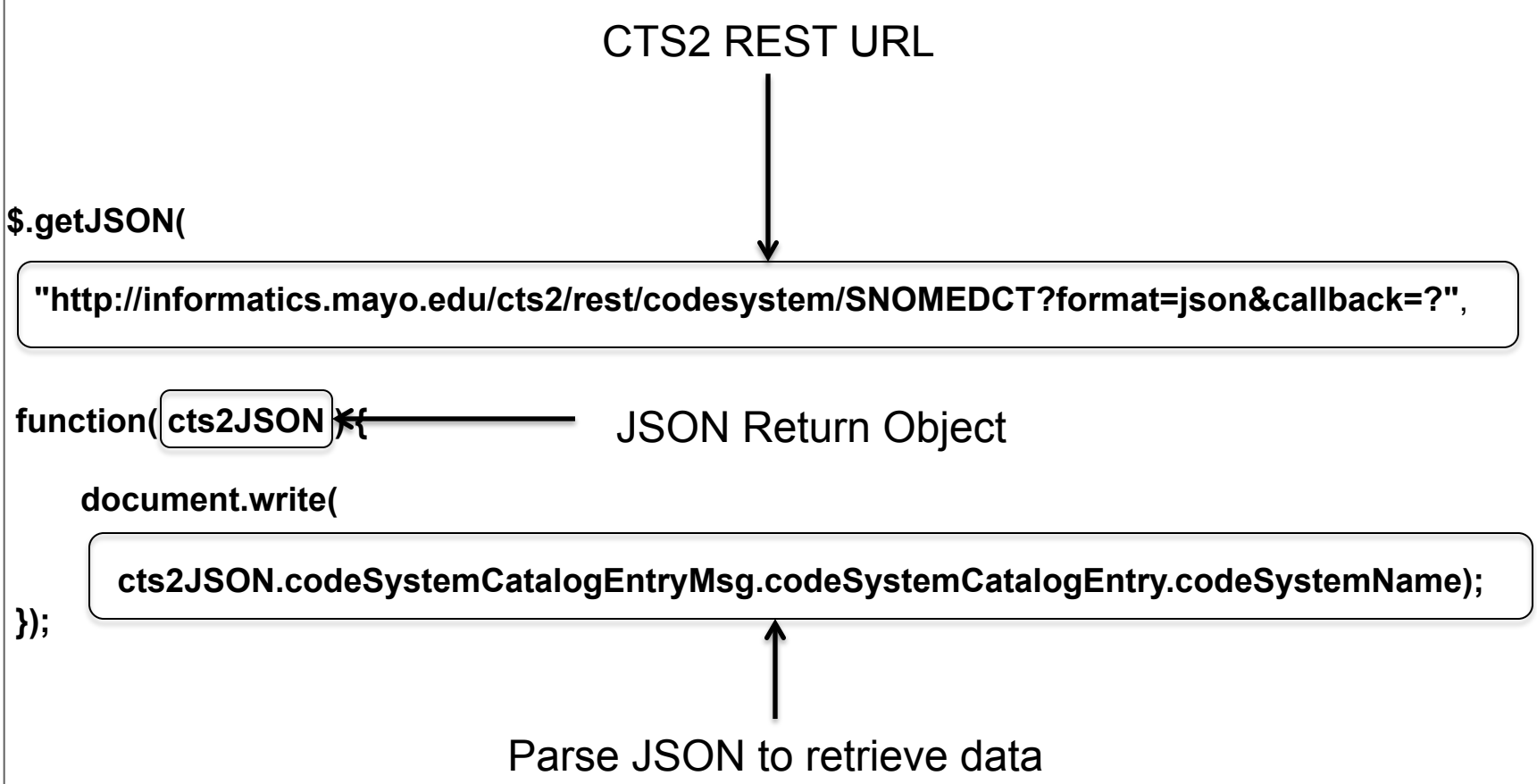
CTS2 JavaScript Examples
The Basics

- JavaScript and JSON
- Quickly and easily:
  - Query and display a code system:
    - Browser
      - http://informatics.mayo.edu/cts2/rest/codesystem/SNOMEDCT?format=json
    - JavaScript

CTS2 JavaScript Examples
The Basics

- Simple requirements:
  - URL
  - REST call that returns JSON object
  - Parse/display the JSON

CTS2 JavaScript Examples
The Basics



CTS2 JavaScript Examples
The Basics – Code Systems

- Quickly and easily:
  - Access a listing of codes systems
  - Access code system properties

- http://jsfiddle.net/coryendle/yPC4K/

CTS2 JavaScript Examples
The Basics – Value Sets

- Quickly and easily:
  - Access a listing of value sets
  - Access value sets from different CTS2 services
  - Access value set properties
  - Access value set entities (resolved)

