EMAP: Enterprise Mashup Application Platform

Applies to:
Enter SAP technology or product release (release number, SP, etc.) to which this article or code sample applies (optional).

Summary
Enterprise Mashup Application Platform (EMAP) is a light-weight Mashup platform which provides a browser based application composition environment, tools and run-time that simplify developing applications on top of existing complex Enterprise Platforms and Services from conceptualization to deployment.

EMAP allows the user to create faster time to market Do It Yourself (DIY) applications quickly and helps the users to create variety of useful enterprise solutions.

EMAP is a Research project from SAP Research Center, Palo Alto.

Author: Rama Gurram
Company: SAP
Created on: 30 January 2009

Author Bio
Rama Gurram is a research scientist in the SAP Research Center at SAP Labs, Palo Alto, USA. In his current role at SAP, he is involved in the research of Advanced Web Technologies. Rama's primary areas of expertise include the architecture and design of Internet and Intranet applications, Service Oriented Architectures (SOA), Enterprise Application Integration, Internationalization, and Human Computer Interaction.
Table of Contents

Background.........................................................................................................................................................3
EMAP Objectives ..................................................................................................................................................3
Key Differentiating Points ..................................................................................................................................4
Currently Supported Features ............................................................................................................................4
User Management ..............................................................................................................................................5
Collaboration .......................................................................................................................................................5
Using EMAP to Build an Application....................................................................................................................6
   Enterprise Widget Repository ..........................................................................................................................7
   Event Hub Inspector .........................................................................................................................................7
   Event List .......................................................................................................................................................7
   Theme Switcher ..............................................................................................................................................7
   Application Switcher ......................................................................................................................................7
   Enterprise Widget Editor .................................................................................................................................7
   Policy Editor ..................................................................................................................................................7
Building an Opportunity Management Application ...............................................................................................8
Building Your Own Widgets ..................................................................................................................................13
Building a Publish/Receive Chat Application ....................................................................................................15
   Send Widget ................................................................................................................................................17
   Chat Log Widget .........................................................................................................................................18
Video .................................................................................................................................................................20
Further Reading ................................................................................................................................................20
Copyright...........................................................................................................................................................21
Background

Traditionally, enterprise applications are complex and monolithic. Efficient business operations are a critical factor in determining the competitiveness of an organization. Therefore, great amount of customization has to be made available in enterprise applications. An organization will have a big competitive advantage if it can support new business requirements faster. Today, the customization and extension of enterprise applications is hard, time consuming, and needs to be done by IT experts. By means of extending Mashup concepts which are popular in consumer space into enterprise application space it is possible make the enterprise platform more friendly for a new generation of developers and users.

Enterprise Mashup Application Platform (EMAP) is a light-weight Mashup platform which provides a browser based application composition environment, tools and run-time and simplifies developing applications on top of existing complex Enterprise Platforms and Services from conceptualization to deployment. EMAP allows business users to compose Enterprise Mashups by selecting Mashup building blocks or components from enterprise sources, web sources and component registries to easily create, deploy and share customized web applications with enterprise qualities such as security, scalability and manageability quickly. The basic building blocks of enterprise Mashups are OpenAjax Widgets compliant meta-data enhanced widgets. The meta-data in the widget allows EMAP to compose new solutions on the fly by using asynchronous events to exchange messages between the widgets in a loosely coupled fashion. Widgets communicate with the back-end data sources using light-weight RESTful resources and protocols, RSS and Atom to display the business data. The major architectural objectives of EMAP are to be light-weight, simple to use, and standards compliant.

EMAP allows the user to create faster time to market Do It Yourself (DIY) applications quickly and helps the users to create variety of useful enterprise solutions.

EMAP is a Research project from SAP Research Center, Palo Alto.

