cPath Architectural Overview

Last Updated: March 1, 2004

Introduction

This document provides an architectural overview of the cPath database and web application. Information about the database schema, data access objects, servlet architecture, configuration, and unit testing is provided.

Tools and Libraries

cPath is currently built with the following open source tools and libraries:

- Apache Tomcat: open source servlet/jsp engine. Information available at: http://jakarta.apache.org/tomcat/.
- Ant: used for the cPath build and deployment process. Information available at: http://ant.apache.org/.
- Castor: Java/XML Binding Framework. Information available at: http://www.castor.org/.
- CVS: used for source control and revision tracking. Information available at: http://www.cvshome.org/.
- JDOM: Java XML API. Information available at: http://www.jdom.org.
- JUnit: used for all cPath unit tests. Information available at: http://www.junit.org.
- Log4j: open source logging framework. Information available at: http://logging.apache.org/log4j/docs/.
- Lucene: open source text indexer; provides cPath full text search functionality. Information available at: http://jakarta.apache.org/lucene/docs/index.html.
- MySQL: Backend database is built in MySQL. Information available at: http://www.mysql.com.
- Struts: open source framework for building web applications. Information available at: http://jakarta.apache.org/struts/.
- Xerces: fast, validating XML Parser. Information available at: http://xml.apache.org/xerces-j/.

cPath Database

The cPath database consists of 8 relational tables. These tables are grouped into four set:

- Import: used for importing new data into cPath.
- Core Entity Tables: used to store core entities, such as interactors and interactions, and internal/external links.
- External Database Tables: used to store information about external databases, such as SWISS-PROT, NCBI, etc.
- Administrative Tables: used for logging and caching purposes.

A. Import Table

The import table contains information about XML/text records which are scheduled for import into cPath. For example, if an administrator wants to import a new PSI-MI file, the XML data is first loaded into the import table, where it is logged, and recorded. From here, the XML document is parsed and chopped into its constituent parts and loaded into the core cPath tables. The import table contains the following structure:

import	
IMPORT_ID	*Primary ID*
DESC	*Record Description*
DOC_BLOB	*Document Text/XML*
DOC_MD5	*MD5 Fingerprint*
STATUS	*Record Status*
CREATE_TIME	*Timestamp Created*
UPDATE_TIME	*Timestamp Updated*
EX_DB_ID	*Reference to External DB_ID*
LO_ID	*Reference to LinkedOut ID*

B. Core cPath Tables

The cPath core consists of three tables: cpath, internal_link, and external_link. The cpath table contains core entities, such as interactors and interactions. Each entity record contains a short name, description and type. Type must be specified as either: PHYSICAL_ENTITY or INTERACTION. Each record also contains an XML document fragment, written in PSI-MI format. To obtain the full information for an entity, the cPath code must extract and parse the specified XML document fragment.

The internal_link table stores links between cPath records. For example, a cPath interaction record will specify bidirectional links between the interaction record and all its interactors. The external_link table records links to external databases, such as SWISS-PROT, NCBI, etc. Information about these external databases is provided in the next section.

The core cPath tables contain the following structure:

cpath Contains core cPath Entities	
CPATH_ID	*Primary ID*
NAME	*Entity Name*
DESC	*Entity Description*
TYPE	*Entity Type*
SPEC_TYPE	*Entity SubType*
NCBI_TAX_ID	*NCBI Taxonomy ID*
XML_CONTENT	*XML Document*
CREATE_TIME	*Timestamp Created*
UPDATE_TIME	*Timestamp Updated*

external_link	
EXTERNAL_LINK_ID	*Primary ID*
CPATH_ID	*Reference to CPATH_ID*
EXTERNAL_DB_ID	*Reference to External_DB_ID*
LINKED_TO_ID	*Linked to Identifier*
CREATE_TIME	*Timestamp Created*
UPDATE_TIME	*Timestamp Updated*

