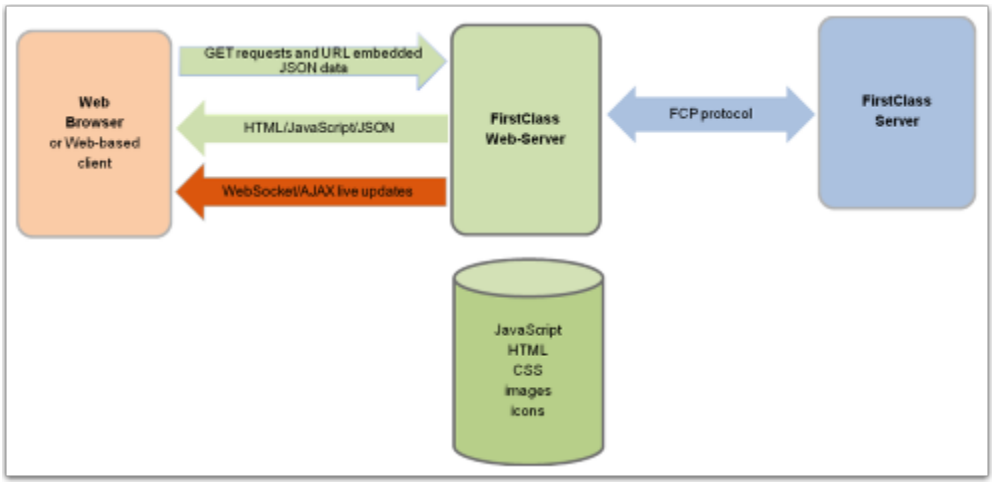
FirstClass 12 API

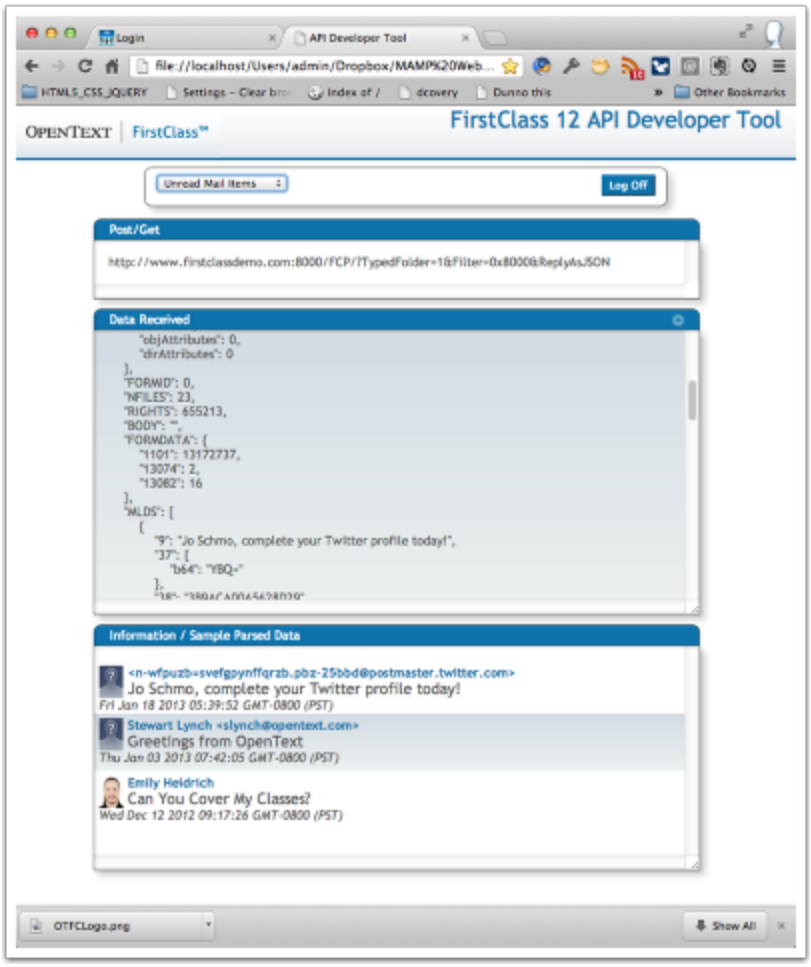
Introduction

An application programming interface (API) is a protocol intended to be used as an interface by software components to communicate with each other. The FirstClass Web Services server (FCWS) is a Python implementation of a web server that facilitates access to FirstClass server data and functionality. The FirstClass web client (or any other web-based client) uses HTML 5.0 to communicate with the FirstClass web server. Requests issued to the FirstClass web server are translated into the FirstClass-proprietary protocol FCP and sent to the FirstClass server for processing. The FirstClass server responses are also encapsulated in the FCP protocol, which is then processed by the FirstClass web server and subsequently sent back to the web client. The API is documented and web developers can use this API to create applications that can present secure, authenticated data within their own applications.



The FirstClass 12 API Developer Tool

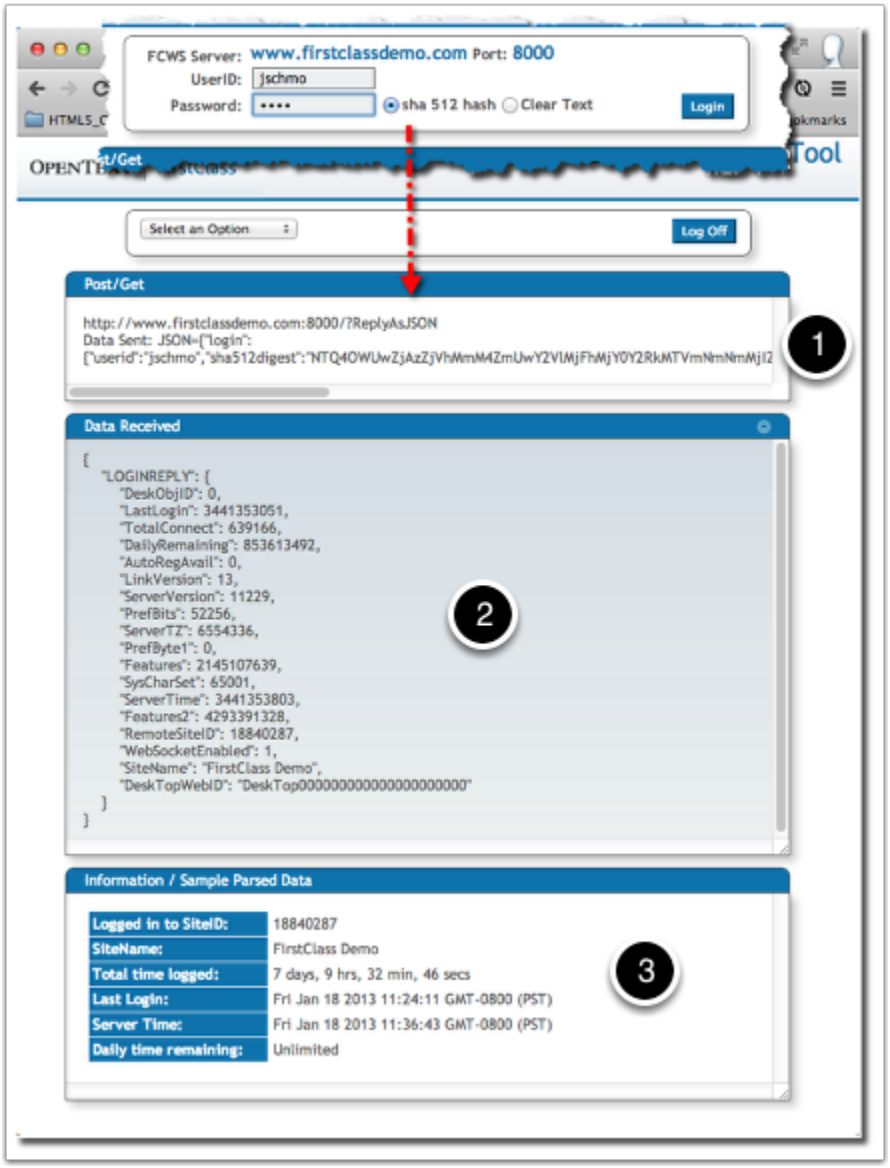
A developer tool has been created to help web developers understand the API documentation and to provide some examples on how to retrieve and parse information returned from an API call to the FirstClass server.



