jTerm: A Server for Terminological Systems
Michael A. Hogarth, Jim Stone, Fredric A. Gorn
1 Oracle Corporation, Redwood Shores, CA
2 UC Davis School of Medicine, Davis, CA

Background: The need for terminology systems in healthcare has long been established (1). However, to be useful in computer environments, controlled terminologies must be made accessible to computer applications. A common approach is to use a network accessible "terminology server". In this paper, we describe an open source terminology server written in the Java programming language and implemented to maximize simplicity, scalability, and cross-platform compatibility.

Design Issues: The jTerm implementation consists of four elements: [1] command/query mechanism, [2] a generic information model, [3] transport protocol, and [4] output format. Command/Query: Any network accessible service or server must have a mechanism by which a computer application can instruct it to perform a task. Ideally, this "command/query" mechanism should be simple, efficient, leverage existing frameworks, and be independent of a particular programming language. We chose to focus on simplicity and usability. Furthermore, we wanted to ensure that the method chosen had an array of existing tools that could facilitate implementation without incurring much cost, or a particular programming language or framework. We implemented the query mechanism as a command string embedded in a standard Web request. This has the added advantage of being simple and easy to implement in most client application environments. The general syntax for the API is:
For example, to get all the next level child concepts of the SNOMED RT concept "Living organism" (ConceptID=4748007), one issues the following HTTP request:
http://<server address>/snomedhandler?Function=getChildren&String=4748007
To obtain concept information (definition string and termID for the preferred term):
http://<server address>/snomedhandler?Function=getConceptByIdConceptID&String=4748007
To obtain term information for the preferred term for this concept (TermID=8916011)
http://<server address>/snomedhandler?Function=getTermByTermID&String=8916011

The data is returned in XML using a simple DTD. Although the query API uses a http:// URL syntax, it is important to note that jTerm was not designed for web browser clients — the HTTP protocol and URL mechanism can be invoked by any application.

Information Model: The information model used in jTerm follows the prevailing concept-oriented view of terminological systems (2). Transport Protocol: We chose HTTP as the Internet connection protocol for jTerm. HTTP is efficient, simple, and well known by most programmers. Most programming languages and integrated development environments (IDE's) include an HTTP component. HTTP is also advantageous when communication must occur through a firewall as port 80 is often allowed. We found this to be a limitation of the National Library of Medicine's Knowledge Sources Server, which uses ports 8042 and 8043. Server Output: jTerm output is provided as an XML document stream. This allows client applications to leverage widely available XML parser implementations to obtain the data returned.

jTerm Architecture: jTerm was developed as a J2EE "middle-tier" application for use with J2EE application servers. The storage is provided by a relational database server, which is accessed through a tier IV JDBC driver. jTerm can be deployed using any J2EE application server, ANSI-SQL92 RDBMS, and tier-IV JDBC driver. The JDBC driver class and database connection parameters are loaded from an editable property file at startup. As a J2EE application, it can be deployed on a wide array of operating systems and servers, thus providing cross-platform compatibility and upward scalability (in higher-end hardware).

Conclusion: jTerm is an open source, multi-platform capable, network accessible terminology server which supports the use of a wide variety of existing terminological systems. When developing a terminology server, consideration should be given to four essential components — query mechanism, underlying information model, transport protocol, and data output format. jTerm respectively implements a URL-based query mechanism, a concept-oriented object model, the HTTP transport protocol, and XML as an output format. jTerm is open source and can be found at http://www.jterm.org.

References: