# The UMLS Knowledge Source Server

Anantha Bangalore
Karen Thorn

## Outline

- Background
- Current System
  - Design tenets
  - System Architecture
  - Authentication
  - RMI & Socket Server API
  - Web Access
  - UMLSKS Object Model
  - Drawbacks

## Outline

- New System
  - Introduction
  - System Architecture
  - Authentication
  - Web Services
  - Portal
- Demo

## Background

- UMLS involves the development of a set of widely distributed Knowledge Sources: Metathesaurus, Semantic Network and SPECIALIST Lexicon
- Can be used to compensate for the differences in the way concepts are expressed in a variety of biomedical vocabularies.
- Currently, over 1900 individuals and institutions have signed the UMLS License Agreement, enabling them to receive the UMLS data either on CD-ROM or through the UMLS Knowledge Source Server (UMLSKS)

## Background

- UMLS is large and complex and presents significant challenges in retrieving information in a comprehensive way.
- Centrally managed UMLSKS provides system developers with UMLS information remotely and on demand.
  - developers do not need to invest time and effort in understanding the structure of the data files and other details to use the UMLS data in their applications

## Background

- First version of the internet based UMLSKS was made available in 1995.
- Ran on a single server and was written in “C” programming language.
- New system is written in Java and runs on multiple servers.
- New system is more flexible, extensible, scalable and provides better access to UMLS data.

## Design tenets

- Extensibility for ease of new feature integration.
- Flexibility by providing a rich API set to allow system developers access to all UMLS data elements
- Access to data through multiple channels (web, XML/socket API, and Java API)
- Provision of a unified data model for the Knowledge Sources for use by application developers.

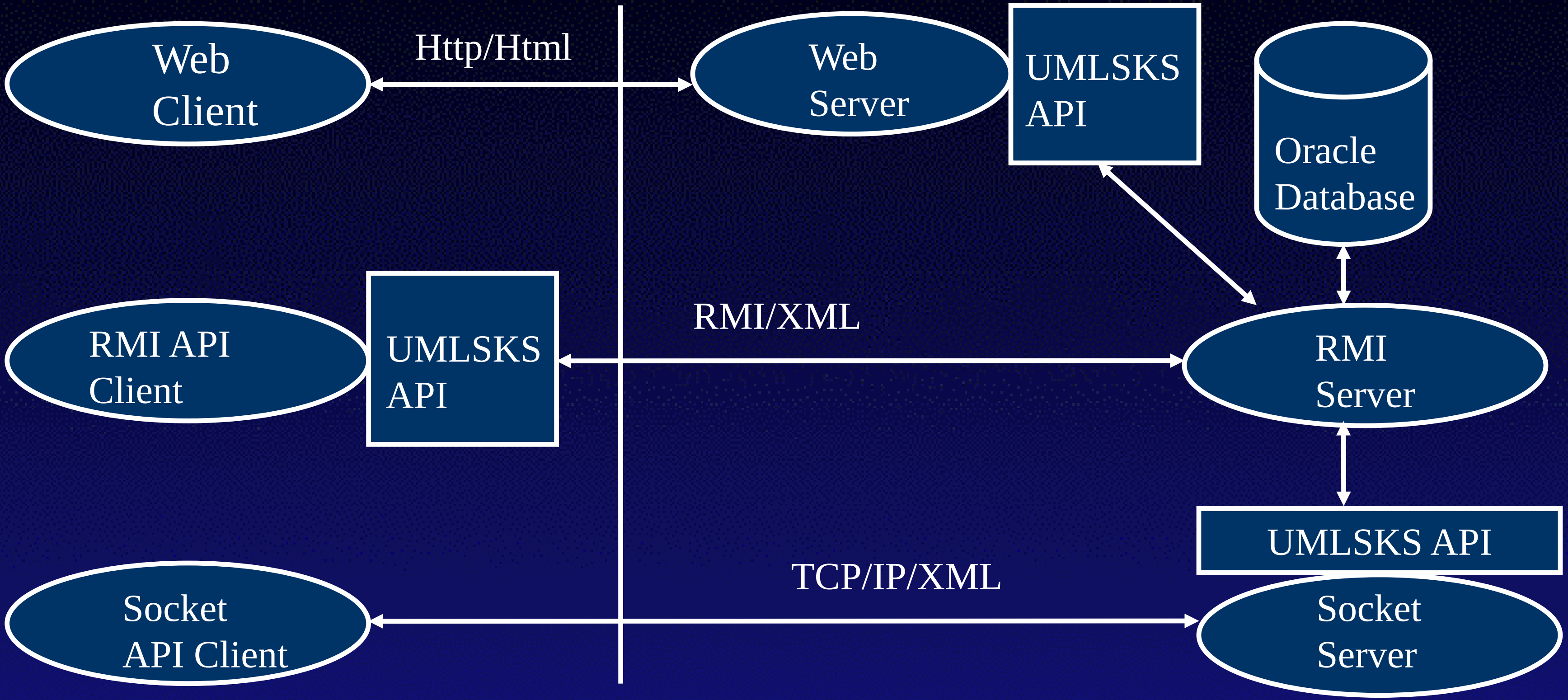
## Design tenets

- Scalability in handling ever increasing user loads and increasing numbers of UMLS source vocabularies
- Performance enhancement to provide fast access to UMLS data
- Ease of administration by NLM staff and contractors