Logging In

As mentioned, the current implementation of the API requires an authenticated login. The tool allows you to choose between a base64 sha 512 hash encrypted or a clear text password. After clicking the Login button, you will see the following:

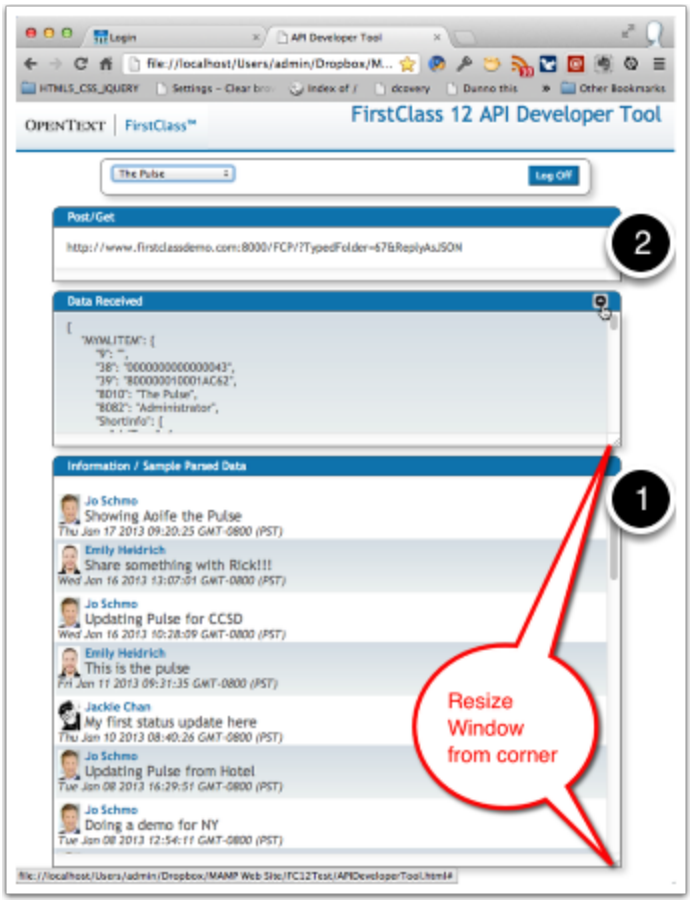
1. The URL that is sent to the server along with any POST data that accompanies the request.
2. The data that is returned from the server in its raw JSON form.
3. A sample of how a web developer may wish to parse and display the returned data.



The Interface

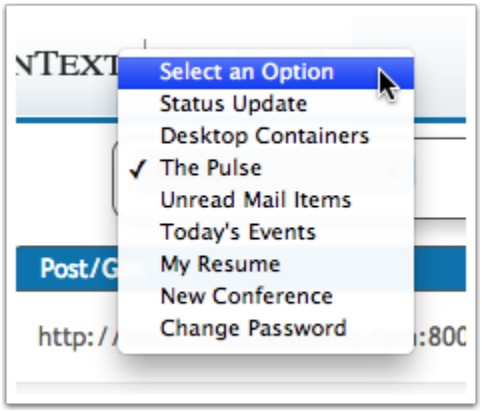
The tool has a simple interface that allows you to expand and shrink the Data Received and Information / Sample Parsed Data sections so you can better view the content.

1. Each window has a resizing corner that you can use to increase or reduce the height.
2. You can dismiss the Data Received window entirely by clicking on the collapse button. This is a toggle so clicking again will expand it back to its previous size.



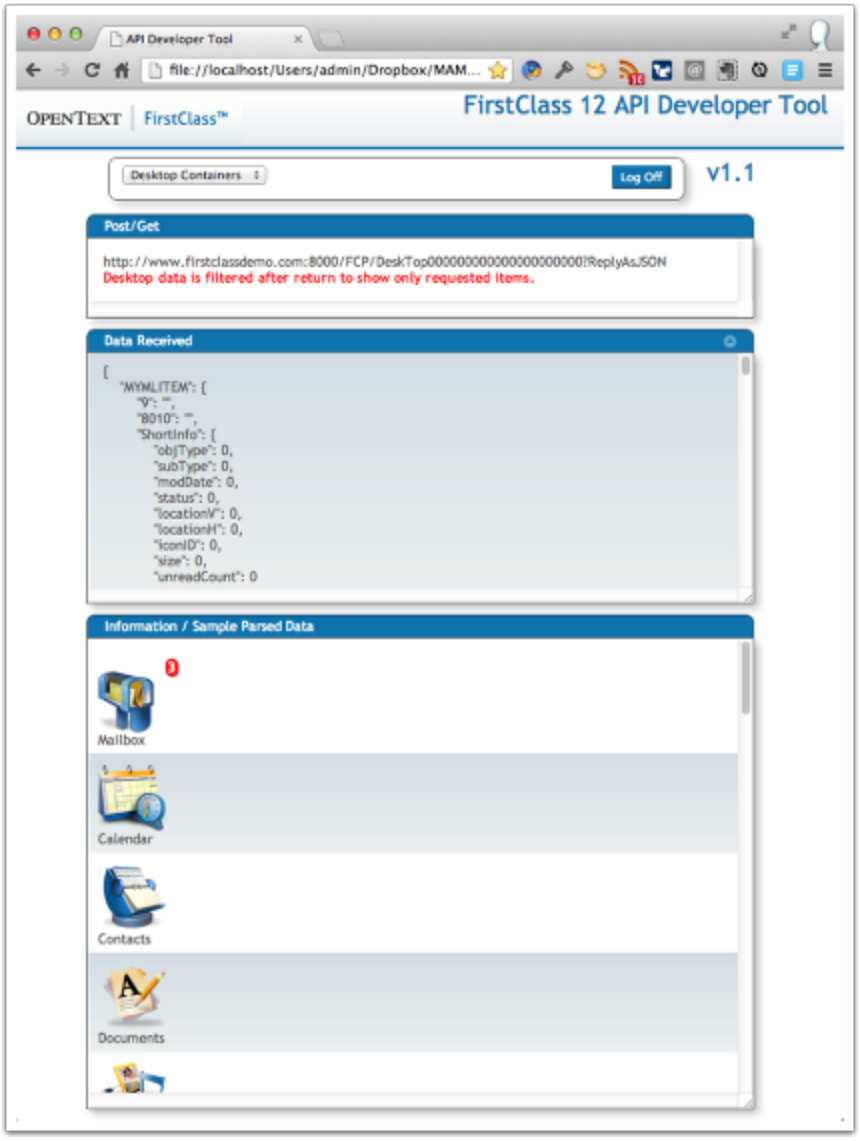
Options

A number of different options are available once logged in. An effort has been made to provide developers with a range of examples that might prove useful if they are trying to incorporate FirstClass content within their own web applications.