EMAP Objectives

Following are the EMAP’s key objectives:

- Provide a browser based enterprise class Mashup application framework for creating end-user driven lightweight Mashup applications using components /widgets that connect to several data resources
- Create a browser based development and run-time environment that simplifies developing data oriented Mashup applications from conceptualization to deployment
- Make the platform friendlier for a new generation of developers and give the end user the power to innovate
Key Differentiating Points
Following are some of the key differentiating points from the Mashup frameworks that are mainly available in consumer space:

- Simple & Light Weight Framework to compose the applications quickly using Saplet (aka widget, gadget) components as building blocks
- True interaction and communication between Saplets on the client side using the metadata rather than pure display of individual data
- Supports wiring and coupling of different Saplets, 3rd party components from different sources/vendors on the client side
- Mix of design and runtime (design-time view = runtime view), providing means for customizing and adapting during runtime, resulting in a very adaptive working environment
- Data-centric integration using REST, Atom / RSS by consuming data oriented services
- Extensible - Supports Open standards (OpenAjax, Ajax, JavaScript …) and 3rd party integration

Currently Supported Features

- Creating Mashup applications using Saplets in the browser
- Saplet Repository to store the Saplets.
- OpenAjax conforming metadata enhanced Saplets as building blocks of Mashup application
- 3rd party widget support (Google Gadgets)
- Support for design time tools (Saplet Editor, Saplet Repository, Event Inspector)
- Client-side events between Saplets using OpenAjax conforming Event Hub
- Application Manager to manage and store the state of the Mashup applications
- Simple user management and sharing of applications and Saplets
User Management

The EMAP user management is setup to work on top of the SAP NetWeaver platform. We have evaluated several authentication methods that are suitable for this environment, and found that OpenID is the most promising. Support for OpenID is planned, however the current version only supports simple HTTP authentication, which would requires you to register a username.

After registering a username with EMAP, you can authenticate via HTTP and access your distinct application contexts, as seen in Figure 1. Here, you are allowed to create new, as well as access and edit existing applications.

Collaboration

When browsing your application contexts, you will also see applications that other users created. Those applications have previously been published by the owners. Every application you create makes you its owner, and the application is private by default. That means, that nobody except you will be able to access it. You may choose to publish an application, so it will become available to other users and be displayed in their application contexts.

Every application consists of a set of wired Saplets that are configured with properties. Those properties might include sensitive information, e.g. your username and/or password for third party services. Consider a calendar in your application that should display both your Microsoft Outlook events from your enterprise and an online Google calendar in which you scheduled business related talks you want to attend in your area. As long as there is no full support for OAuth, you will have to provide the application with your Google calendar credentials.

That is why you won’t be able to use a working application somebody else published. When entering a public application context that is not yours, you will instead create a clone of that application and be required to reconfigure certain properties. Before you enter your cloned application context, you’ll be asked to provide a unique name for your clone. The new application will be private again, and you will be the new owner. If you choose to enhance the application you just cloned, you may to publish it again. You are encouraged to name it similar to the one you based your work on, and include a version number to the end. Be aware that other people can clone applications you created in the same way. You might revoke a public application you’re the owner of by making it private again or deleting it, however the applications already cloned from it by other users will not be affected.

The repository of Saplets knows only public Saplets. Every Saplet added to EMAP will be accessible by any other user that chooses to work with it. Be aware of that when adding new Saplets. You will remain the owner, and as long as the Saplet is not used in other applications, you might delete it. Considering collaboration, the Saplets in the repository will be ratable. That means that every user might rate a Saplet with a number of stars. That way, it will be easier to spot good and reliable work from others to reuse it in your Mashup.

Future versions of EMAP will include OpenID authentication, a finer granularity of sharing applications with your co-workers, friends and the public, and an enhanced model to share and limit access to confidential Saplets. Also, there will be support for application sharing on a group level, so that you may reuse them only within a set of users you choose.
Using EMAP to Build an Application

The steps required to build a Mashup application using EMAP are described below. Open the EMAP (BijouX) URL in the browser; you will see all of the previously created application contexts as seen in Figure 1.

You can either select one to open it or use the “New” button to create your own. You can also right click on applications to delete them. Please select “New” and give your application a unique name.

Once your application is created, click on it and you will see the default application view shown below in Figure 2.

Figure 1: Application Contexts

Figure 2: Newly Created Application
In order to start creating your application, you need to expand the harmonica panel on the left hand side of the screen by pressing the arrow buttons on the top. Now you will see what is shown in Figure 3.

![Figure 3: Development Saplets](image)

This harmonica panel contains all of the design-time widgets, or widgets that allow you to build your application. By collapsing and expanding it, you can continue to see what your application looks like to the end user, while editing the application. The widgets it contains are:

**Enterprise Widget Repository**
This is where all of the widgets are stored. From here you can edit the widgets and also add them to the application.