## System Architecture

- UMLSKS services can be accessed in three different ways:
  - Web
  - Java API
  - Socket-based API
- Data is returned in XML
- A set of classes that provide a data-centric representation of the UMLS is provided with the API and is capable of reading the XML .

# System Architecture



## Authentication

- Username and password based authentication for web users.
- API users have to register all IP addresses from which they will potentially access the backend server.
  - Primitive and potentially unsecure
  - Users using an ISP are not guaranteed to have the same IP address every time.
- Better to have the same authentication mechanism for all users. (web and API)

## Web Access

- User issues  http request to Web server
- Web server is implemeted as a collection of servlets. (Tomcat, Turbine, Velocity)
- Web server issues  request using Java’s Remote Method Invocation (RMI) methodology to execute a particular function.
- RMI server returns result in XML to web server.
- Web server applies XSLT stylesheets to XML to convert it into HTML.

## Access using Java API

- Establish connection to the backend RMI server.
- Make method calls directly from  Java programs.
- Underlying communications mechanism is hidden from the user
- Await receipt of XML response from server
- Server processes requests for Knowledge Source data, accessing an Oracle® database .
- May use the object model to interpret returned XML.

## Java API Usage Example

- Retrieve Basic Concept information about the concept Brain.
- Basic Concept information consists of :
  - CUI
  - Concept Name
  - Definitions
  - Semantic Types
  - Term Information
  - Source Information

# Java API Usage Example

```
// 1) Establish connection to the UMLSKS server
    KSSRetrieverV3_0 retriever =   (KSSRetrieverV3_0)
    java.rmi.Naming.lookup(“KSSRetriever”);
// 2) Request basic concept for CUI ‘C0000001’
    char[] result = retriever.getBasicConceptProperties(  "2003AB",
    "C0015967", null, "ENG, true);
// 3) Interpret the XML result into the Concept instance
      Concept myConcept = new  Concept(new String(result));
// 4) Print out the concept’s name
      System.out.println(“Concept Name=‘” + myConcept.getCN() +
    “’”);
```

## Access using Socket-based API

- Any programming language that supports socket communication can be used.
- Clients establish socket connection to the TCP/IP server
- Compose a UMLSKS API request in XML format.
- XML API request resemble the Java API methods.
- Server sends request to RMI backend.
- Await receipt of XML response backend server.

## XML Query example

```
<?xml version="1.0"?>
   <getBasicConceptProperties version="1.0">
         <release>2003AB</release>
         <cui>C0001175</cui>
         <sablist>
                  MSH
         </sablist>
         <language>ENG</language>
         <noSuppressibles/>
   </getBasicConceptProperties>
```

## UMLS Object Model

- Previously application developers needed to understand the relational data model in order to abstract the UMLS contents into application level components.
- Development of a single extensible  object-oriented data model of the UMLS will enable developers to focus more on application code.
- Object-oriented data model lends well to extension of functionality

