OZONE Widget Framework
Introduction and Quick Start Tutorial

February 17, 2012
“A framework for visually organizing and laying out lightweight web applications (widget) within a user’s browser.”
Government Open Source Software (GOSS)

Computer software available in source code form, for which the source code and certain other rights normally reserved for the development agency, are provided broadly.

- Released under the DoD Community Source Usage Agreement Version 1.1 by the Department of Defense.

- Not a pure FOSS (Free Open Source Software) project such as those covered by popular OSS (Open Source Software) licenses including GPL, LGPL, MIT.

- Dissemination is limited to U.S. Government Agencies and their contractors in accordance with provisions of DoD Directive 5230.25.
(Sec. 923) Requires the Director of the Defense Information Systems Agency to implement a mechanism to publish and maintain on the public Internet specified information on and resources for the Ozone Widget Framework in order to permit individuals and companies to develop, integrate, and test analysis tools and applications for use by DOD and elements of the intelligence community. Requires such Director to encourage and foster the use of such Framework by the computer industry and commercial information technology vendors.
OZONE Widget Framework

OZONE Widget Framework (OWF)
- Framework for visually organizing and laying out lightweight web applications (widgets) within user's browser.
- Provides infrastructure services to facilitate development of workflows and presentation-tier application integration.

OZONE Marketplace (OMP)
- Thin client registry of applications and services similar to commercial application stores.
- Can be utilized with OWF or as an independent application.
OZONE Widget Framework
OZONE Widgets

- Typically developed as a standard Java WAR.
- Exist within OZONE Widget Framework as IFrame.
- By referencing hosted OWF library file application may leverage OWF functionality.
- Alternative deployments may include static HTML pages on external web servers.
- Alternative deployments may include server specific solutions such as .NET solutions on IIS, PHP applications on Apache through mod_php, etc…
OZONE Widgets

<table>
<thead>
<tr>
<th>OZONE Widget API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventing API</td>
</tr>
<tr>
<td>Preferences API</td>
</tr>
<tr>
<td>Logging API</td>
</tr>
<tr>
<td>Widget Launcher API</td>
</tr>
<tr>
<td>Drag and Drop API</td>
</tr>
<tr>
<td>Chrome API</td>
</tr>
</tbody>
</table>

- Primarily developed with the Dojo Javascript library. Accessible via the `owfdojo` object.
- All APIs are available by referencing the hosted OWF library Javascript file.
Tutorials assumes the following:

- The developer will be developing an OZONE widget packaged as a Java WAR.
- The widget will be deployed directly to an instance of Jetty also running the OWF server application. OWF is accessible at the url:

  `https://localhost:8443/owf`

- The widget will utilize the Maven build system on top of Java 1.6+
Anatomy of an OZONE Widget

- Looks exactly like a servlet-based Java WAR project.
- Developed exactly like a servlet-based Java WAR project.
- If the developer is especially clever the widget can function both within and external to the OWF application.
- Can be initialized utilizing the standard Maven webapp-javaee6 archetype.
  
  or

- In Netbeans the developer need only create a new Maven Web Application project.
Example EmptyWidget Project

- Bare-bones skeleton project.
- Contains settings to build and deploy both project WAR and required Jetty context files to a local Jetty instance.
- Deploys to Jetty as `https://localhost:8443/EmptyWidget`
- Can be easily imported into Netbeans or built using command line tools.
  
  **Normal Build**
  
  `mvn clean install`
  
  **Build and Deploy**
  
  `mvn clean install -Pdeploy-jetty`
<?xml version="1.0" encoding="UTF-8"?>
<Web-app version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee"
  <display-name>${project.name}</display-name>
  <context-param>
    <description>OWF Context Path</description>
    <param-name>owf-context-path</param-name>
    <param-value>${owf.context.path}</param-value>
  </context-param>
</Web-app>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<title>Example Empty OWF Widget</title>

<% String owfContextPath = application.getInitParameter("owf-context-path"); %>

<link rel="stylesheet" href="<%= owfContextPath %>/css/widgetContents.css" type="text/css"/>
<script type="text/javascript"
    src="<%= owfContextPath %>/js/owf-widget.js"></script>

<link rel="stylesheet" href="css/style.css" type="text/css"/>
<script type="text/javascript" src="js/index.js"></script>
</head>
<body>Develop Me! </body>
</html>
1. Click Launch Menu Button

2. Double Click Widget Edit Icon
OZONE Widget Deployment

1. Enter Widget Information

2. Click Apply Button
OZONE Widget Deployment

1. Click Add Button

2. Select Test Admin 1
OZONE Widget Deployment

1. Double Click *Empty Widget* Icon
OZONE Widget Deployment

CLIMACTIC FINISH!