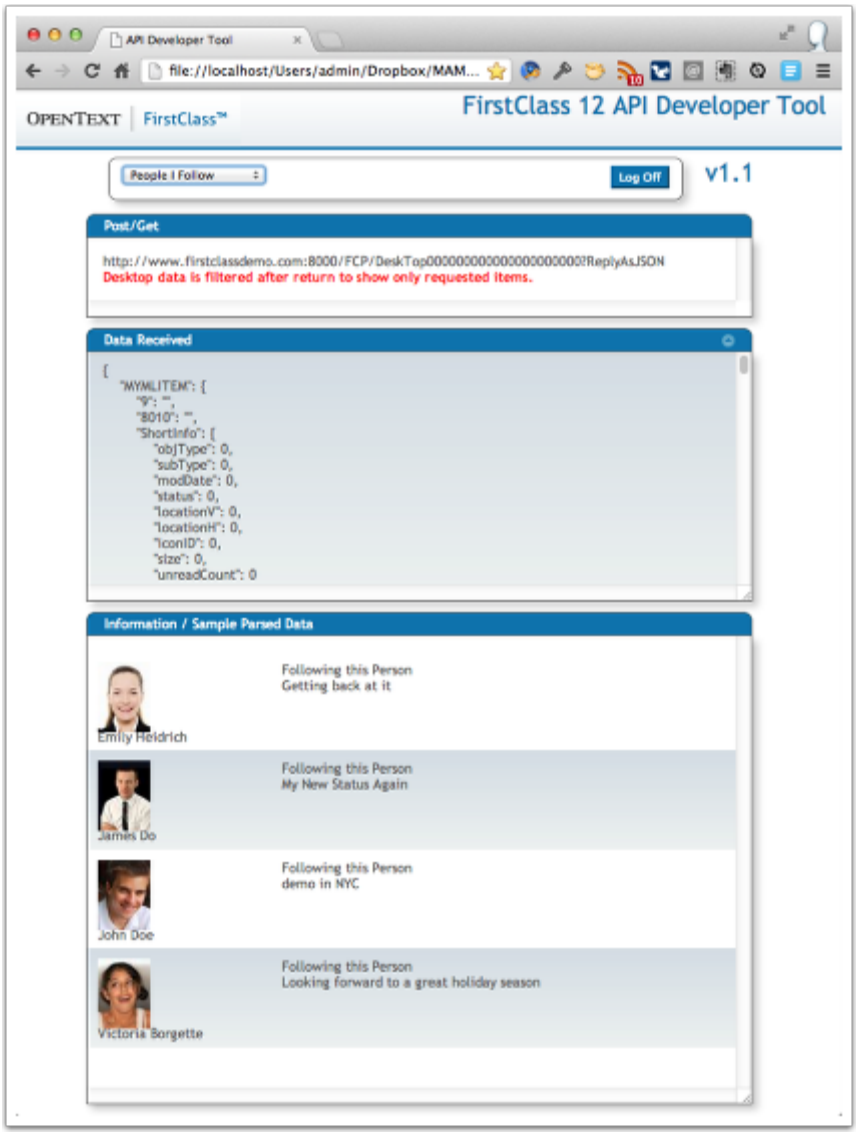
Desktop Containers

Selecting Desktop Containers will retrieve and display all containers on the user's desktop. This starts with the general call to open the user's dektop, fetching everthing that is stored on the user's home screen This includes items that are invisible to the user. In order to display containers only, the developer will need to filter the JSON results to only display the containers



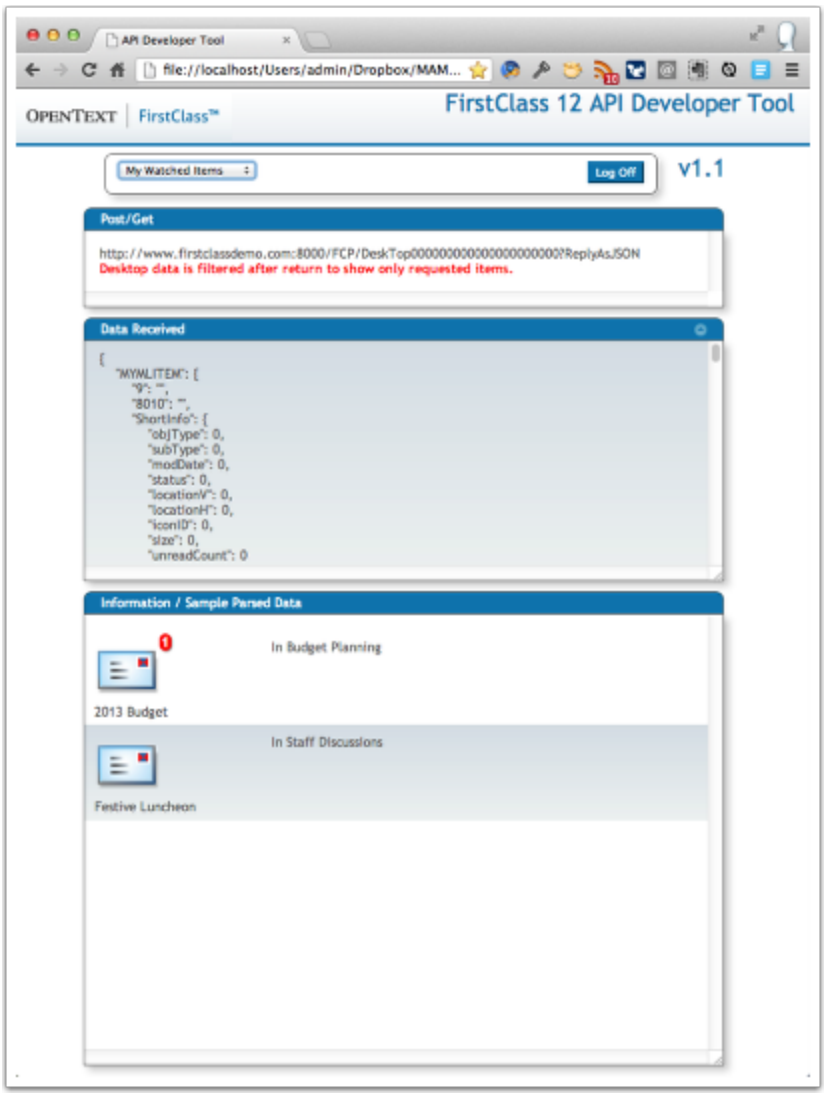
People I follow

As mentioned above, retrieving the desktop retrieves everthing that is stored on the user's home screeen including items that are invisible to the user. To retrieve the People I follow, the desktop is retrieved and the developer must filter the results to display only people.