- http://jsfiddle.net/coryendle/rLSUa/

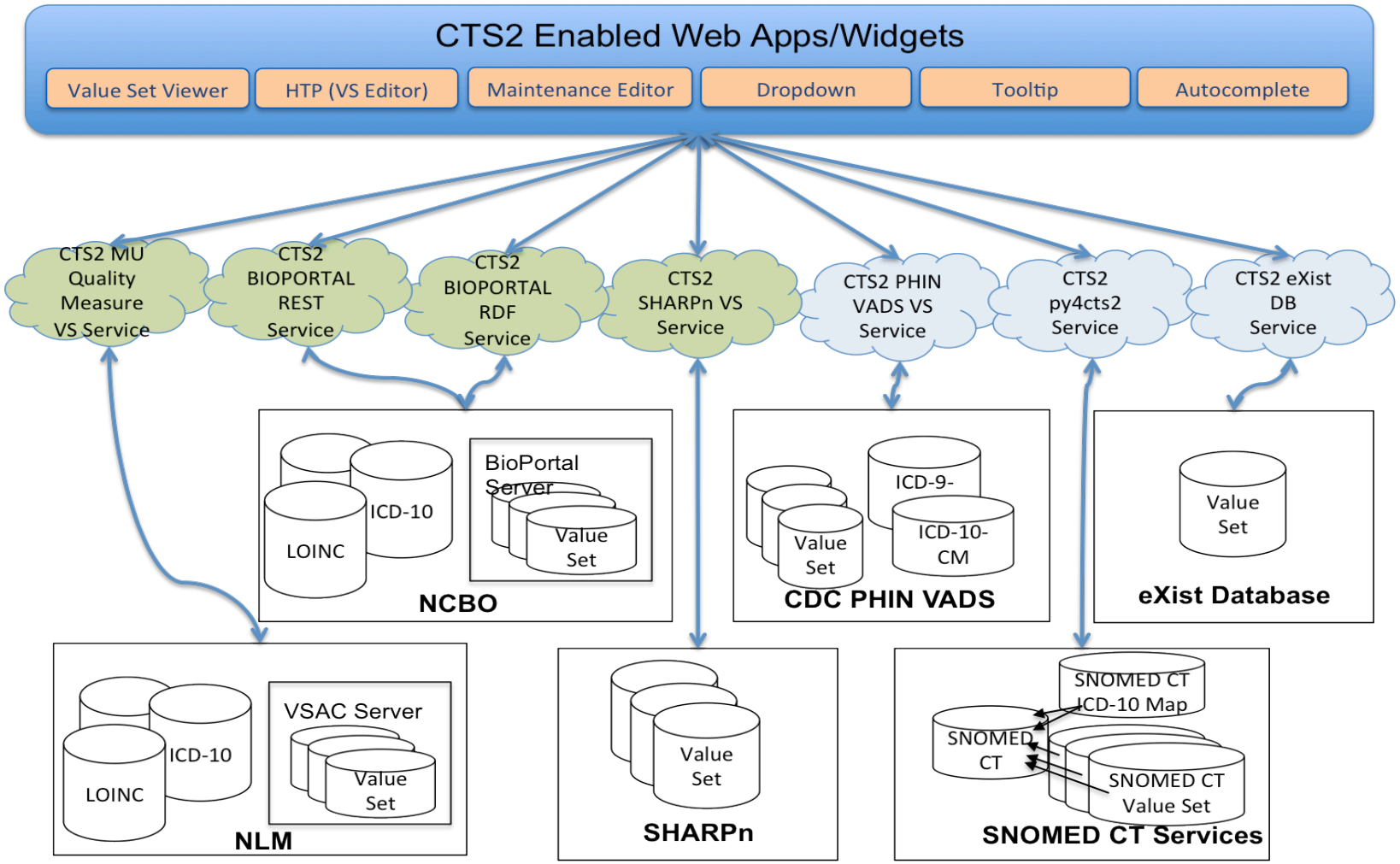
CTS2 JavaScript Examples
How to build from the Basics

- Building blocks
  - We can use the simple calls as the foundation to create more complex UI widgets and applications
- Several JavaScript UI Libraries available to create dynamic content:
  - jQuery
  - GWT
  - SmartGWT
  - Others

CTS2 UI WIDGETS AND APPLICATION EXAMPLES

CTS2 Implementations

Overview



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/

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

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

UI WIDGET: TOOLTIP

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

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

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

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

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

- http://jsfiddle.net/coryendle/n9bWF/
- 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: Add a row

- 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.valueSetCatalogEntry.entryState;

