Web Client Architecture Guide

Intended Audience

This document was created to review the basic architecture. The author(s) assume that the reader is an experienced software developer or architect who has some experience programming JavaScript and some knowledge of advanced JavaScript topics such as JSON, AJAX, and using JavaScript toolkit (APIs).

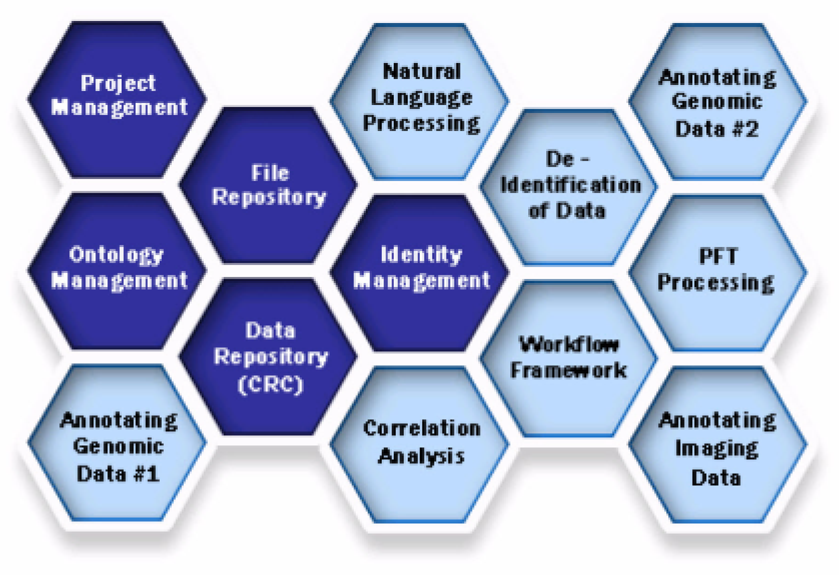
i2b2 Overview

The i2b2 system consists of a front-end client application and a collection of back-end services referred to as the "Hive". The **Hive** is a collection if interoperable services provided by i2b2 "Cells". As a collection, **cells** are loosely coupled and generally do not know about each other, including their relative locality. Instead, activity in the Hive is directed through the use of *Web services* that are invoked manually through the client interface, or automatically by *workflow engines*.

At a minimum a Hive must have a Project Management, Ontology, and Data Repository cells associated with them. Below is a list of some common cells that may be in a Hive:

Cell Name	Cell Code	Description
Project Management	PM	This cell is used to provide user authentication and manage group and roll information.
Ontology Management	ONT	This cell manages the terminology and knowledge information typically used in the hive.
Clinical Research Chart ("Data Repository")	CRC	This cell holds the phenotypic and genotypic data of the hive in a structured format.
Workflow Cell	WORK	This cell is used to process information in steps through various parts of the hive.

An i2b2 Hive



Users cannot directly interact with the Hive's Cells but must go through an i2b2 client application. There are two publicly available clients;

1. Eclipse client: written in Java and operating as a fat-client application installed on the user's desktop computer.

2. Web client: written as a thin-client application that runs in any modern browser.

The only major operational difference is that the fat-client can directly communicate with any Hive Cell whereas the browser-based thin-client can only communicate with the Hive's Cells through a **web proxy cell**.

- ***At this time the supported web browsers are Internet Explorer 6/7/8, Firefox 2/3, and Safari 3.***

Web Client Architecture

Functionality Overview

The **i2b2 Web Client Framework** is a system that is designed and built to operate in a modular fashion. Part of the design effort was to make it simple to develop custom plug-in modules to work within the Framework. There are several design features that were focal points during the design and development of the web client.

Dynamic Loading and Configuration

The web client architecture has been designed to allow for the addition and removal of additional **cells**, **code modules**, and **plug-ins** through the modification of configuration files. All addition and removal of code objects is handled automatically by the ***loader module*** when the user loads the application and logs into a Hive.