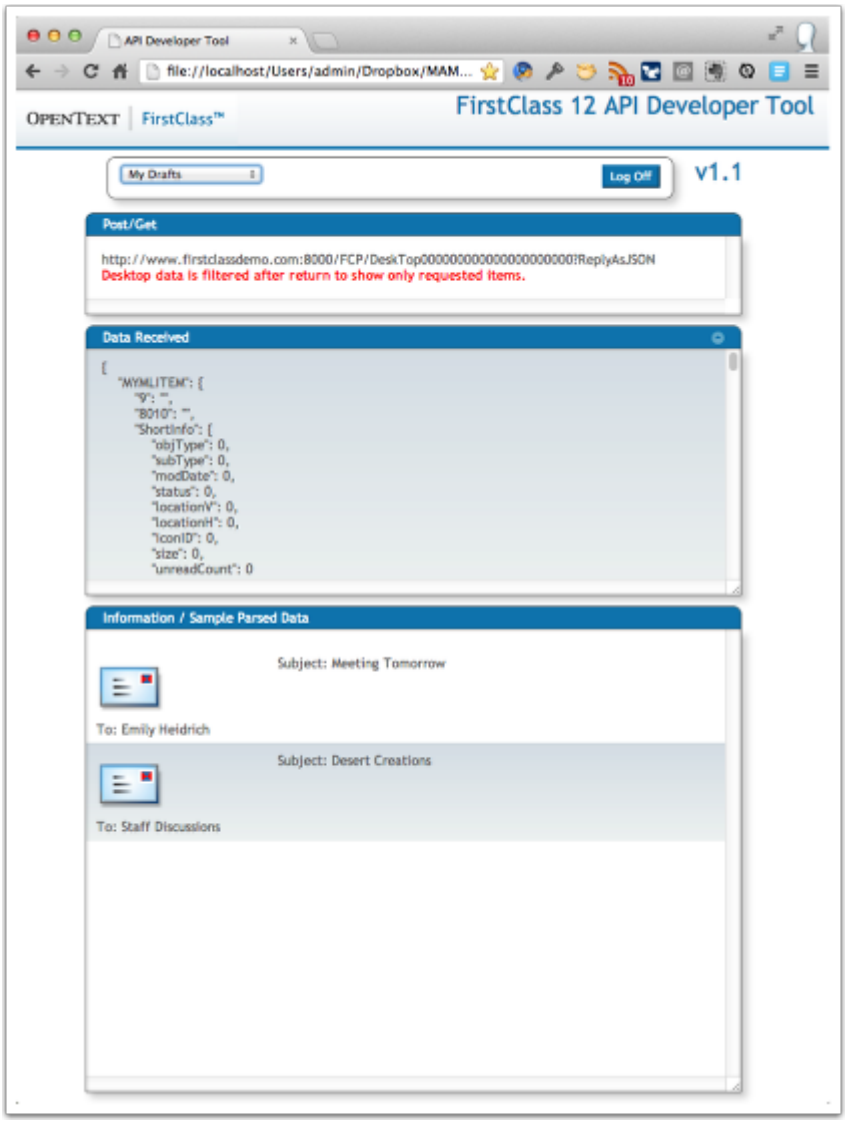
My Watched Items

Similar to other desktop items, to retrieve My Watched Items, the desktop is retrieved and the developer must filter the results to display only the watched items.



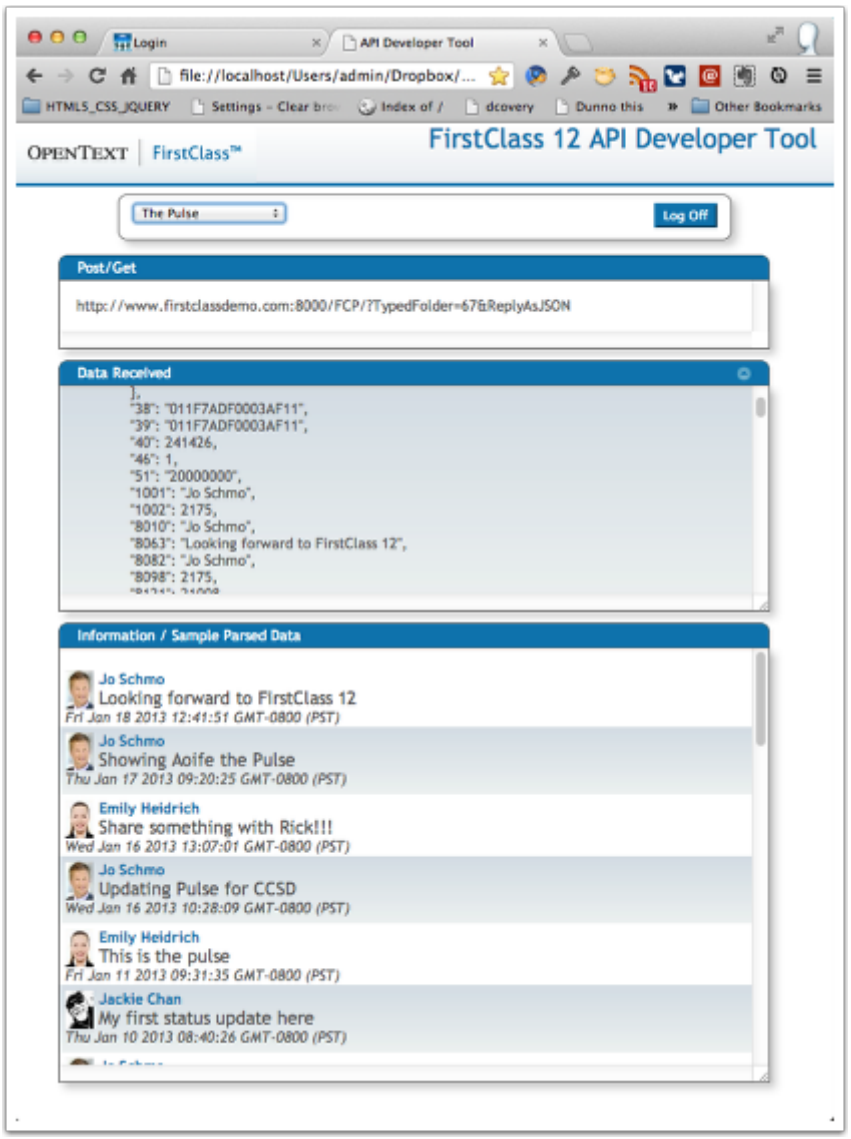
My Drafts

Similar to other desktop items, to retrieve My Drafts, the desktop is retrieved and the developer must filter the results to display only the draft items.