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

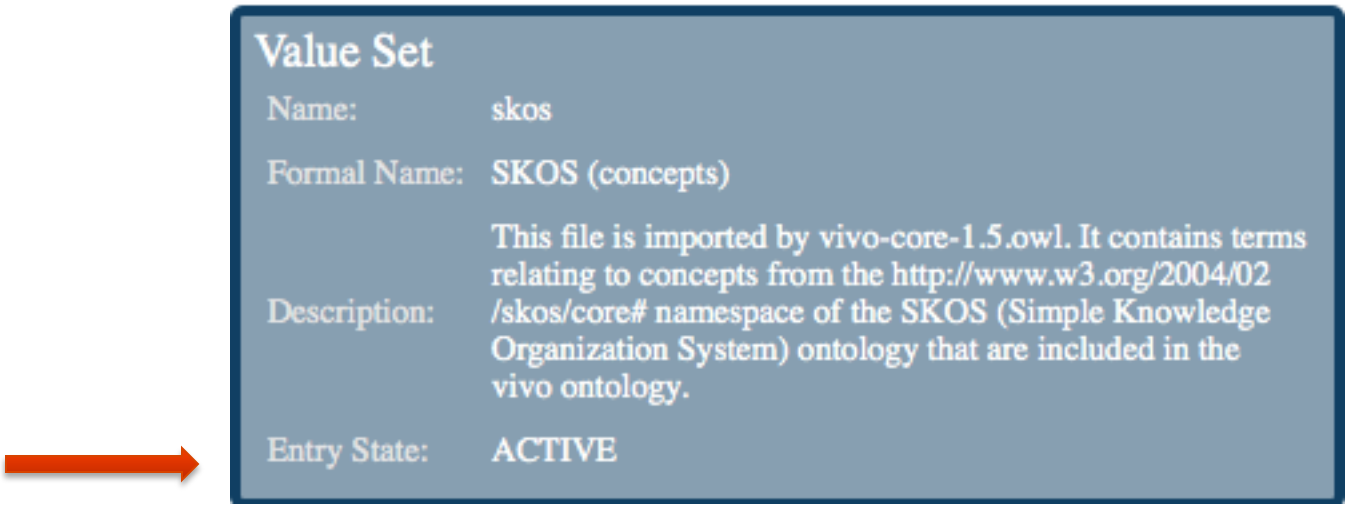
- Add another HTML row:

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

- Select the “Run” button to test your changes

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

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



UI WIDGET: DROPDOWN

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"

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?format=json
- We will use the data returned here to populate our CTS2 dropdown widget

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:

*

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

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:

CTS2 WEB APP IMPLEMENTATIONS

PUTTING IT ALL TOGETHER

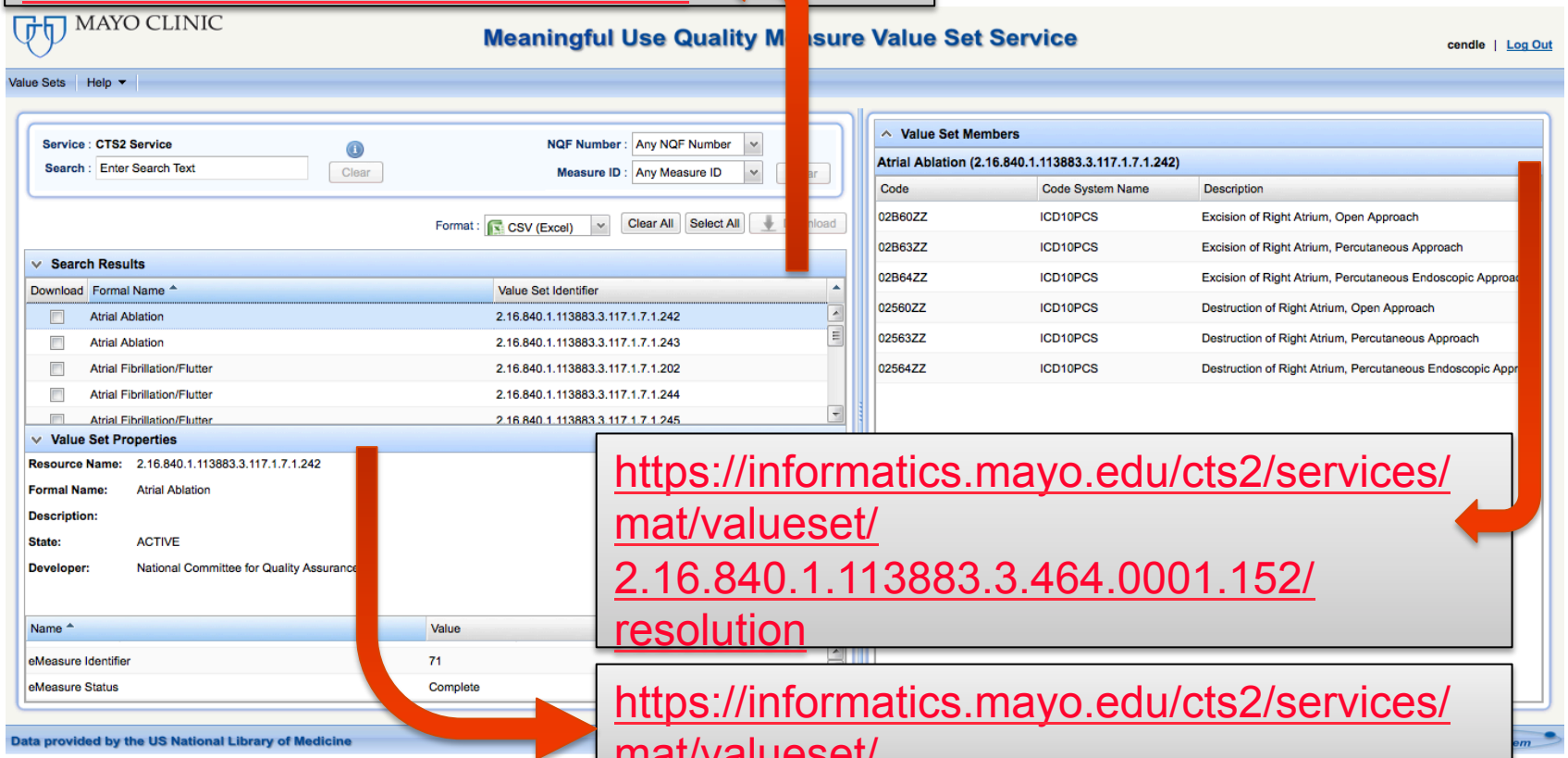
UIs created for CTS2 Implementations
CTS2 Value Set Viewer