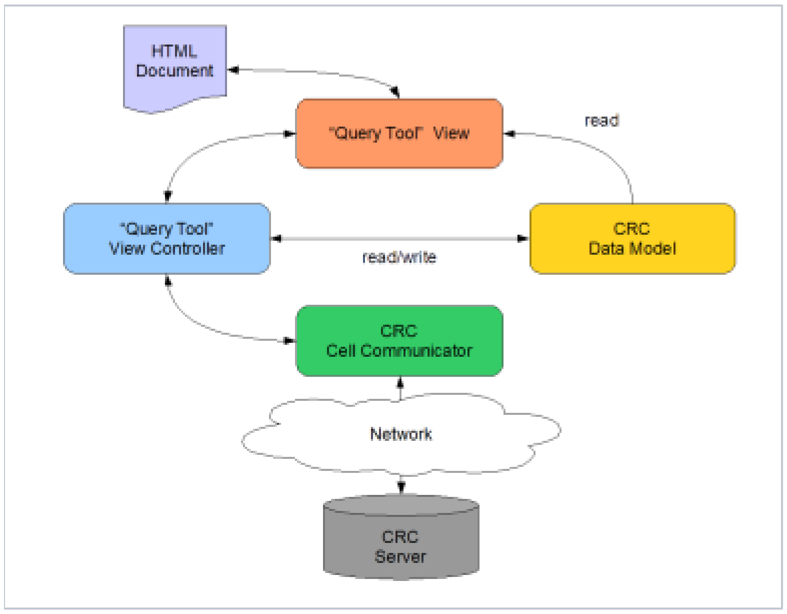
Model-View-Controller Design Pattern

The i2b2 web client was designed to closely follow the **Model-View-Controller (MVC) design pattern**. In the i2b2 web client Framework, the standard Model-View-Controller design is taken one step further with the addition of a "*Communicator*" layer which manages all AJAX communications between the browser and the server.

The general guidelines in determining the placement of code within the MVC pattern is:

Model	JavaScript Object structures containing only data. No executable code.
View	Code that is responsible for rendering, and displaying data to screen. If user interface events need to be captured and processed then those events are attached and detached to DOM objects here. The captured event signals are not processed by code within the view code but passed to the controller code.
Controller	Code which modifies i2b2 data in the model namespace. This code makes calls to objects in the communicator namespace to change data on the Cell's server.
Communicator	Any code which enables communication to a back-end server.

Model-View-Controller-Communicator Design Pattern



The web client framework also makes use of the Router design pattern (especially in the "SDX" communication subsystem).

Standard Data Exchange Subsystem

The state of Web 2.0 technology today is equivalent to the MS-DOS era of desktop computing in that drag & drop, graphical display/manipulation, and inter-application data communication did not exist or existed in extremely simple forms. As a result, applications had to reinvent the wheel each time this type of advanced functionality was needed. Inter-application communication was essentially non-existent.

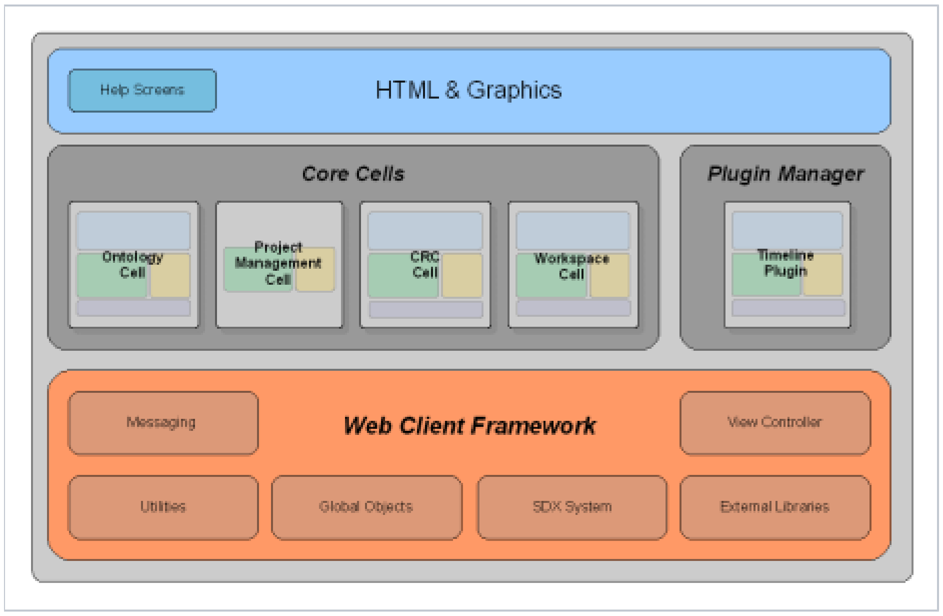
When Microsoft created Windows they created standardized libraries of routines for use by applications. This standardization also allowed for the creation of communication methods to transfer data between different applications. In modern computing, the computer's OS handles all the complexities of data communication internally. However, this functionality ends at the browser window. For multiple web applications/components to exist within a browser window and effectively communicate with one another, a standardized communication channel must be created by the parent framework. This is accomplished in the i2b2 web client through the **Standard Data Exchange subsystem** (or "SDX Subsystem") that is used by all cells wishing to allow drag & drop functionality to, or from, their visual elements.