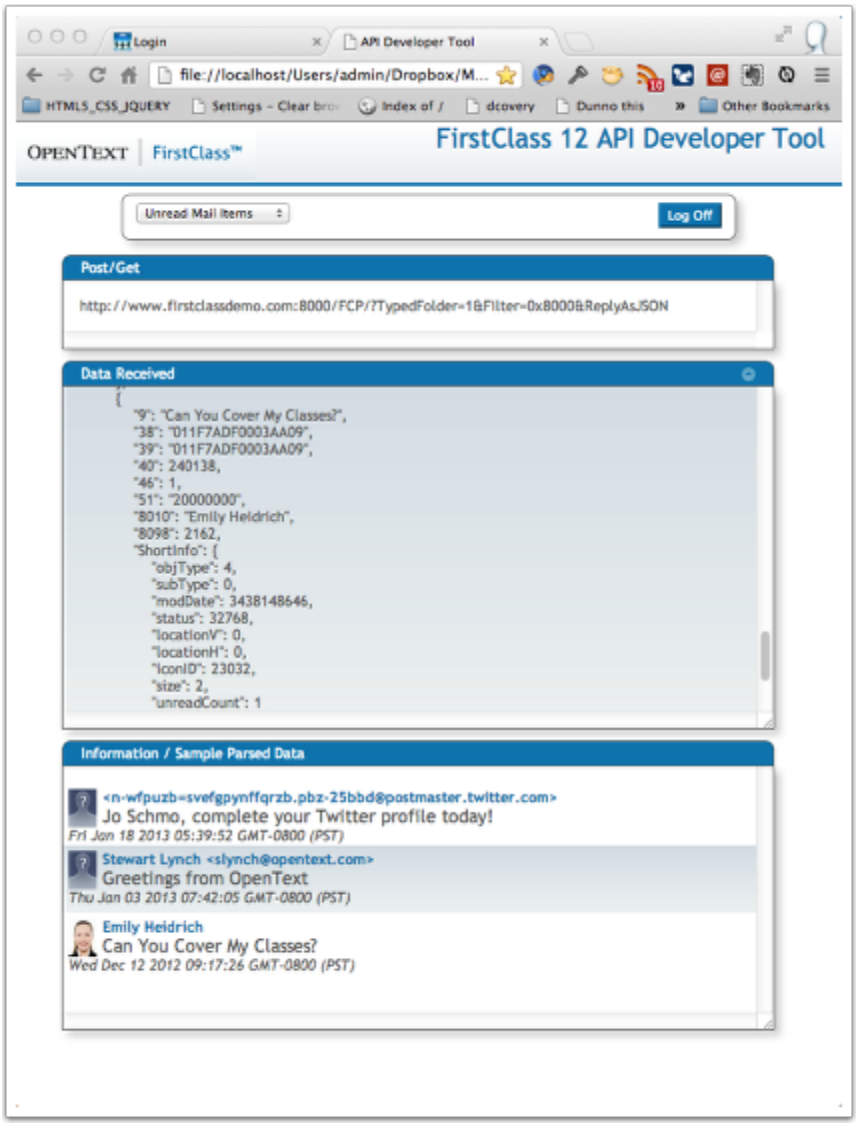
The Pulse

The Pulse returns and displays the current pulse entries for the logged in user.



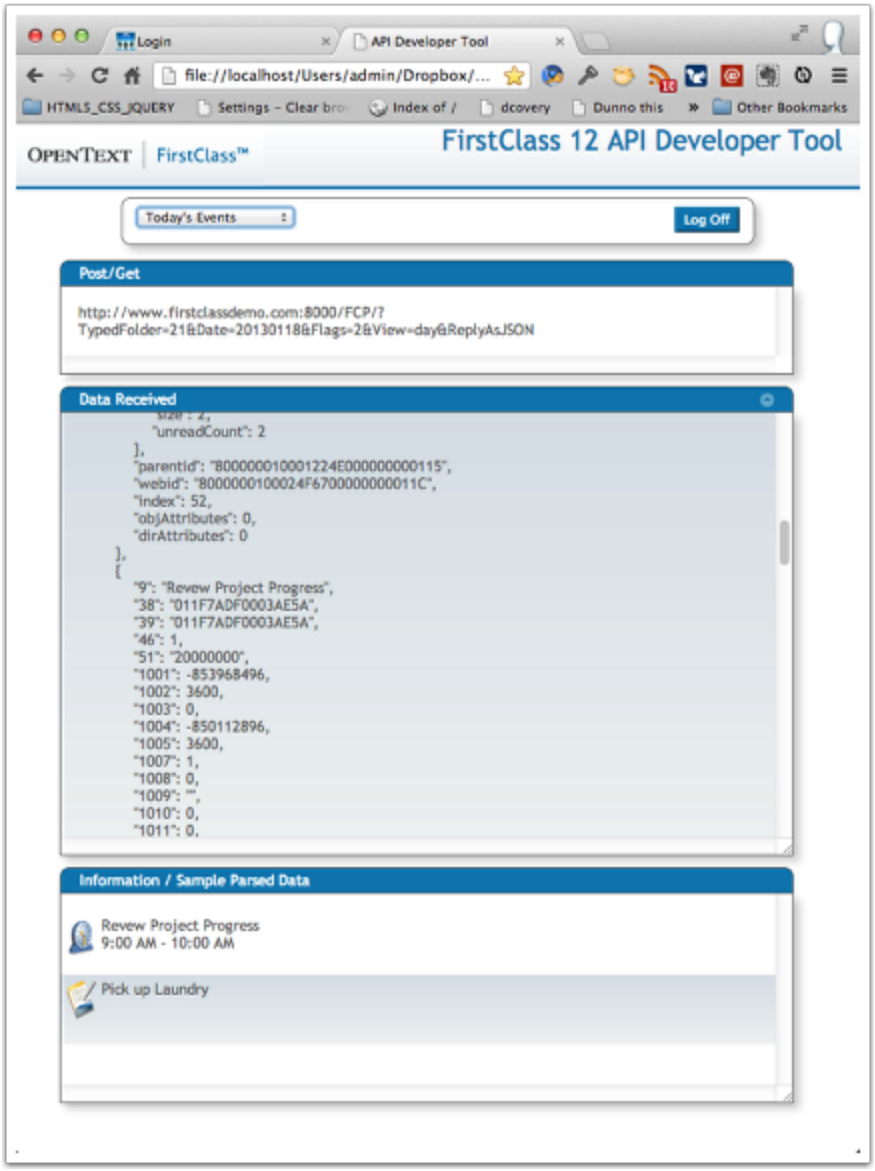
Unread Mail Items

This Unread Mail Items option will list all of the unread mail items for the currently logged in user.