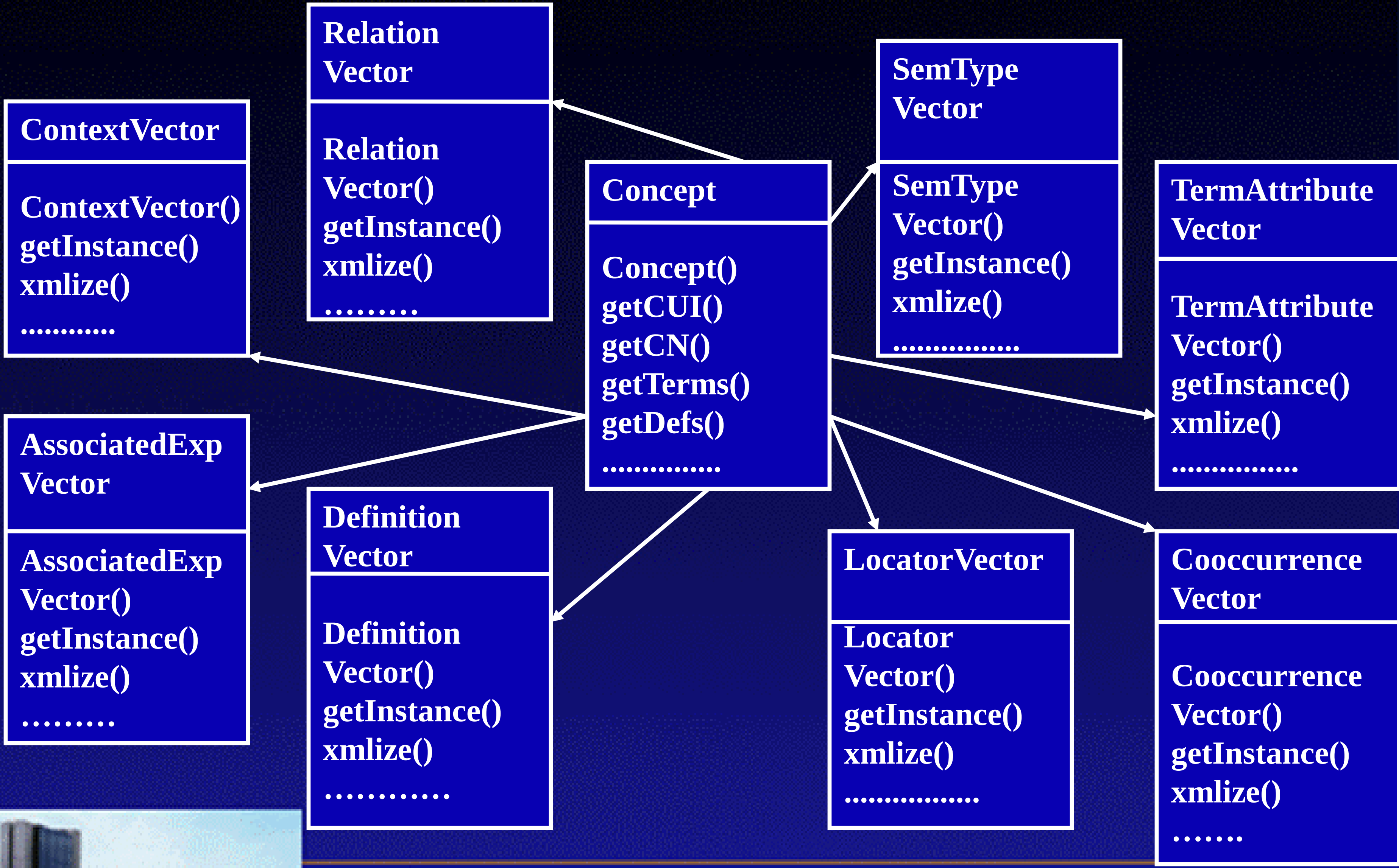
## UMLS Object Model

- Allows developer communities to develop and share software extensions of the model.
- Each Knowledge Source has its own set of classes that form an object-oriented view of the data.

## Metathesaurus Object Model

- Metathesaurus object model makes explicit the complex structure of the UMLS Metathesaurus in a clear, understandable, navigable and extensible way.
- The Meta Object model is composed of approximately 60 classes representing the various Metathesaurus components.
- Subset of these classes can interpret the XML returned by the UMLSKS API methods and instances of each class may be queried for the details as returned from the method calls

# Metathesaurus Object Model



## Semantic Type Object Model

- Semantic network Object model abstracts the nodes within the network and provides methods to extract details about each node.
- Classes also represent the network traversal allowing the network structure to be queried.
- Thirteen classes that comprise the Object Model for the Semantic Network provide XML interpretation capabilities and containers for the details of semantic nodes

## Lexicon Object Model

- Lexicon is intended to be a general English lexicon that includes many biomedical terms.
- Object Model for the SPECIALIST Lexicon contains classes that can interpret the XML generated by the associated UMLSKS API accessing SPECIALIST Lexicon data.

## Advantages

- New system permits quicker incorporation and availability of new UMLS data set releases. (one to two weeks)
- Faster and more reliable access to data by load balancing across multiple servers.
- Rich set of API provides access to all of the UMLS data.
- API's can be incorporated into programs written in any language for any platform.

## Drawbacks

- Not possible to add new features without completely rebuilding the entire server, stopping and restarting the server.
- In  case of API users non-standard ports have to be opened on both the client and the server side. This is increasingly becoming an issue with heightened security restrictions on our side as well as for clients (especially commercial companies).
- Have to maintain two servers on the backend:
  - Rmi for Java Users
  - Socket Server for Non Java users

## Drawbacks

- Authentication on the backend is based on a primitive model of restricting access based on IP addresses.
- While the current interface based on XML and XSLT stylesheets provide a great deal of flexibility in quickly changing an existing display, it will be nice to allow individual users to tailor their own display for their own purposes.