- Generic Value Set Browser/export
  - Uses the building blocks
- 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/?showAll=true

UIs created for CTS2 Implementations

CTS2 Value Set Viewer

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

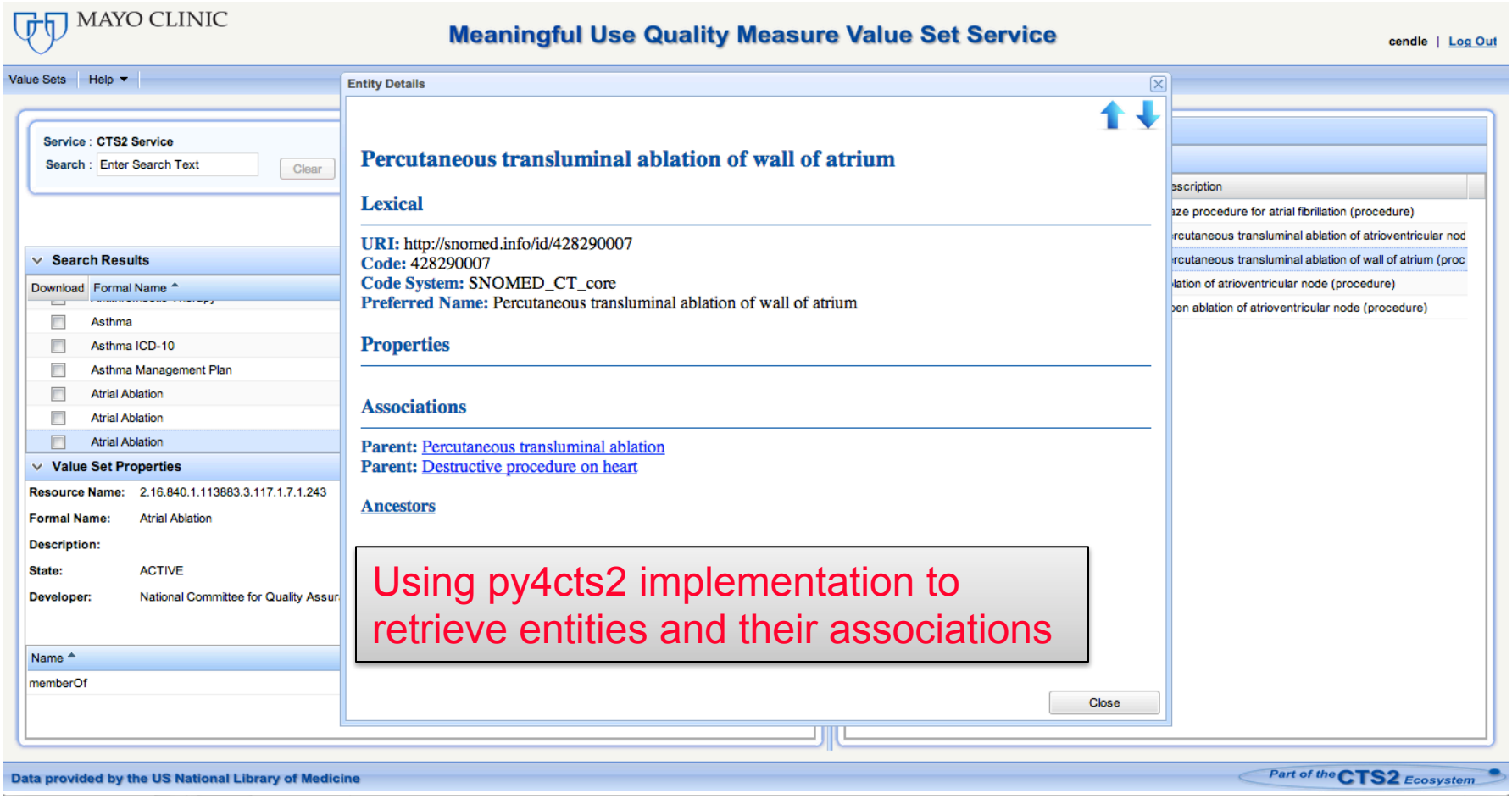


https://informatics.mayo.edu/cts2/services/mat/valueset/2.16.840.1.113883.3.464.0001.152/resolution

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



Web Applications
CTS2 Maintenance Editor

- Application for managing CodeSystems, Entities, and Change Sets
- Demonstration of CTS2 maintenance functionality
- Currently in development
- http://informatics.mayo.edu/exist/cts2/rest/editor/console

CTS2 CLIENT IMPLEMENTATION BUILDING BLOCKS AND EXAMPLES

SUMMARY

UI and Widget Review

- Quickly and easily retrieve CTS2 content
  - Using building blocks
- Customizable UI widgets available to quickly utilize CTS2 content
- Examples of web applications that utilized CTS2 service
  - Created with the building blocks
- Open Source

QUESTIONS

CTS2 JAVA REST CLIENT

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/use 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()); }
```

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); }
```

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";

CTS2 REST Client
Java

And add a request parameter to the connection:

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

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

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)

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.