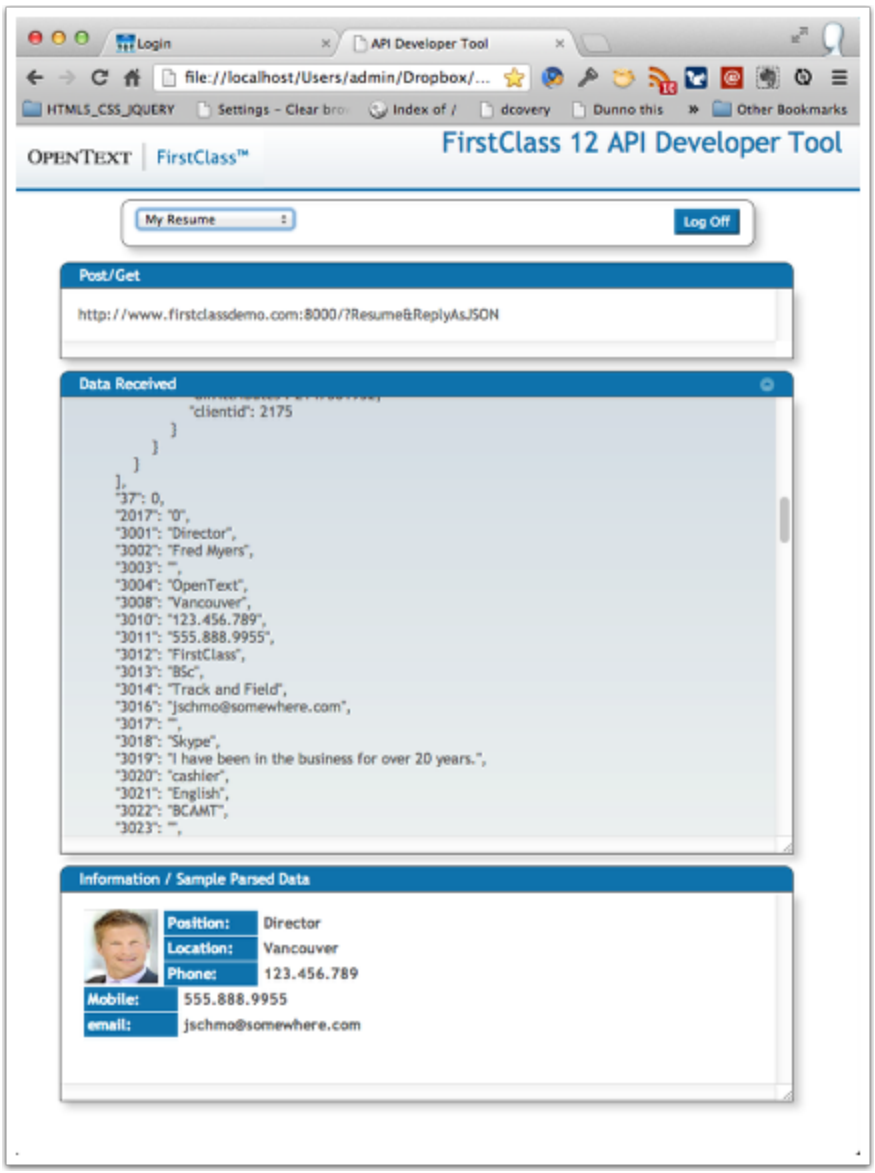
Today's Events

Today's Events returns a list of both events and tasks that are scheduled for the current day.



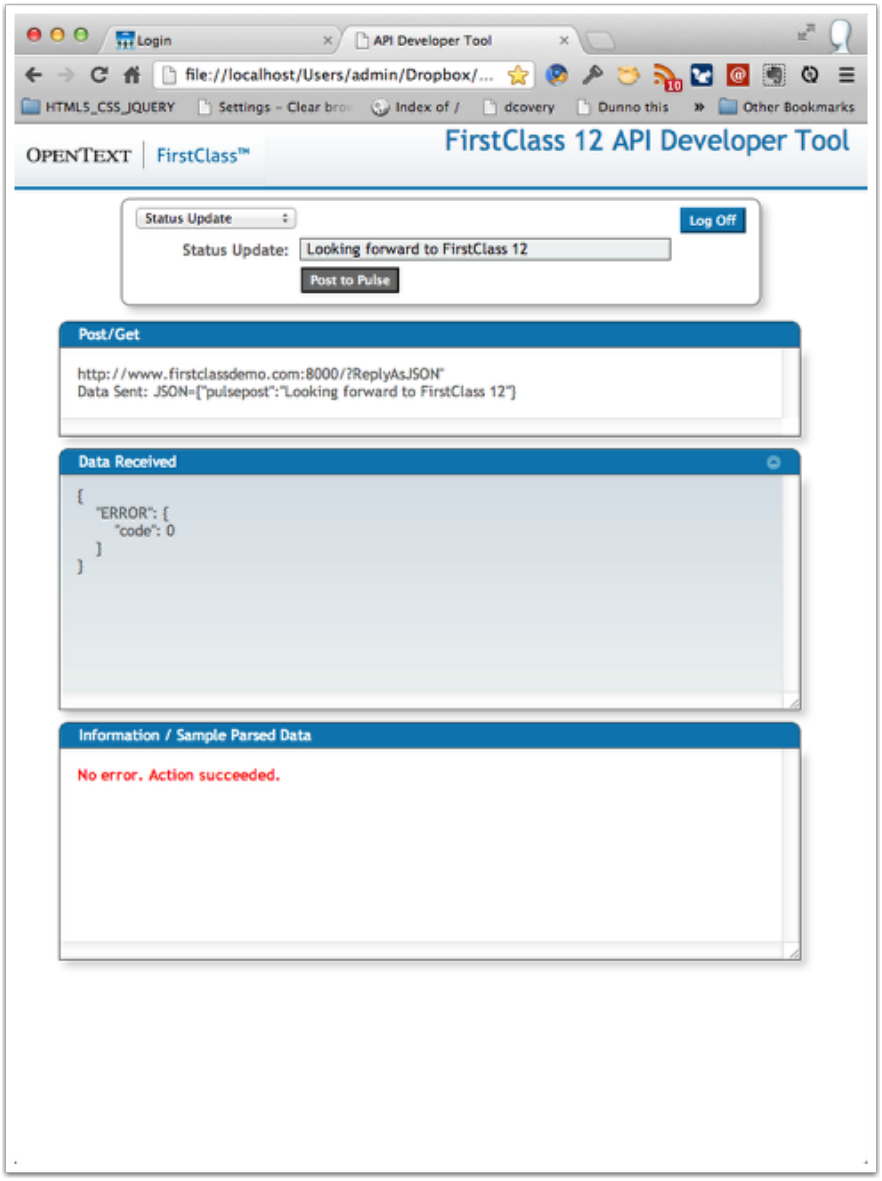
My Résumé

My Résumé returns the user's profile résumé and presents a sample of the retrieved information.



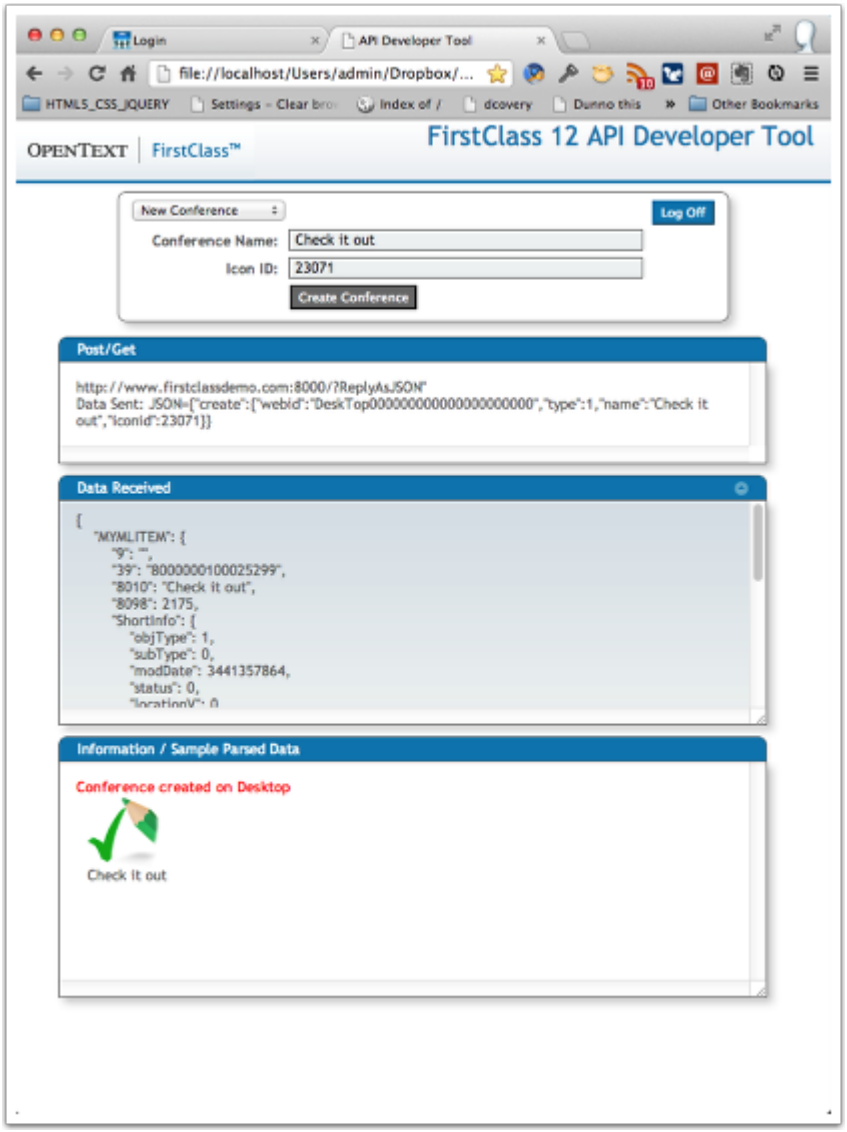
Status Update

**The Status Update is an example of a POST to FirstClass that will update the user's status and post to the Pulse. This requires the completion of a field first before submission.
A subsequent retrieval of the pulse will show that the entry has been submitted.**