Empty Widget

Develop Me!
OZONE Widget Development

Widgets developed and deployed in this method are nothing more than a standard Java WAR which can be accessed at the appropriate application context URL.
Example PreferencesWidget Project

- Simple project to demonstrate both storing and loading of preferences with the OWF API.

- Contains settings to build and deploy both project WAR and required Jetty context files to a local Jetty instance.

- Deploys to Jetty as https://localhost:8443/PreferencesWidget

- Can be easily imported into Netbeans or built using command line tools.
  
  Normal Build
  
  mvn clean install

  Build and Deploy
  
  mvn clean install –Pdeploy-jetty
Utilizes the Ozone.pref.PrefServer class.

User preferences are managed through the setUserPreference and getUserPreference methods.

Preferences are simply a string stored in OWF that is uniquely mapped to a user, namespace, and name combination.

It is suggested that non-trivial preferences be represented as JSON strings for storage and retrieval. If it can be represented as JSON is can be stored as a preference.

Supplied Dojo library provides mechanism for parsing and conversion of JSON strings via the toJson and fromJson methods.
function savePreferences(preferences) {
    Ozone.pref.PrefServer.setUserPreference({
        namespace: "com.geocent.owf.example.bookmarkWidget",
        name: "widgetPreferences",
        value: owfdojo.toJson(preferences),
        onSuccess: function() {
            [INSERT BUSINESS LOGIC HERE]
        },
        onFailure: function(error, status) {
            if (status != 404) {
                Ozone.util.ErrorDlg.show("Error saving preferences.");
            }
        }
    });
}

“Store the JSON value of the preferences object within the preferences system for this user under the namespace com.geocent.owf.example.bookmarkWidget and identified by the name widgetPreferences. When successfully stored execute the onSuccess function.”
function loadPreferences() {
    Ozone.pref.PrefServer.getUserPreference({
        namespace: "com.geocent.owf.example.bookmarkWidget",
        name: "widgetPreferences",
        onSuccess: function(storedPreferences) {
            var preferences = owfdojo.fromJson(storedPreferences.value);
            if( preferences != undefined && preferences != null ) {
                [INSERT BUSINESS LOGIC HERE]
            }
        },
        onFailure: function(error, status) {
            if (status != 404) {
                Ozone.util.ErrorDlg.show("Error loading preferences.");
            }
        })
    });
}

“Load from the preferences system for this user the preference under the namespace com.geocent.owf.example.bookmarkWidget and identified by the name widgetPreferences. When successfully retrieved execute the onSuccess function.”
Preferences Widget

Bookmark Widget

- Geocent
- Geocent Labs
- OWF Developers Group
- OWF GOSS
- OWF SoftwareForge Project

New Bookmark

Title

URL

Store
PreferencesWidget – Demo

Code Walkthrough

and

Widget Demonstration
Example CommunicationWidget Project

- Two part project to demonstrate the eventing communication framework and widget launcher framework with the OWF API.

- Contains settings to build and deploy both project WARs and required Jetty context files to a local Jetty instance.


- Can be easily imported into Netbeans or built using command line tools from the common parent CommunicationWidget project.

  Normal Build

  mvn clean install

  Build and Deploy

  mvn clean install –Pdeploy-jetty
Utilizes the Ozone.eventing.Widget class.

Eventing messaging is managed through the subscribe and publish methods.

Message payloads are published to a specified channel name and are received at all widgets that are subscribed to that channel.
OZONE Widget Framework - Eventing

- Functionally message payload are passed from the originating widget’s containing IFrame, to the parent OWF dashboard, down into the subscribed widgets’ containing IFrames.