**Event Hub Inspector**
This widget displays all of the events that occur within the application.

**Event List**
The Event List provides the user a list of all the topics that they can subscribe and publish to.

**Theme Switcher**
The user can switch between different themes which are provided as CSS files.

**Application Switcher**
Here, the user can switch to another application.

**Enterprise Widget Editor**
This editor widget can be opened by clicking it under EMAP in the Widget Repository, or it can be opened by right clicking any widget and selecting “Edit”.

**Policy Editor**
This editor can be opened from selecting it from the Widget Repository. From there, the application developer can decide which widgets will be able to communicate.
Building an Opportunity Management Application

To demonstrate how to build Mashup applications in EMAP, we are going to consider an Opportunity Management application, which is based on a SAP CRM scenario. In this scenario, a sales person needs to explore his or her pursued opportunities and the details of each opportunity. This scenario is going to showcase the use of EMAP’s CRM Saplets and 3rd party Google Gadgets to build an Enterprise Mashup application.

Now we can start building our Mashup application (Opportunity Management Application). To do this, we will start adding our widgets to the application:

- First, you may want to add the “My Opportunities” widget from the “CRM” category in the Widget Repository. When selected, it will be added to the center column of the application.
- Feel free to move around or resize the widget or the columns, until you are happy with the layout.
- You can also play around with the theme switcher to see if you would prefer another look and feel.

Next we will add two additional widgets to the application:

- Add the “Opportunity Contact Details” widget from the “CRM” category and the “RSSAtomReader” from the “Readers” category – both to be found in the Widget Repository in the upper left corner of EMAP.
- Select an opportunity from the “My Opportunities” widget. The details for this opportunity will now show up in the “Opportunity Contact Details” widget.
- Open the “RSSAtomReader” widget properties. Enter the URL of a newsfeed that is related to your customer’s field of business or any other feed that you are interested in.

It would be useful to display a contact’s location on a mapping widget. We will do so by importing a Google Gadget (3rd party widget):

- Select the Gadget you would like to import. In our case, we would like a map. Search for the “US Weather Radar” widget and then select it. You will then see the widget and a link to “View source”. Copy the location of that link (Figure 4).
In order to run the “US Weather Radar” widget in our application context, open the Widget Editor from the “EMAP” category of the Widget Repository. Select the “Import” button at the top and choose the option “Google Gadget” as shown in Figure 5.

![Figure 5: Import a Google Gadget](image)

Enter a unique ID and the URL you previously copied into the text box and press “Import”.

Your widget will be imported into the “Uncategorized” category of the Widget Repository and the imported code will show up in the Widget Editor.

To finally add the new imported widget to the application, simply select it from the Widget Repository.
Now that we have all of the necessary widgets, we will wire them together.

- What we want to achieve is having the details as well as the location of a selected contact displayed in our “Opportunity Contact Details” widget and the “US Weather Radar”.

- To wire the mentioned widgets together, we need both of them to listen to certain messages from the “My Opportunities” widget. In particular, we need the latter to pass the prospect id of the selected contact to the “Opportunity Contact Details” widget and the contact’s zip code to the “US Weather Radar”.

- We’ll start setting up the receivers. Expand the properties of the “Opportunity Contact Details” widget by selecting the icon in the upper right corner that shows two arrows pointing downwards. Choose “Edit Properties” as seen in Figure 6.

![Figure 6: Editing a widget's properties](image)

- You can see that the query property is subscribing to a topic called “crm.prospect_selected”. We can either change the value of this topic, or leave it be. We have to make sure, that our “My Opportunities” widget is publishing to that exact same topic. Make sure the option “Subscribe” is checked and press the “Save” button.

- Also edit the properties of the “US Weather Radar” and make it subscribe to the “loc” topic as seen in Figure 7.

![Figure 7: Setting the properties of the “US Weather Radar”](image)

- Now we want to ensure that our “My Opportunities” widget is able to send messages to the receivers. As mentioned above, it has to be ready to “publish” events on the same topics, we set the subscribers to. Any message, passed on a topic that has active subscriptions will be processed by its subscribing widgets.

- Configuring a widget to publish on a certain topic is done in the same way as setting up a subscription, except checking the “Publish” box this time.
Figure 8 shows the right settings for our CRM application.

![Figure 8: Properties of the “My Opportunities” widget](image)

We can see that the widget indeed publishes to both topics, the subscribers are set to.