## New UMLSKS

- Backend is being implemented as a set of web services.
  - Fast becoming a commonly accepted standard.
  - Address firewall issues
  - Employ dynamic deployment mode of new functionality
  - Facilitate machine and API programming language freedom
  - Provides software framework for easier integration of new feature

## New UMLSKS

- Authentication for API users will be based on username and password.
  - Single Sign-on solution based on Yale’s CAS
  - Uses a Kerberos type proxy ticket for granting access to Backend web services

Web Interface is based on Portal Technology and AJAX
  - Allows building of dynamic custom views
  - More interactive

## System Architecture

- Tomcat is used as the deployment platform  for all the web services and the portlets.
  - enables deployment without a restart of the entire system
- Apache Axis is used for developing the web services.
- New functionality may be added to the system without impacting users currently connected to the services

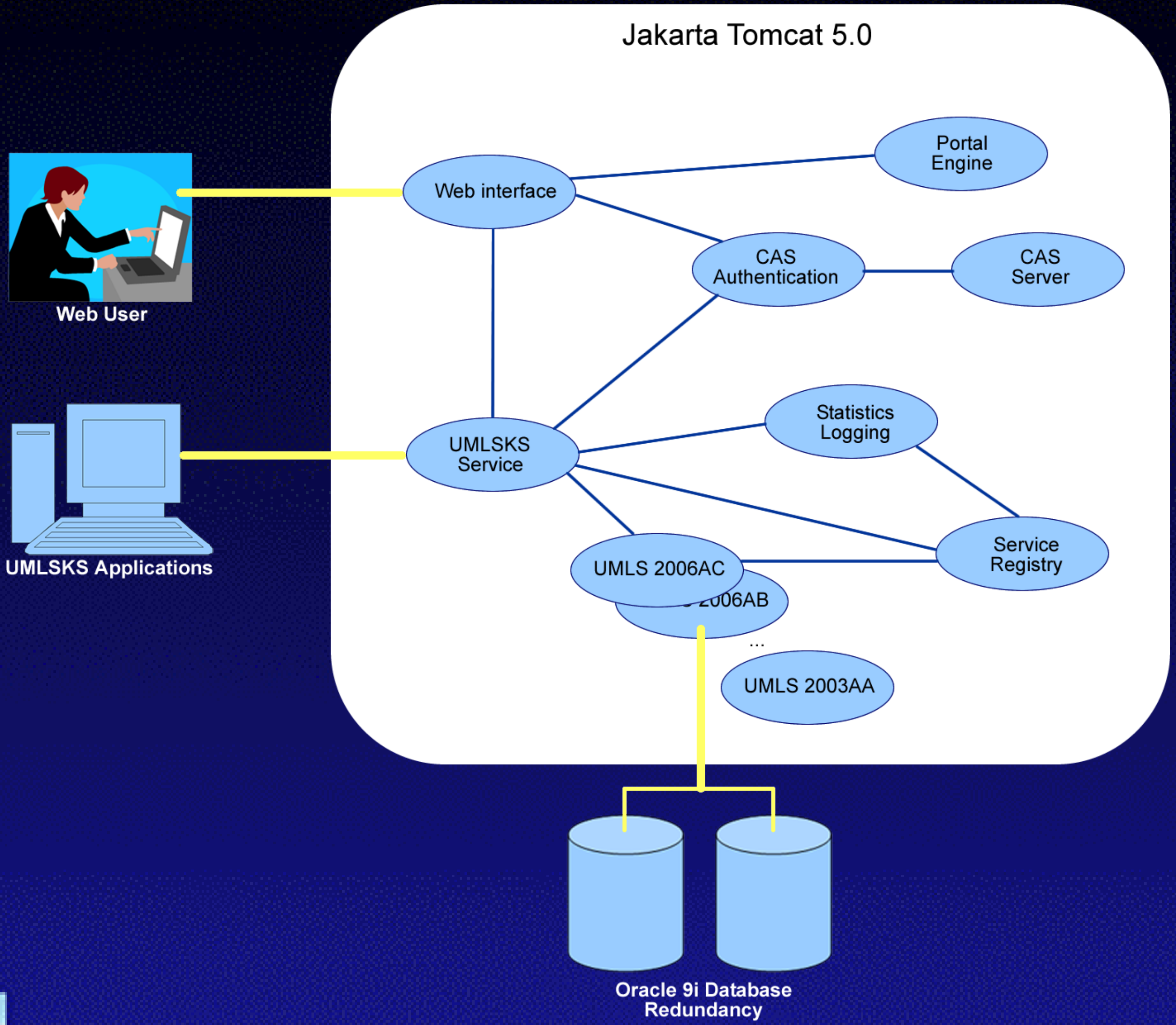
## System Architecture

- Each web service is designed as a small set of coherent, related classes that provide a focused set of functions giving the software a small footprint and allowing it to execute at a higher level of performance
- Users can program against the Web services by implementing a web services client.
- CAS server is used for authentication by both API and web users. In case of API users, NLM will provide a digital certificate that has to be stored on the client site and used for accessing the Web services on the server.