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();

CTS2 REST Client
Java

Create a main method and run one of the methods:

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

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

CTS2 PLUGIN TUTORIAL OVERVIEW

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

CTS2 IMPLEMENTATION
SERVICE PLUGIN TUTORIAL

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

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

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

CTS2 Development Framework Service Plugin Example
Goals for this part of tutorial

- Connect to a database
- 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

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

Interactive Demo
Service Plugin

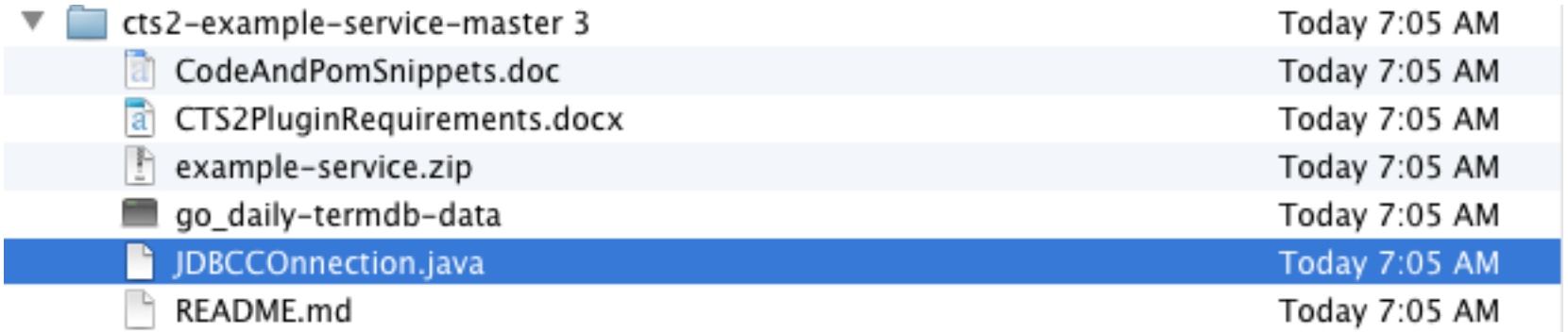
After running the clean install command your output should look something like this:

```
[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
```

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



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;
```

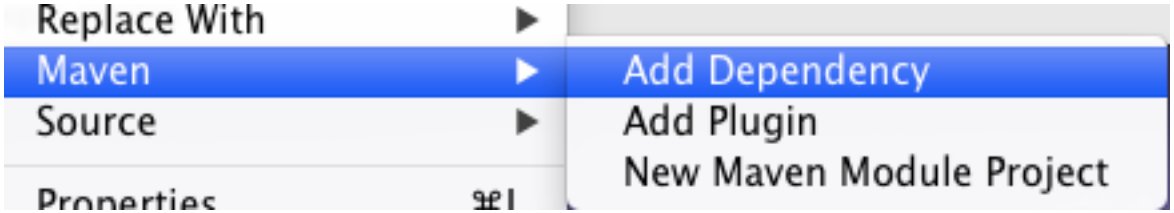
Interactive Demo
Service Plugin

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)
```