Taking a look at the source code of the publishing widget, we can see that it modifies the “loc” topic when a contact is selected (Figure 9).

```
this.grid = new Ext.grid.GridPanel({
  store: this.store,
  id: this.id + '_grid',
  viewConfig: {
    forceFit: true
  },
  columns: [
    { header: 'First Name', sortable: true, dataIndex: 'firstname' },
    { header: 'Last Name', sortable: true, dataIndex: 'lastname' },
    { header: 'Email', sortable: true, dataIndex: 'email' },
    { header: 'Telephone', sortable: true, dataIndex: 'phone' },
    { header: 'Country', sortable: true, dataIndex: 'country' },
    { header: 'Postal Code', sortable: true, dataIndex: 'postcode' }
  ],
  on: new Ext.grid.RowSelectionModel({
    singleSelect: true,
    listeners: {    
      rowselect: function(se, rowIndex, r) {
        var rec = o.grid.store.getAt(rowIndex);
        a.setProperties({value:'loc', rec_get('postcode')});
      }
    }})
});
```

Figure 9: Source code of the “My Opportunities” widget
Since we know now, that the communication is set up properly, we can give it a try. If we select a contact in the "My Opportunities" widget, notice that its details are shown in the "Opportunity Contact Detail" widget and the "US Weather Map" shows the right location (Figure 10).

We also have the capability to let users run their widgets on their desktop which means that they are isolated from the browser window (optional step):

- The same drop down menu we need to change a widget’s properties shows an option named “About this Gadget” (Figure 11). In order to enable this functionality you need to install a small program on your local computer. We are working on removing this dependency.

- If we choose the option “About this Gadget”, we see two links. The first one opens the widget in a new browser window as a standalone widget. The second link opens the desktop widget application and runs the standalone widget there.

![Figure 10: The CRM Application up and running](image)

![Figure 11: "About this Gadget"](image)
Building Your Own Widgets

Finally, we will discuss what the actual content of the widget contains, in case you need to make changes to the widget source code. Figure 12 shows the general format of these widgets, which is in OpenAjax format. Details are discussed below.

As before, use the Widget Editor to open the source code of the widget. Here you will see the widget code, which is in OpenAjax format. This format can be broken up into three major sections:

a. **Widget**: The `<widget>` element is the root element of a widget description file. Each widget description file describes a single widget.
   i. **name**: unique name, this will be used as the Saplet ID
   ii. **id**: unique id for the widget, good idea to use a URI
   iii. **height**: optional height of the widget
   iv. **<title> element**: title to display in the repository
   v. **<categories> element**
      1. contains one `<category>` element with the name attribute of the category

b. **Properties**: The `<properties>` contains all of the properties for the widget, which can be used to publish or subscribe to events.
   i. Each `<property>` element must contain
      1. **name**: the name of the property
      2. **default**: the default value of the property
   ii. and may contain
      1. **publish**: set to “true” if it publishes
      2. **subscribe**: set to “true” if it subscribes
      3. **topic**: if the property publishes or subscribes, the topic
      4. **<title> element**: the display name of the property
      5. **<options> element**: options if the property is an enum

Figure 12: OpenAjax Widget Format
c. **Content:** The `<content>` element either needs to reference an HTML page or contains the actual content for the widget.
   i. If type="fragment" (the default), then the content MUST be suitable for inclusion within an HTML `<div>` element. The content must also be inside `<![CDATA[...]]>` tags.
   ii. If type="page", then the content MUST be contained within an external file referenced by the `src` attribute. The referenced URL MUST produce a complete Web page (e.g., an HTML file that begins with an `<html>` element) and therefore suitable as the target for an HTML `<iframe>` element.

Building a Publish/Receive Chat Application

Next, we'll create simple chat publish and chat receive widgets.

- First open the Editor, create a new widget, let's call it "Send". Press "Save".
- Next, let's create a property to publish the chat message, so add the following code to the widget, between the <title> and <content> elements.

```xml
<properties>
  <property name="chat" topic="chat_message" publish="true" />
</properties>
```