# System Architecture

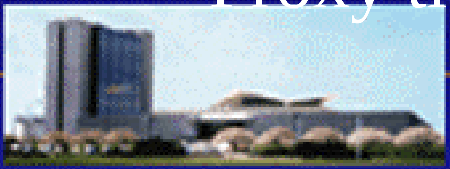
- Web interface is implemented using a Portal framework
  - Uportal – freeware portal framework
  - JSR-168 Portlet API
- Can be customized by end-user both in look and feel and in available content using applications provided by the portal

# System Architecture

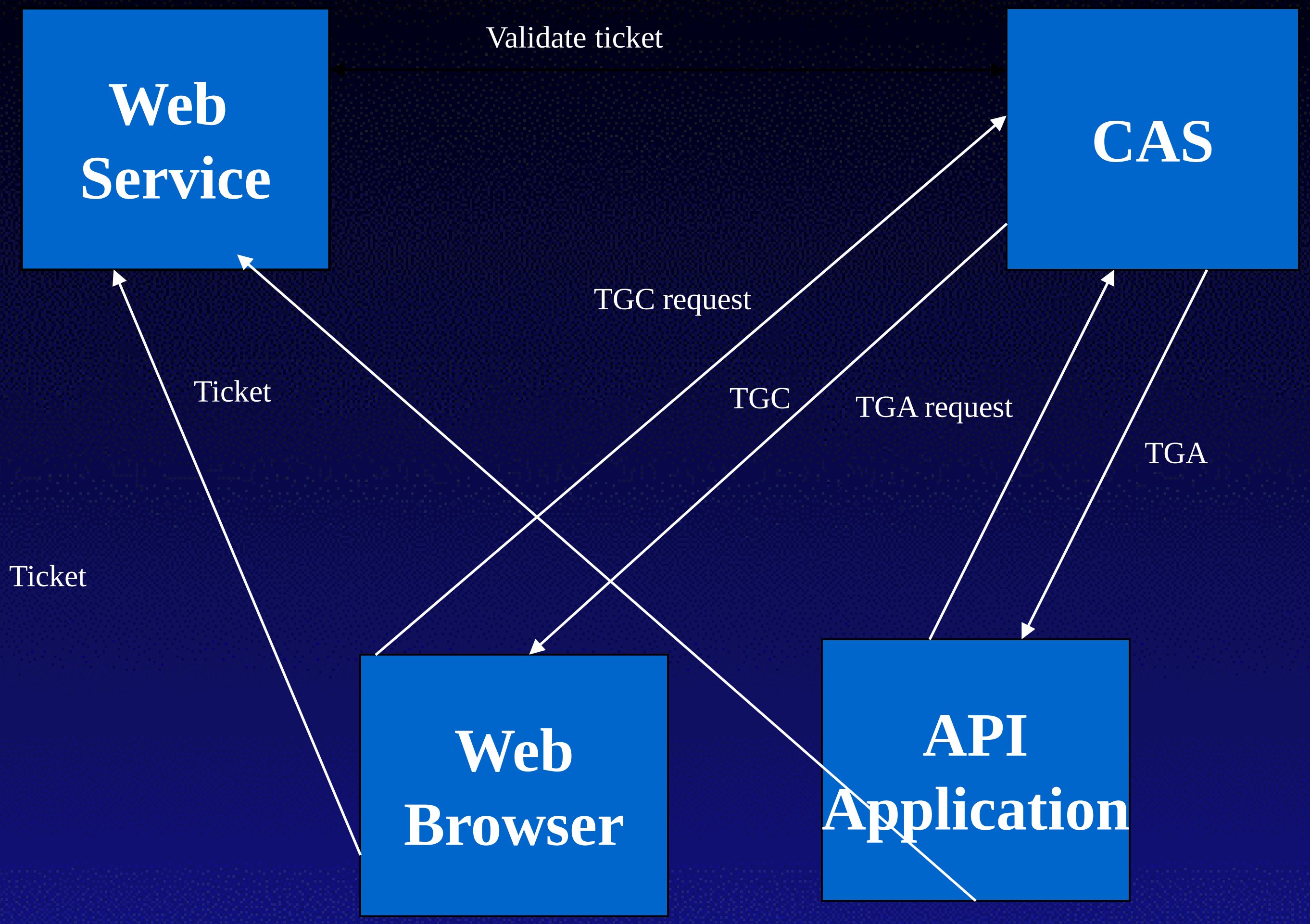


## CAS

- Single Sign On Solution based loosely on Kerberos from Yale University
- Designed as a standalone web application implemented as a set of Java servlets that run on a secure web server.
- Can easily be configured to use many types of authentication on the server side
  - LDAP, UNIX, DB based
- Generates one use Proxy Ticket or a Proxy Granting Ticket which can be used to generate Proxy tickets .