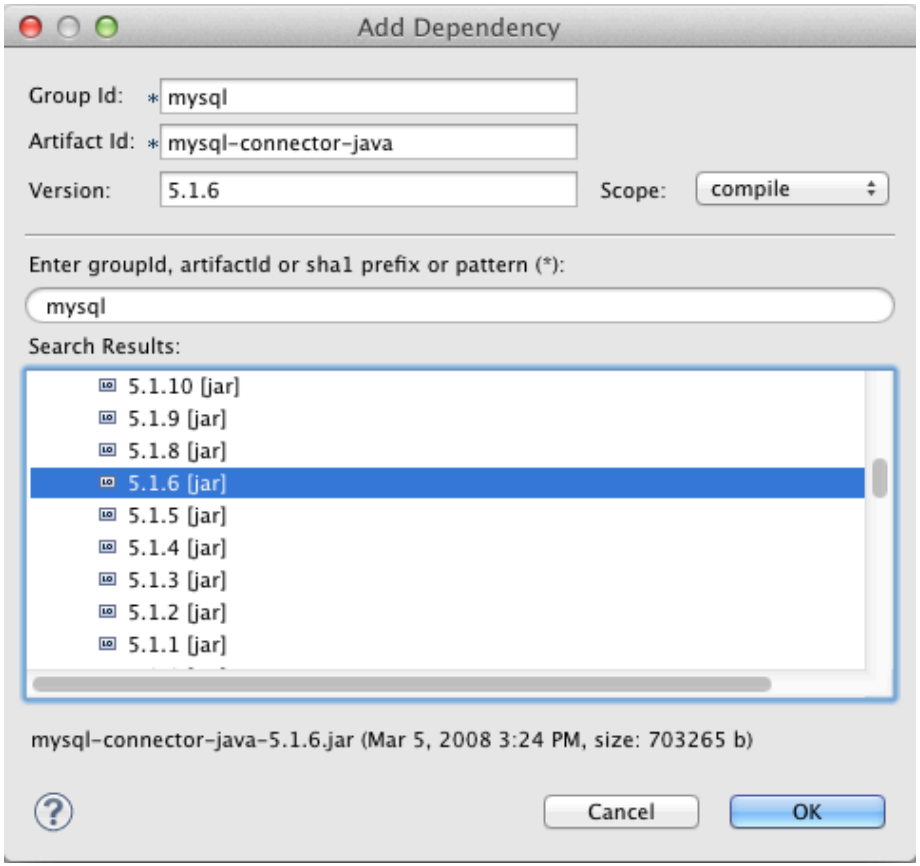
Interactive Demo
Service Plugin

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



Interactive Demo
Service Plugin

Add the MySQL connector version 5.1.6 to the pom file:



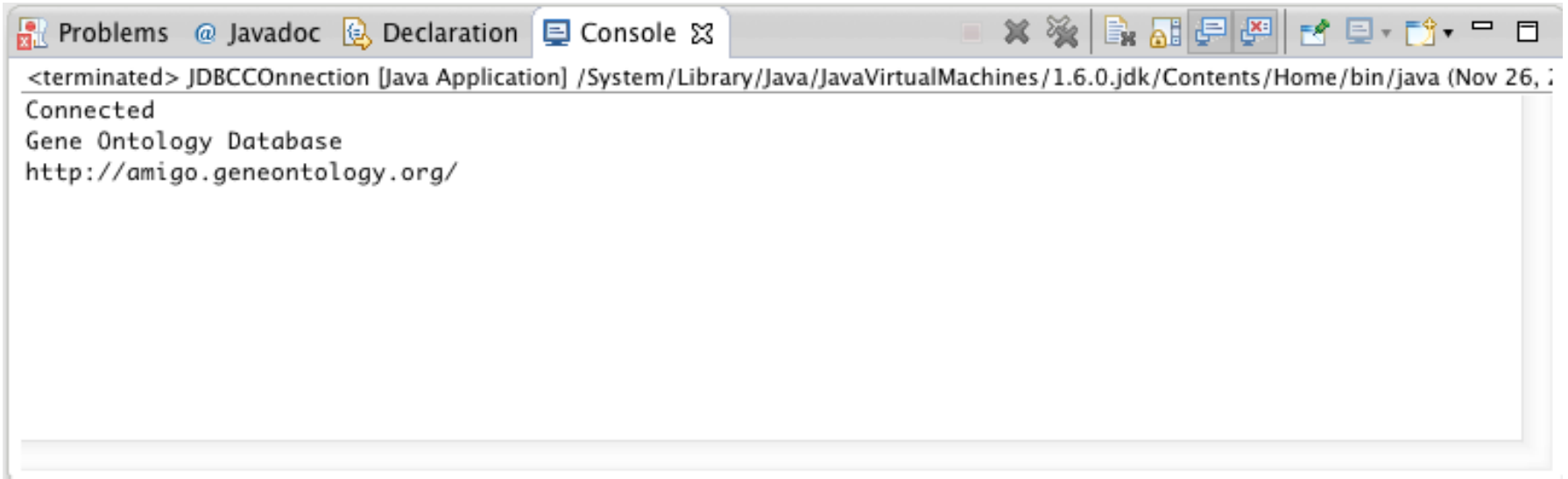
Interactive Demo
Service Plugin

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>
```

Interactive Demo
Service Plugin

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



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>
```