Since data is internally transmitted within the OZONE Widget Framework via HTTP GET and POST requests developers should limit payload sizes to 2KB to fit within Internet Explorer’s limitations. For more information see: http://support.microsoft.com/kb/208427
OZONE Widget Framework – Widget Launching

- Utilizes the `Ozone.launcher.WidgetLauncher` classes.

- Widget launching is executed using the `launchWidget` method.

- The GUID of the target widget to launch must be known in order to launch the target widget. If this information is not known, the widget information can be discovered using the `Ozone.pref.PrefServer.findWidgets` method.

- The `findWidgets` method can perform searches on any combination of the `widgetName`, `widgetVersion`, and `widgetGuid` fields utilizing the `%` character as a wildcard match.
Ozone.pref.PrefServer.findWidgets({
    searchParams: {
        widgetName: "Widget Name"
    },
    onSuccess: function(widgets) {
        processWidgets(widgets);
    },
    onFailure: function(error, status) {
        if (status != 404) {
            Ozone.util.ErrorDlg.show("Error finding widgets.");
        }
    }
});

“Find all widgets whose title is Widget Name. When the search successfully completes execute the onSuccess function.”
function processWidgets(widgets) {
  if(widgets.length == 1) {
    var widgetEventingController = Ozone.eventing.Widget.get Instance();

    Ozone.launcher.WidgetLauncher.getInstance(widgetEventingController).
      launchWidget({
        guid: widgets[0].id,
        launchOnlyIfClosed: true
      });
  }
}

“Only launch the widget if one and only one is found. Get the widget eventing controller singleton and launch the first located widget if and only if the widget is currently closed.”
var widgetEventingController = Ozone.eventing.Widget.getInstance();

widgetEventingController.publish(
    "com.geocent.owf.example.announcer",
    owfdojo.toJson({
        timestamp: (new Date()).getTime()
    })
);

“Get the widget eventing controller singleton and publish the string representation of a JSON object containing the current timestamp to the channel com.geocent.owf.example.announcer”
var widgetEventingController = Ozone.eventing.Widget.getInstance();

widgetEventingController.subscribe(  
    "com.geocent.owf.example.announcer",  
    function(sender, msg, channel) {  
        var messageData = owfdojo.fromJson(msg);  

        [INSERT BUSINESS LOGIC HERE]  
    }  
);

“Get the widget eventing controller singleton and subscribe to the channel com.geocent.owf.example.announcer. When a message is receives process the message with the inline handler.”
ComWidgetAnnouncer / ComWidgetReceiver

Announcer Widget

Sending message in 5 seconds.

Receiver Widget

Message sent at Mon Feb 06 2012 15:42:30 GMT-0600 (CST)
Message sent at Mon Feb 06 2012 15:42:36 GMT-0600 (CST)
Message sent at Mon Feb 06 2012 15:42:42 GMT-0600 (CST)
Message sent at Mon Feb 06 2012 15:42:48 GMT-0600 (CST)
Message sent at Mon Feb 06 2012 15:42:54 GMT-0600 (CST)
Message sent at Mon Feb 06 2012 15:43:00 GMT-0600 (CST)
Message sent at Mon Feb 06 2012 15:43:06 GMT-0600 (CST)
Message sent at Mon Feb 06 2012 15:43:12 GMT-0600 (CST)
Message sent at Mon Feb 06 2012 15:43:18 GMT-0600 (CST)
ComWidgetAnnouncer / ComWidgetReceiver – Lifecycle

Load Announcer

Receiver Loaded?

Y

Set Timeout

Timeout Expired

Publish to Channel

N

Find Widget

Launch Found Widget

Subscribe to Channel

Receive on Channel

Display Message

Publish to Channel

Display Receiver

Display Message
Code Walkthrough

and

Widget Demonstration
Questions?
OZONE Widget Framework Pages

- Google Groups – ozone-developers
  http://groups.google.com/group/ozone-developers
- OWF GOSS Project Page
  http://owfgoss.org
- SoftwareForge Project Page
  https://software.forge.mil/sf/projects/ozone_widget_framework_owf

Contact Information

- OWF Community Support and GOSS Infrastructure Questions
  goss-support@owfgoss.org
- Me (Any other questions)
  joshua.penton@geocent.com
  joshua.penton@tech.geocent.com