# CAS


Validate ticket
Web Service
CAS
TGC request
Ticket
TGC
TGA request
TGA
Ticket
Web Browser
API Application

## Web Services

- Web services as defined by Sun Microsystems are “web-based enterprise applications that use XML-based standards and transport protocols to exchange data with calling clients.”
- Deployed using an Application server such as Tomcat on port 80
- Apache Axis is an open-source, XML-based web service framework that implements the Simple Object Access Protocol (SOAP) to exchange XML messages between software components.

## Web Services

- UMLSKS components are developed and deployed to Tomcat with Axis performing the SOAP messaging between the client and web service implementation.
- Significant decoupling of components and results in multiple web services to perform the same functions available in the current release.
- Single web service is deployed to handle requests from users.
  - accesses internal services that access each UMLS release's data, authorization, registry and logging function

## Web Services Client

- UMLSKS Webservice description Language (WSDL)
  - Defines the available operations from the UMLSKS web service
- Configuration
  - Service endpoint
- Invocation
  - Stub-based model
  - Dynamic Invocation Interface

## Stub-based Model

- Axis WSDL2Java
- Builds stubs, skeletons and data types from the UMLSKS WSDL
- Java client calls instantiated stub to connect to service
- Java classed created that represent argument and return types to the method calls

## Dynamic Invocation Interface (DII)

- Dynamic creation of call to service
- Client creates Axis Service and Call objects
- Client registers argument types and return types with call
- Client "invokes" call object to obtain results

## UMLSKS Webservice Invocation

```
org.apache.axis.client.Service service = new Service();
org.apache.axis.client.Call call = (Call)service.createCall();

//   Register the argument types
String namespaceURI = “http://umlsks.nlm.nih.gov”;
javax.xml.namespace.QName argName = new QName(namespaceURI,
     “ConceptIdExactRequest");

call.registerTypeMapping(ConceptIdExactRequest.class, argName,
      new BeanSerializerFactory(ConceptIdExactRequest.class, argName),
      new BeanDeserializerFactory(ConceptIdExactRequest.class,
     argName));
```

*… (type mapping registration of remaining parent classes for the argument type*

# UMLSKS Webservice Invocation

```
//  Register the return object types
javax.xml.namespace.QName retName = new
    QName(namespaceURI, “ConceptIdGroup");
call.registerTypeMapping(ConceptIdGroup.class, argName,
      new BeanSerializerFactory(ConceptIdGroup.class,
    argName),
      new BeanDeserializerFactory(ConceptIdGroup.class,
    argName));
```

*… (type mapping registration of remaining parent classes for the return type*

## UMLSKS Webservice Invocation

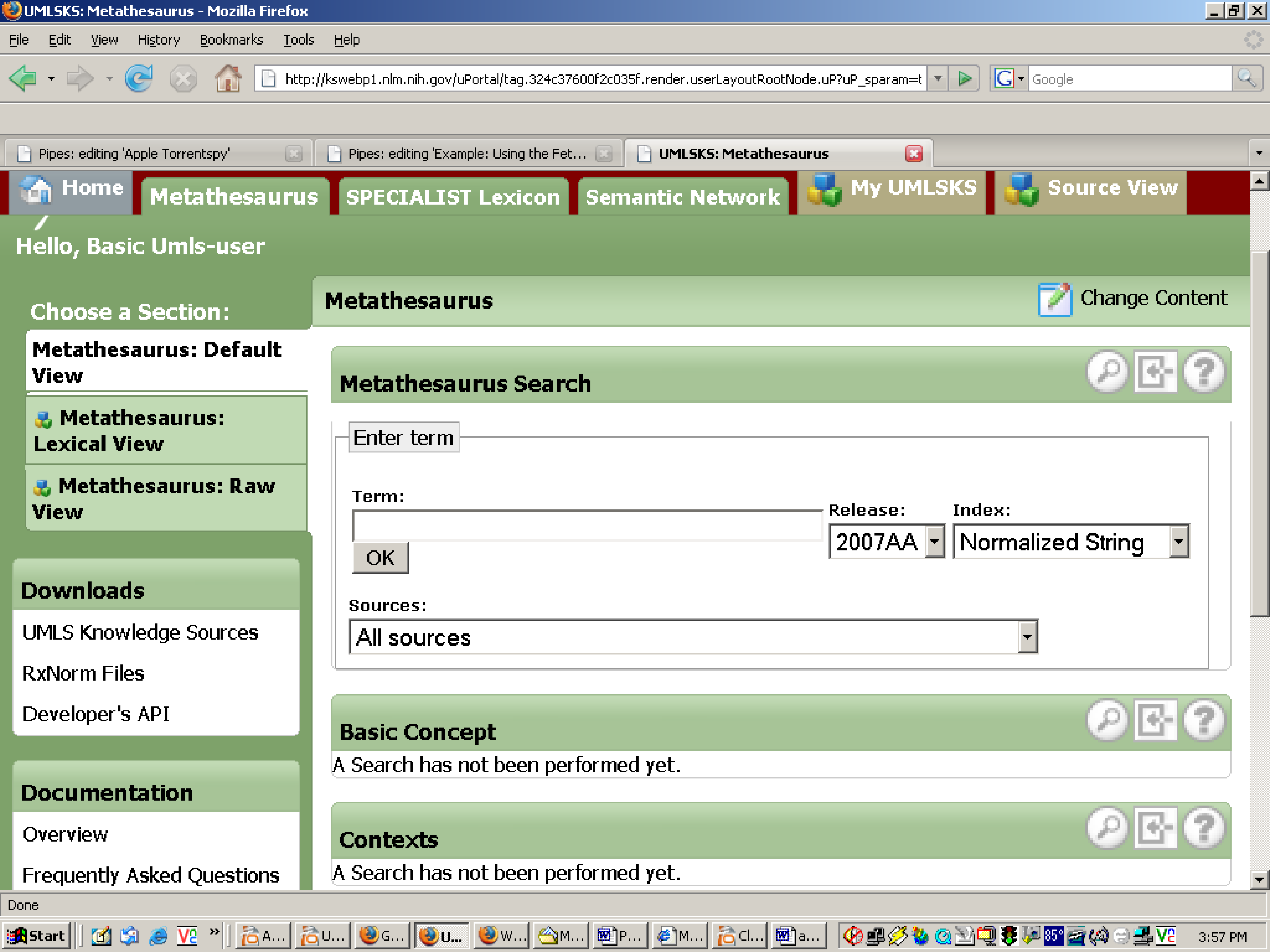
```
call.setTargetEndpointAddress(endpointURL);
call.setOperationName(new QName(namespaceURI,
    “findCUIByExact"));
call.addParameter("arg1", argName, ParameterMode.IN);
call.setReturnType(retName, ConceptIdGroup.class);

Object[] args = new Object[1];
args[0] = request;

return (ConceptIdGroup)call.invoke(args);
```