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

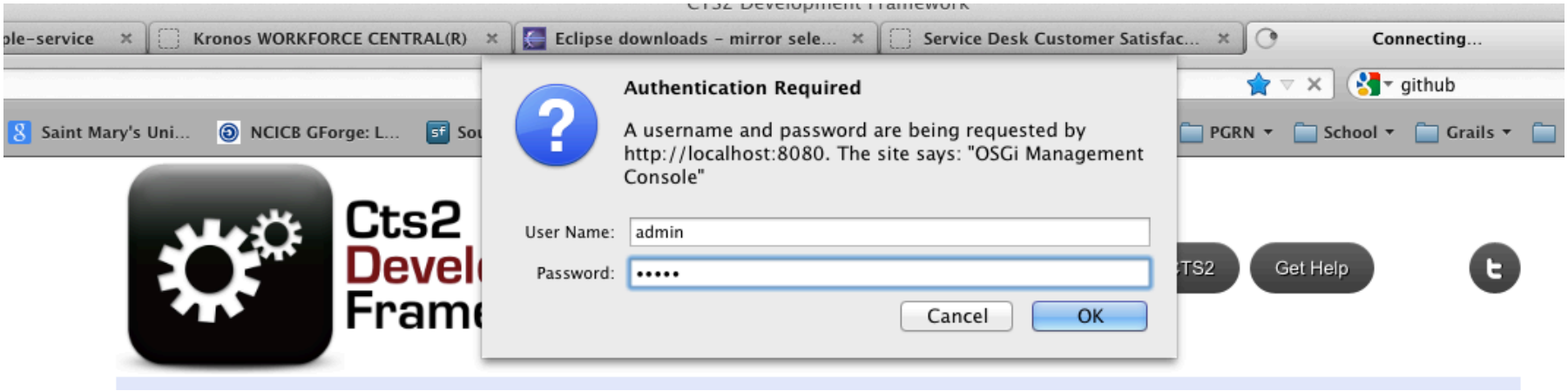
Interactive Demo
Service Plugin

Start the standalone server:

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

Interactive Demo
Service Plugin

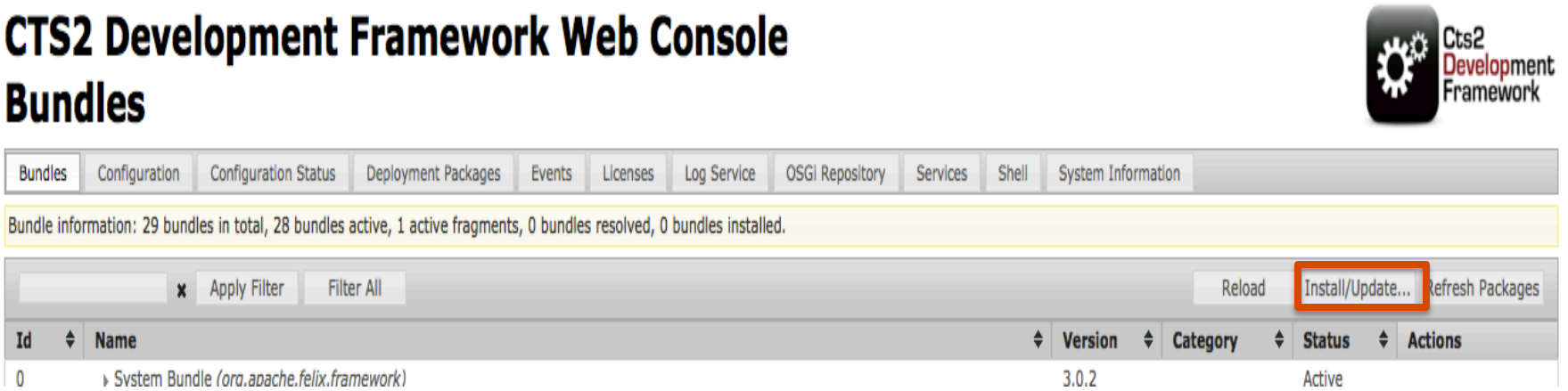
Click on the Admin Console button on the resulting web page:



Login using admin/admin.

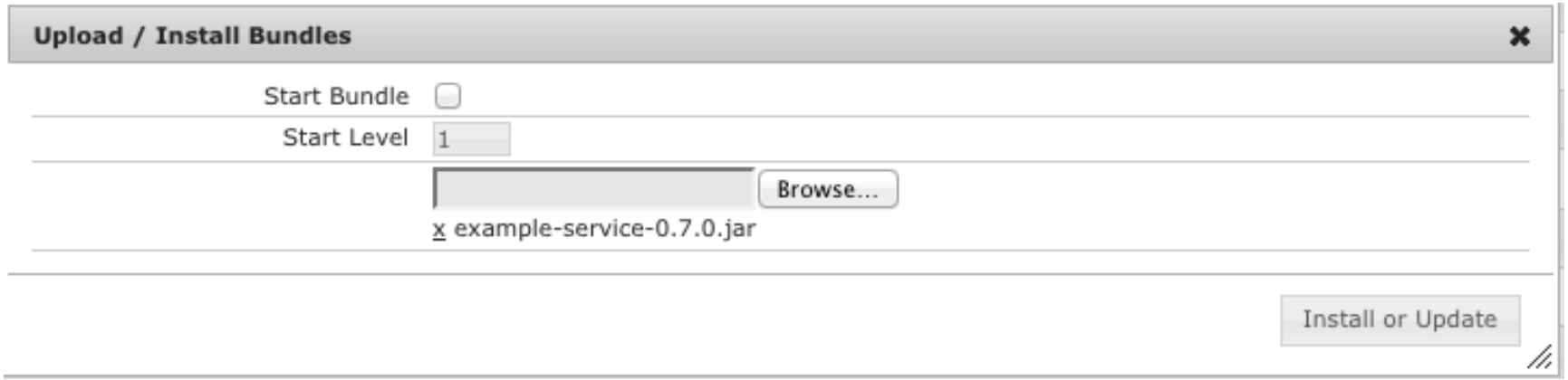
Interactive Demo: Service Plugin
Import Plugin to Framework