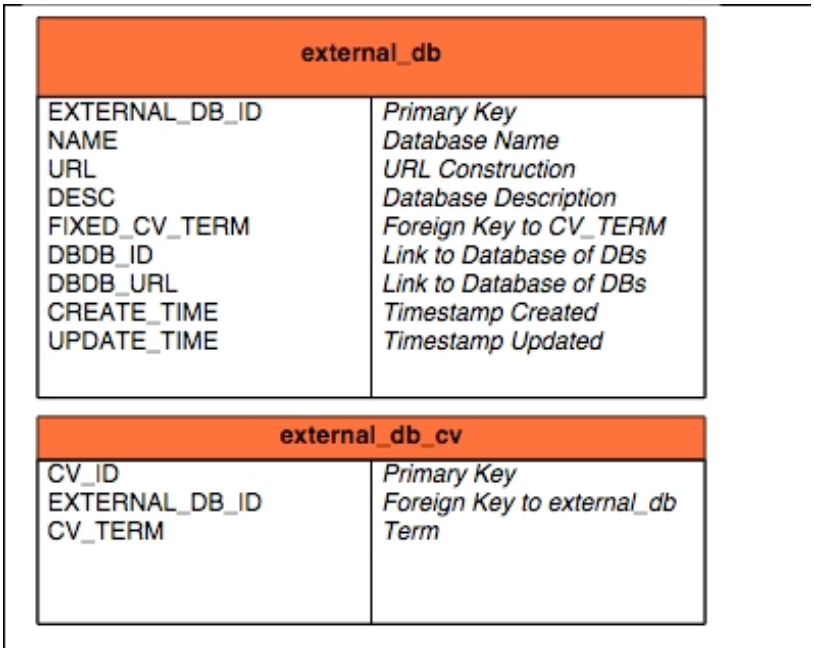
internal_link	
INTERNAL_LINK_ID	*Primary ID*
CPATH_ID_A	*Reference to First Entity*
CPATH_ID_B	*Reference to Second Entity*

C. External Database Tables

The external database tables contain information about external databases and URL construction rules for connecting to those databases. The external_db_cv table contains controlled vocabulary terms, which match a specific database. For example, cPath recognizes: SWISS-PROT, SWP, and SWISSPROT, and all these CV terms point to the same SWISS-PROT record in the external_database table. The external_database table contains URLs for connecting to the external database. The token %ID% is replaced dynamically with the linked to ID. For example, here is a URL for connecting to SWISS-PROT:

http://us.expasy.org/cgi-bin/niceprot.pl?%ID%

The external database tables contain the following structure:



Package Structure

All cPath code is contained in the package: org.mskcc.pathdb. It contains the following sub-packages:

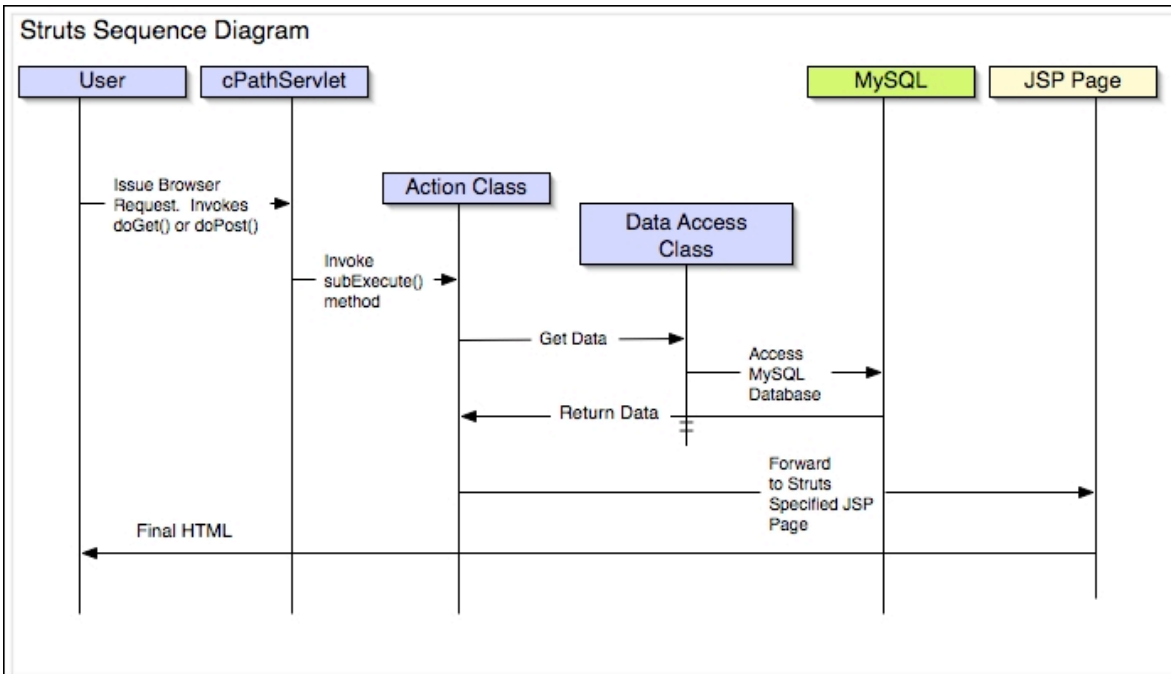
Package	Description
action	Struts Action Classes. Each action class corresponds to a user-initiated action.
controller	Classes for processing and validating web service API requests.
form	All Struts Action Forms. Each form corresponds to an HTML form.
lucene	Classes for connecting to the Lucene Full Text Search engine.
model	JavaBean objects which encapsulate cPath records, such as