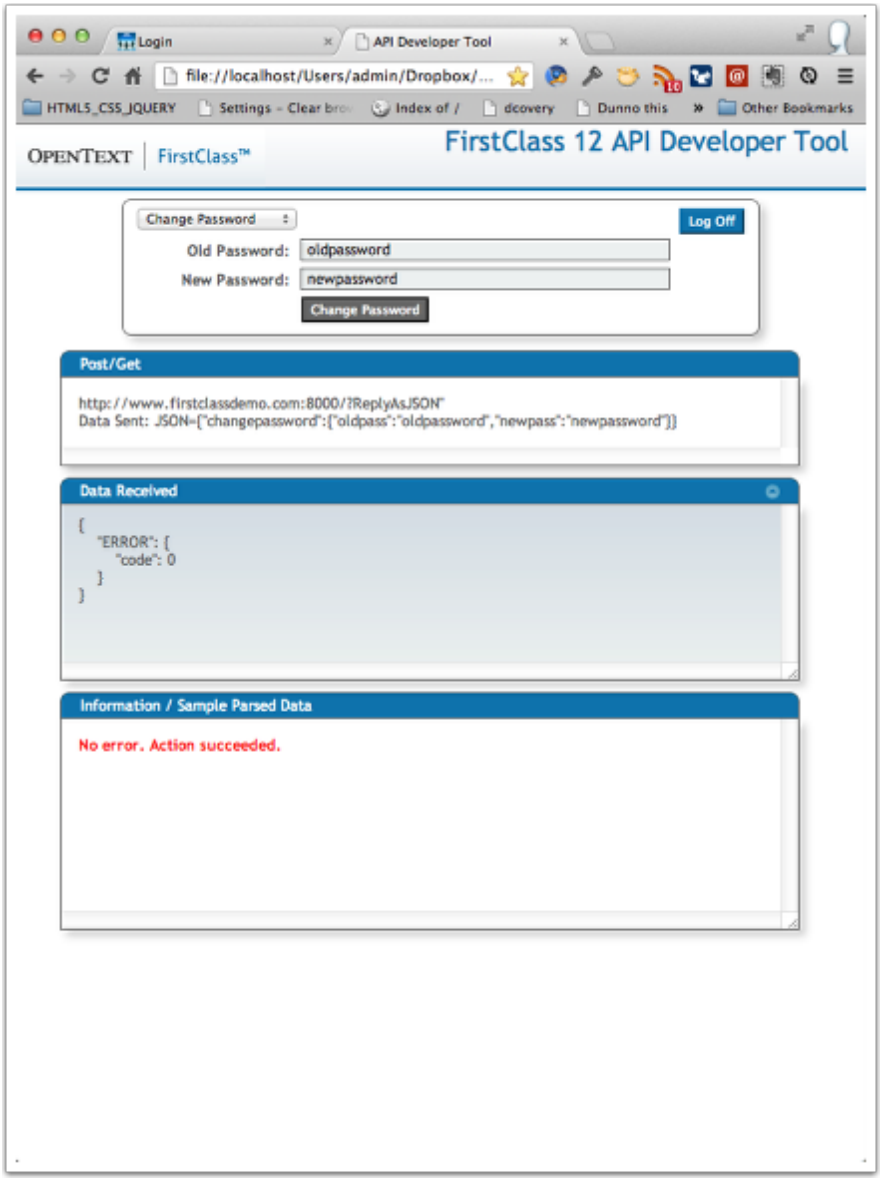
New Conference

The New Conference option is another example of a POST that requires a name and iconID. This creates the conference on the user's desktop and displays the resulting container and icon in the Information window.



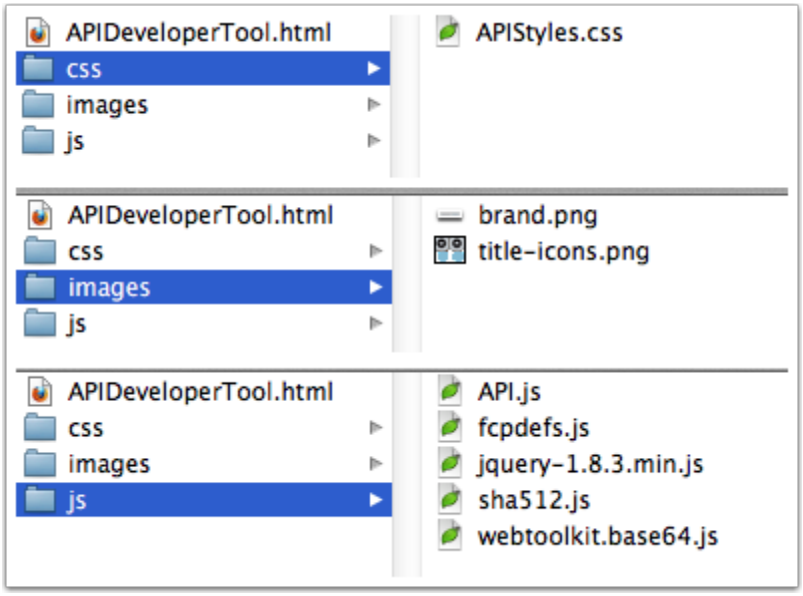
Change Password

The final example, Change Password is another POST example that requires the old and new password to be submitted (POSTED) to the server. An error code of 0 indicates that the action was successful.