Click on install/update

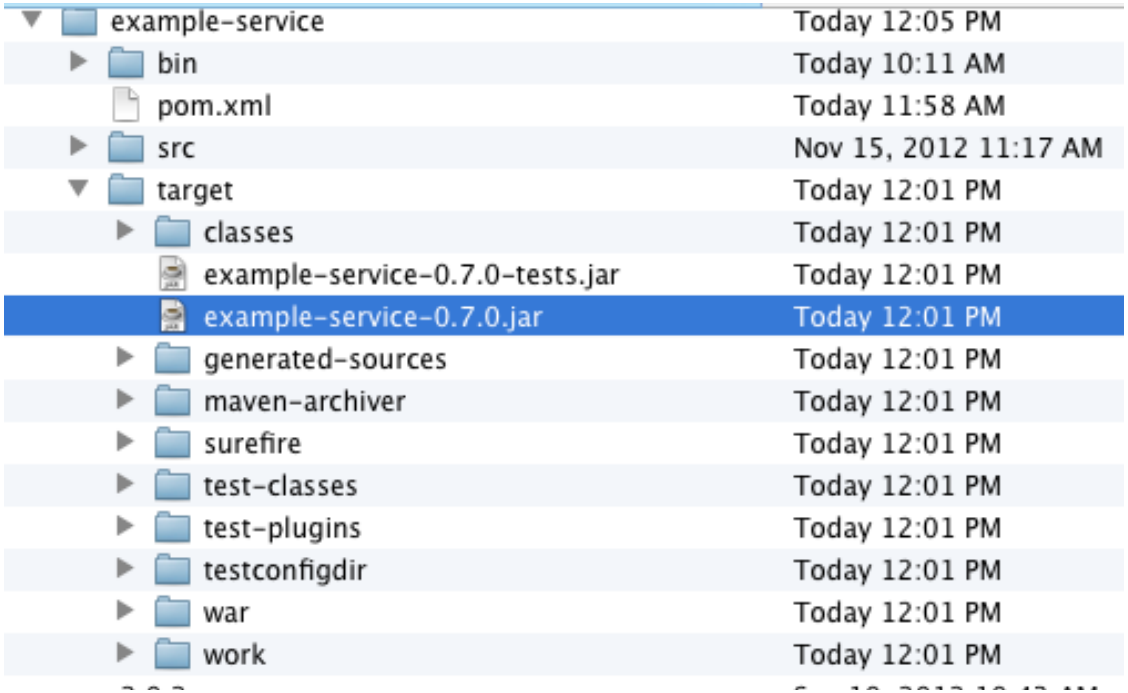


Interactive Demo: Service Plugin
Import Plugin to Framework

Upload the jar

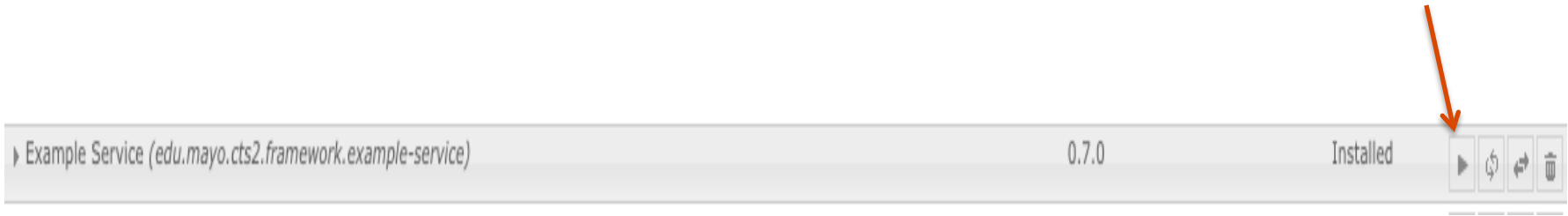


Interactive Demo: Service Plugin
Import Plugin to Framework



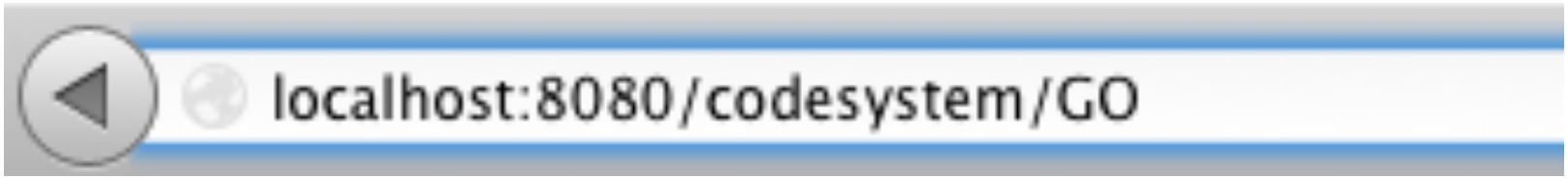
Interactive Demo: Service Plugin
Import Plugin to Framework

If necessary start the plugin:



Interactive Demo: Service Plugin
Import Plugin to Framework

Create a REST call in a browser address bar:



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>
```

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

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/

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

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/

Questions & Discussion