## Portal

- Web interface that can be customized by the end-user both in look and feel and in available content using applications provided by the portal
- Functions as an aggregator of portlets which can be thought of as miniature Web applications that run inside a portal page alongside any number of similar entities.
- Portlet can be thought of as a miniature Web application that is running inside of a portal page along side any number of similar entities

# Portal

- Each of the portlets generate fragments of mark-up, which the portal container ultimately pieces together to create a complete page
- Portlet registry, similar to the web services registry, contains a list of available portlets
- Java Portlet Specification (JSR168) enables interoperability for portlets between different web portals
- open source, freely available portal framework called uPortal was selected for the development of our portal-based interface


UMLSKS: Metathesaurus - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google
Pipes: editing 'Apple Torrentspy'
Pipes: editing 'Example: Using the Fet...
UMLSKS: Metathesaurus
Home
Metathesaurus
SPECIALIST Lexicon
Semantic Network
My UMLSKS
Source View
Change Content
OK
Done
Start
3:57 PM

http://kswebp1.nlm.nih.gov/uPortal/tag.324c37600f2c035f.render.userLayoutRootNode.uP?uP_sparam=t

Hello, Basic Umls-user

Choose a Section:

- Metathesaurus: Default View
- Metathesaurus: Lexical View
- Metathesaurus: Raw View

**Downloads**

- UMLS Knowledge Sources
- RxNorm Files
- Developer's API

**Documentation**

- Overview
- Frequently Asked Questions

**Metathesaurus**

**Metathesaurus Search**

Enter term

Term:

Release: 2007AA

Index: Normalized String

Sources: All sources

**Basic Concept**

A Search has not been performed yet.