Web Client Architecture

The web client system is made up of HTML for presentation and JavaScript code. The architecture is built in a modular for having a large base of shared code making up the lattice of the client's framework, and various code modules (core cells and other plug-ins). Within the web client framework are a variety of sub systems and reusable code:

Messaging	This is the communications message sniffer subsystem
Utilities	Various cross-browser utilities used by the framework and code modules (including plug-ins)
Global Objects	Base classes such as plug-in module, view controller, SDX data package, Scoped_callback class
SDX System	This is the SDX subsystem for inter-browser object communication (For more information on the SDX please see the document called "*Web Client SDX System*")
Comm System	Standardized subsystem used to communicate with the i2b2 Hive (For more information on the comm system, please see the document called *"Web Client Communicator Subsystem*")
External Libraries	Yahoo User Interface (YUI), PrototypeJS, etc.
View Controller	Highest level view controller used to manage the core cell's view windows

Client Architecture within the Web Browser

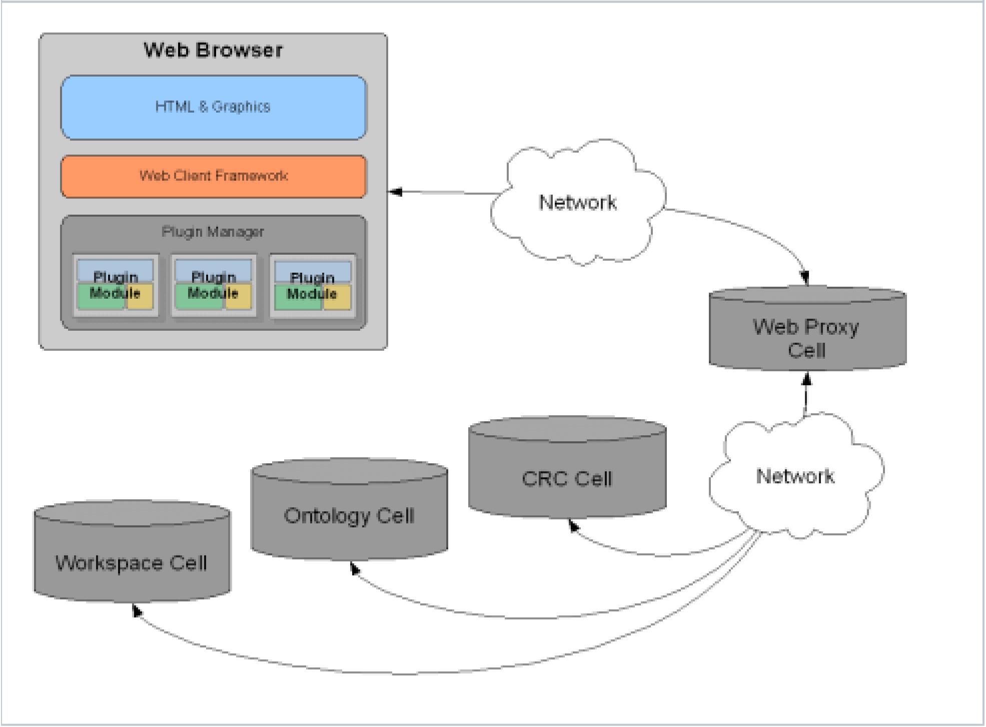


Web Proxy Communication with a Hive

All modern web browsers follow the a universal security standard which does not allow JavaScript code to directly communicate with any server that is not the exact same base URL that was used to load the web page. This behavior is critically important in preventing a hacker's code from offloading your personal information to a random IP overseas; it also prevents web-based applications from directly engaging an enterprise's SOA web services.

The solution to this problem is to use a web proxy that forwards the request to the correct service and then returns the response back to the browser. This extremely simple cell enables the browser based web client to participate fully with the i2b2 Hive.

Overview of Web Client's Communication



Standardized Communication with the i2b2 Hive

Starting with Web Client v1.4, all communication objects have been rewritten to utilize a standard communication object. This standardization greatly simplifies extending new i2b2 Hive functionality to the Web Client. In addition, the standardized communication objects allow the Web Client to follow the software engineering principles of code reuse.

For more information on the communication process please see the document called "*Web Client Communicator Subsystem*".