	ImportRecord, CPathRecord, etc.
servlet	All cPath Servlets.
sql	All Database Access code.
taglib	All Custom JSP Tags.
task	Long-term Tasks, which require mulit-threading execution.
test	All JUnit Unit Tests.
tool	All Command Line Utilities and Programs
util	Misc. Utility classes, such as an XML validator, Cross-Site scripting filter, etc.
xdebug	Live Debugging/Diagnostics Facility.
xmlrpc	XML-RPC Services for uploading new data into cPath.

Struts/Servlet Architecture

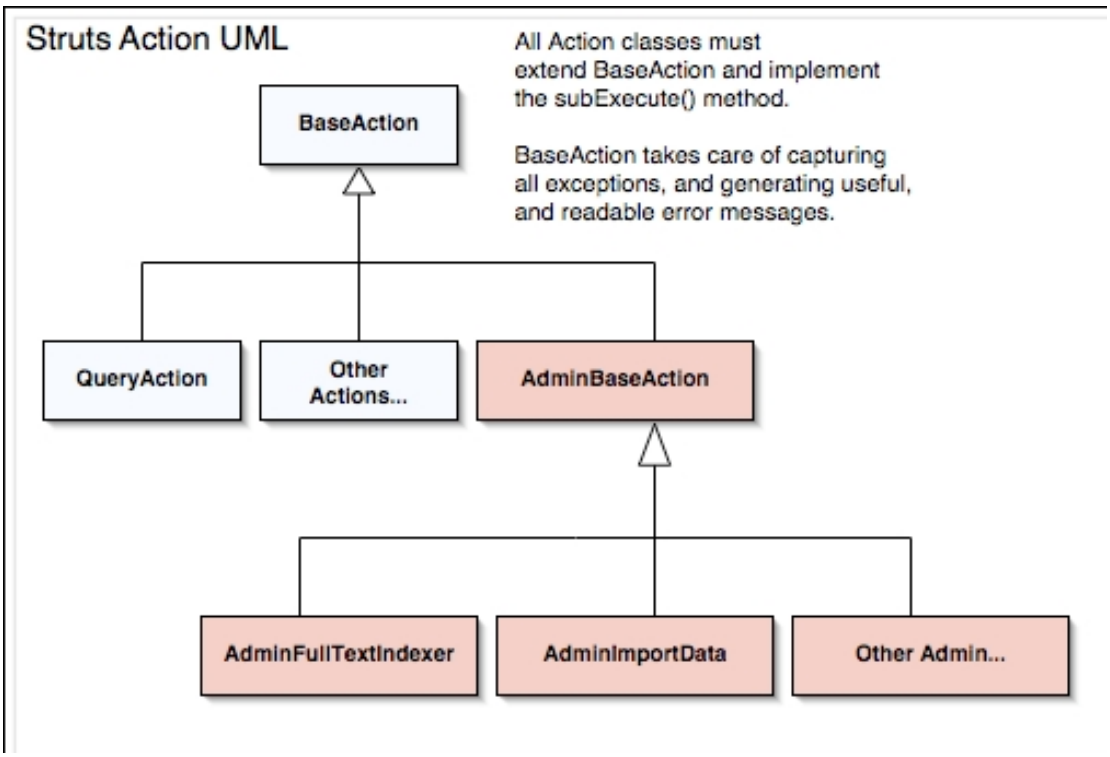
cPath is a Servlet/JSP web application built with the open source Struts Framework. The Struts framework provides a number of built-in advantages, including: clean separation of logic and presentation, form validation, centralization of request handling, and centralization of exception handling. Each time a user initiates a web request, the user request travels through several architectural layers. Each of these layers is summarized in the text and figure below:

- All user requests go through the central cPathServlet class.This class provides a central spot for cPath configuration.
- Based on the URL requested, Struts will check the Struts.xml configuration file, and determine which action class is invoked.
- All database access is centralized in Data Access Objects (DAO).
- Each table in cPath has a corresponding DAO Object. For example, the import table has an ImportDao class.
- All final HTML construction is done in Java Server Pages, and custom JSP tags.



Struts Action Classes

All cPath action classes inherit from the BaseAction class, and all password protected administrator classes inherit from the AdminBaseAction class.



Database Access Objects

Style Sheets, JSP Templates and JSP Custom Tags

Lucene Text Indexer

Web Application Configuration

Unit Testing