**Contexts**

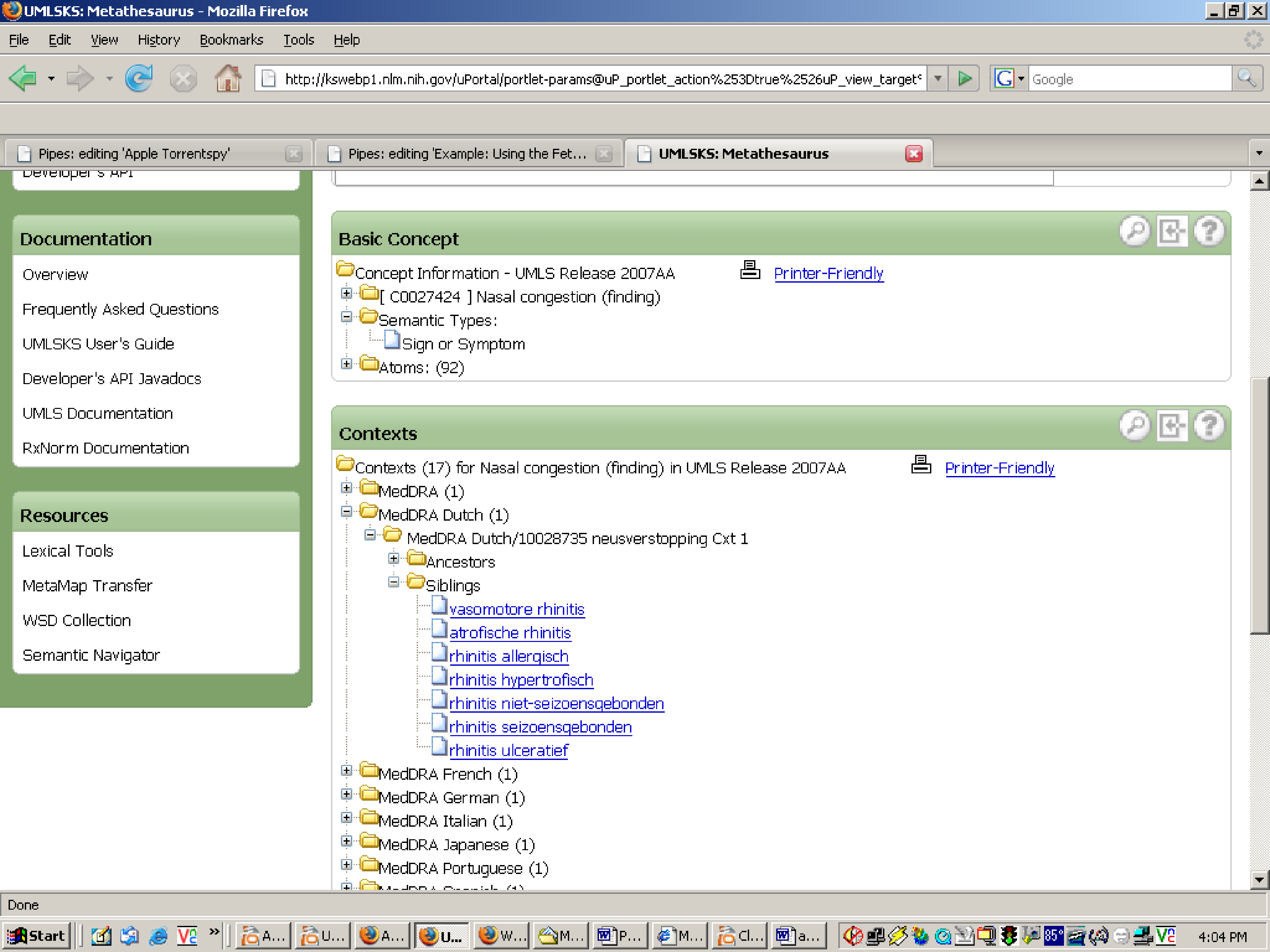
A Search has not been performed yet.

# AJAX

- AJAX (Asynchronous Javascript and XML) allows creation on highly interactive sites.
  - More responsive to users
- Allows the exchange of small amounts of data with the server behind the scenes, eliminating the need for the entire web page to be reloaded each time the user requests a change.
- Increases the interactivity speed and usability of web pages

## AJAX

- Bandwidth usage
  - HTML generated locally within browser using javascript.
  - Ajax web pages load relatively quickly since payload coming down is smaller in size.
- Separation of data, format style and function
- Reliance on Javascript
  - Not handled uniformly across all browsers.



UMLSKS: Metathesaurus - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://kswebp1.nlm.nih.gov/uPortal/portlet-params@uP_portlet_action%253Dtrue%2526uP_view_target

Pipes: editing 'Apple Torrentspy' | Pipes: editing 'Example: Using the Fet... | UMLSKS: Metathesaurus

**Documentation**

Overview

Frequently Asked Questions

UMLSKS User's Guide

Developer's API Javadocs

UMLS Documentation

RxNorm Documentation

**Resources**

Lexical Tools

MetaMap Transfer

WSD Collection

Semantic Navigator

**Basic Concept**

Concept Information - UMLS Release 2007AA    Printer-Friendly

- [ C0027424 ] Nasal congestion (finding)
- Semantic Types:
  - Sign or Symptom
- Atoms: (92)

**Contexts**

Contexts (17) for Nasal congestion (finding) in UMLS Release 2007AA    Printer-Friendly

- MedDRA (1)
- MedDRA Dutch (1)
  - MedDRA Dutch/10028735 neusverstopping Cxt 1
    - Ancestors
    - Siblings
      - vasomotore rhinitis
      - atrofische rhinitis
      - rhinitis allergisch
      - rhinitis hypertrofisch
      - rhinitis niet-seizoensgebonden
      - rhinitis seizoensgebonden
      - rhinitis ulceratief
- MedDRA French (1)
- MedDRA German (1)
- MedDRA Italian (1)
- MedDRA Japanese (1)
- MedDRA Portuguese (1)

Done

4:04 PM

# Demo