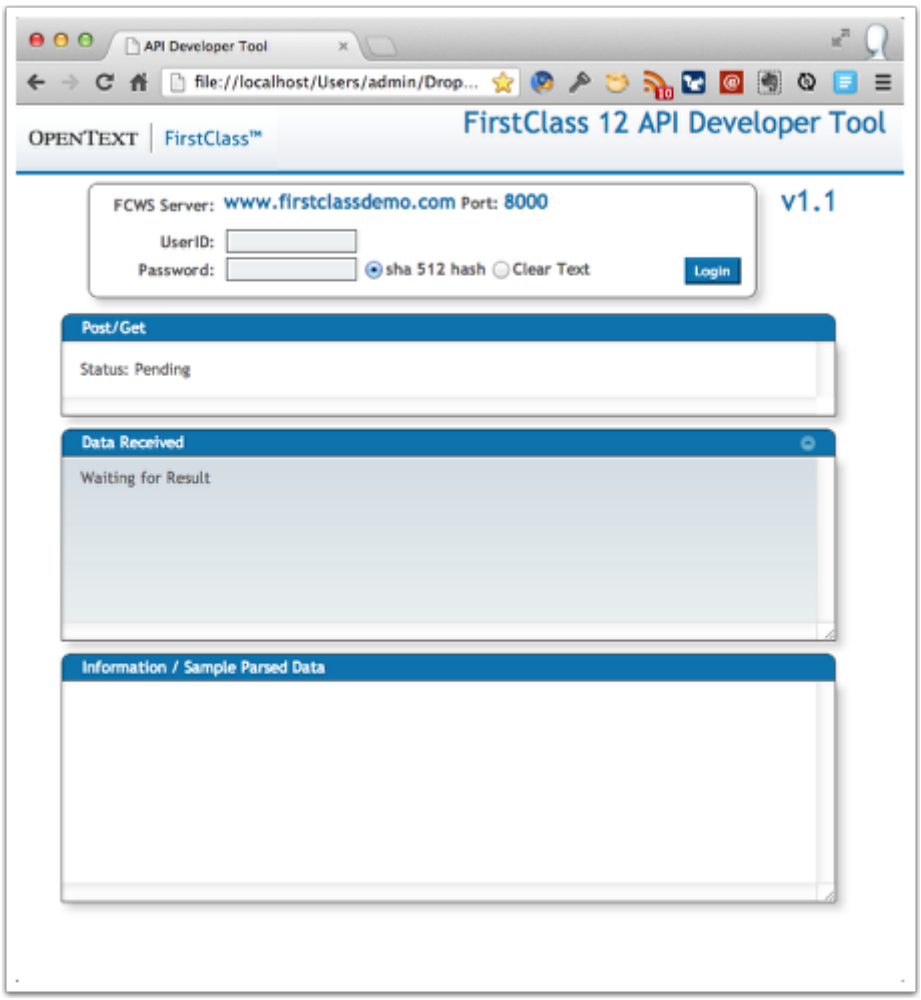
Development Tool Files

The development tool consists of a APIDeveloperTool.html file along with the API.js javascript file that is the core to understanding how the application functions. There is also an images folder and a supporting CSS file for the presentation of the parsed data along with some javascript libraries used by API.js to help interpret and present the returned data.



APIDeveloperTool.html

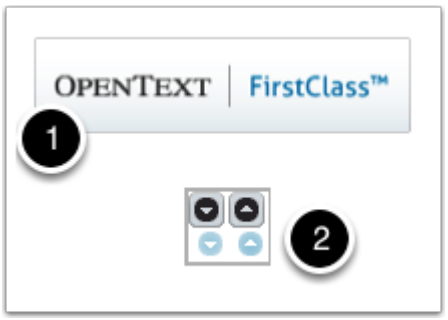
This is the presentation layer that presents the tool to the user in the web browser.



images folder

Consists of:

1. The top logo for the html page
2. A 4 image sprite that is used for the Data Received window collapse/expand button



CSS File

This is used for both styling and layout. Selectors are arranged by category to assist the developer in following through the code.

```
/* Login and option fields
-------------------------------------------------------------------------*/
#LI_fieldset {
  font-weight: bold;
  margin-left: auto;
  margin-right: auto;
  width: 525px;
  border: 1px solid #555555;
  -webkit-border-radius: 8px;
  -moz-border-radius: 8px;
  border-radius: 8px;
  -moz-box-shadow: 5px 5px 10px #cccccc;
  -webkit-box-shadow: 5px 5px 10px #cccccc;
  box-shadow: 5px 5px 10px #cccccc; }

.LoginEntry {
  float: left; }

.LI_input, .option_input {
  font-family: "Trebuchet MS", Helvetica, sans-serif;
  font-size: 10pt;
  color: #000;
  background: #ECF0F0;
  border: 1px solid #555555;
  padding-left: 5px; }

#postFields1, #postFields2 {
  margin-bottom: 2px;
  clear: both; }

#postLabel1, #postLabel2 {
  width: 10em; }

#LI_Login {
  color: #FFF;
  background: #0073ad;
  border: 2px outset #0073ad;
  float: right; }

#actionButton {
  color: #fff;
  background: #666;
  border: 2px outset #222222;
  margin-left: 13em; }

/* Data fields for content display
-------------------------------------------------------------------------*/
.dataTitle, #receivedHead {
  border: solid 1px #555555;
  -webkit-border-top-left-radius: 8px;
  -moz-border-top-left-radius: 8px;
  -webkit-border-top-right-radius: 8px;
  -moz-border-top-right-radius: 8px;
  border-top-left-radius: 8px;
  border-top-right-radius: 8px;
  color: white;
  margin-left: -1px;
  padding: 3px 16px;
  width: 600px;
  background-color: #0073ad; }
```

JavaScript files

There are 5 javascript files included with the tool. The main one is the API.js along with 4 javascript libraries as described below.

API.js

This is the core file to understanding how the development tool works. Using this file along with the API documentation, experienced developers should be able to interpret and build their own rich web applications to retrieve from and modify data on a FirstClass server.
Every attempt has been made to modularize the code into functions so that the developer can reuse it within their own solutions.

```
function get_or_Post(){
    var dataToGet = $('#GetItem').val();
    var GetURL = '';
    var dataSent = '';
    switch(dataToGet) {
        case "Select":
            break;
        case "UpdatePulse":
            GetURL= ServerAddr + "/?ReplyAsJSON";
            dataSent = 'JSON={"pulsepost":"' + $("#postInput1").val() + '"}';
            break;
        case "DesktopContainers":
            GetURL=ServerAddr + "/FCP/DeskTop0000000000000000000000?ReplyAsJSON";
            break;
        case "Pulse":
            GetURL = ServerAddr + "/FCP/?TypedFolder=67&ReplyAsJSON";
            break;
        case "UnreadMail":
            GetURL= ServerAddr + "/FCP/?TypedFolder=1&Filter=0x8000&ReplyAsJSON";
            break;
        case "TodayEvents":
            var d = new Date();
            var y = d.getFullYear().toString();
            var m = (d.getMonth() + 1).toString();
            var dy = d.getDate().toString();
            var today = y + ("0" + m).slice(-2) + ("0" + dy).slice(-2);
            GetURL= ServerAddr + "/FCP/?TypedFolder=21&Date=" + today + "&Flags=2&View=day&ReplyAsJS
            break;
        case "MyResume":
            GetURL= ServerAddr + "/?Resume&ReplyAsJSON";
            break;
        case "NewConference":
            GetURL= ServerAddr + "/?ReplyAsJSON";
            dataSent = 'JSON={"create":{"webid":"DeskTop0000000000000000000000","type":1,"name":"'+ $(
            break;
        case "ChangePassword":
            GetURL= ServerAddr + "/?ReplyAsJSON";
            dataSent = 'JSON={"changepassword":{"oldpass":"' + $("#postInput1").val() + '","newpass'
            break;
    }
    // if the 'GetItem' is in this list, it is a 'Post' action, otherwise it is a 'Get' action
    var postItems = ["NewConference", "ChangePassword", "UpdatePulse"];
    if (postItems.indexOf(dataToGet) >= 0) {
        // This is a 'Post' so it requires data to send
        $('#SentData').html(ServerAddr + '/?ReplyAsJSON"<br />Data Sent: ' + dataSent);
        createObject(GetURL,dataSent,dataToGet);
    }
    else {
        // This is a 'Get'
        $('#SentData').html(GetURL);
        getObject(GetURL,dataToGet);
    }
}
```

Library javascript files

The 4 library files are included and are required as follows:

- jquery-1.8.3.min.js - Though the tool does call to the Google jquery CDN to retrieve this library, it is included as a fallback should that not be available.
- sha512.js and webtoolkit.base64.js are used to encrypt the password into a base64 sha512 digest for logging in securely to the server.
- fcpdefs.js - a FirstClass library containing a number of functions and global variables that can be useful to the developer. In particular, the date parsing functions are used by the tool.