- Now we need to create a text box in the widget that publishes to that topic, so we need to add the textfield to the content by adding the following to the widget:

```javascript
Send = function() {
  this.chatField = new Ext.form.TextField({
    anchor: '100%',
    enterIsSpecial: true,
    value: this.getPropertyValue('chat'),
    listeners: {
      'specialkey': {
        fn: function(field, event) {
          if (event.getKey() == Ext.EventObject.ENTER) {
            this.setPropertyValue('chat', field.getValue());
          }
        },
        scope: this
      }
    }
  });

  this.onReady = function() {
    var viewport = new Ext.Viewport({
      border: false,
      hideBorders: true,
      style: 'padding: 5px; background: #fff;',
      layout: 'fit',
      items: {
        layout: 'anchor',
        items: this.chatField
      }
    });
  }
};
```
Next we'll create the chat widget. To do this, we again create a new widget, let's call it “ChatLog”. Again, we need to create a property that listens to that event, so we will add the following to the widget:

```xml
<properties>
  <property name="chat" topic="chat_message" subscribe="true" />
</properties>
```

Now we need to have this message append to a list of messages, so we add the following to the content:

```xml
<content type="fragment">
  <![CDATA[
  ChatLog = function() {
    this.chatField = new Ext.form.TextArea({
      anchor: '100%',
      height: 200,
      width: 300,
      enterIsSpecial: true,
      value: this.getPropertyValue('chat')
    })

    this.onChatChange = function() { 
      var currString = this.chatField.getValue();
      currString += this.getPropertyValue('chat') + "\n";
      this.chatField.setValue(currString);
    }

    this.onReady = function() { 
      var viewport = new Ext.Viewport({
        border: false,
        hideBorders: true,
        style: 'padding: 5px; background: #fff;','
        layout: 'fit',
        items: {
        layout: 'anchor',
        items: this.chatField
      }
    });
      }
    }
  ]]></script>

In the end you should have the widgets consisting of the source code on the following pages. Figure 13 depicts how they should be rendered in your browser. Your code should look like:
Send Widget

<widget xmlns="http://openajax.org/metadata"
    name="Send" id="http://openajax.org/bijoux/widgets/Send"
    height="45">

    <title>Send</title>
    <categories>
        <category name="chat"/>
    </categories>

    <properties>
        <property name="chat" topic="chat_message" publish="true"/>
    </properties>

    <content type="fragment">
        <![CDATA[
            <!--
            Send = function() {
                this.chatField = new Ext.form.TextField({
                    anchor: '100%',
                    enterIsSpecial: true,
                    value: this.getPropertyValue('chat'),
                    listeners: {
                        'specialkey': {
                            fn: function(field, event) {
                                if (event.getKey() == Ext.EventObject.ENTER) {
                                    this.setPropertyValue('chat', field.getValue());
                                }
                            },
                            scope: this
                        }
                    },
                }));
                this.onReady = function() {
                    var viewport = new Ext.Viewport({
                        border: false,
                        hideBorders: true,
                        style: 'padding: 5px; background: #fff;',
                        layout: 'fit',
                        items: {
                            layout: 'anchor',
                            items: this.chatField
                        }
                    });
                }
            } -->
        ]]>
    </content>
</widget>
Chat Log Widget

<widget xmlns="http://openajax.org/metadata"
  name="ChatLog" id="http://openajax.org/bijoux/widgets/Send"
  height="220">

  <title>ChatLog</title>
  <categories>
    <category name="chat"/>
  </categories>

  <properties>
    <property name="chat" default="" topic="chat_message" subscribe="true" />
  </properties>

  <content type="fragment">
    <![CDATA[
      ChatLog = function() {
        this.chatField = new Ext.form.TextArea({
          anchor: '100%',
          height: 200,
          width: 300,
          enterIsSpecial: true,
          value: this.getPropertyValue('chat')
        })

        this.onChatChange = function(){
          var currString = this.chatField.getValue();
          currString += this.getPropertyValue('chat') + '\n';
          this.chatField.setValue(currString);
        }

        this.onReady = function() {
          var viewport = new Ext.Viewport({
            border: false,
            hideBorders: true,
            style: 'padding: 5px; background: #fff;',
            layout: 'fit',
            items: {
              layout: 'anchor',
              items: this.chatField
            }
          });
        }
      }
    ]]>}
  </content>
</widget>
<table>
<thead>
<tr>
<th>Send</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Good, thanks for asking!</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ChatLog</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello</td>
<td></td>
</tr>
<tr>
<td>Testing testing 1 2 3</td>
<td></td>
</tr>
<tr>
<td>Hi, How are you?</td>
<td></td>
</tr>
<tr>
<td>Good, thanks for asking!</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 13: Send and Receive Chat Widgets**
Video

Further Reading
Copyright

© Copyright 2009 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.