International Journal of
Electronic Commerce Studies
Vol.1, No.2 , pp.139-148, 2010

MAIL PLUGIN FOR IBM MASHUP CENTER

Pinaki Mitra,
Indian Institute of Technology, Guwahati, India
pinaki@iitg.ernet.in

Girish Sundaram,
IBM-ISL, Pune, India
gisundar@in.ibm.com

Senthil Kumar G
IBM-ISL, Bangalore, India
senthigk@in.ibm.com

Vishal Kurup,
Indian Institute of Technology, Guwahati, India
v.kurup@iitg.ernet.in

ABSTRACT

A mashup gathers useful information from various sources and presents it in a meaningful way. The IBM Mashup Center is one of the most widely used mashup editors. Data comes from different sources in various formats and the mashup editor combines them together and gives the output in a different format. The IBM Mashup Center currently supports feed generation from various sources like Excel files, Access files, database files, CSV files, etc. Currently, however, it does not support feed generation from a mail server. Thus, we have implemented a mail plugin that adds this feature to the mashup center. The mail plugin fetches the mail using the POP3/IMAP protocol and converts it into ATOM feeds. The mail plugin uses a Java mail API for connecting to and accessing the mail server. The generated feed gets a unique URL which can be used by widgets to access the feed. It supports the dynamic addition of new mail servers to its trust store. Security is further enhanced by providing a digital signature feature that is in enveloped form, i.e. it goes with the feed as one of its elements. The digital signature is checked at the receiver side to assure data integrity.

Keywords: Web 2.0, mashup, mashup editor, IBM Mashup Center, mail plugin

1. INTRODUCTION

One of the most important reasons for the popularity of Web 2.0 is the sharing of data. A mashup1 consists of applications which can retrieve data from different sources and other mashups, and display it in a single webpage. The basic idea of a mashup is to gather useful information from various sources and present it in a meaningful way. Today, mashups are becoming extremely popular because they support rich internet applications like Google Maps, etc., that can be used to improve business services. The biggest advantage is that you can use a third party service for your needs and there is no need to prepare the desired features from scratch. Many editors are available to create mashups, examples include the IBM Mashup Center2, Smash3, Google Mashup Editor, Microsoft Popfly, Intel Mashmaker, Yahoo Pipes, etc. A mashup can usually be assembled on the client side, with the editor residing on the server side. It can also be assembled at the server side. The common link in both is the interface used for assembling content, is done thru common browsers like Firefox, IE, Opera, Safari, etc.

The IBM Mashup Center is one of the most widely used mashup editors. Data comes from different sources in various formats and the mashup editor combines them together and gives the output in a different format. The mashup editor uses an internal data model, which is a unified schema of the entire data. The IBM Mashup Center uses a graph-based approach or XML format as the internal data model. The IBM Mashup Center basically contains three parts, the first part takes contents from different sources and converts it into an ATOM feed, the second part filters and combines ATOM feeds from different sources, and the third part displays them in widgets. Currently, the first part supports data from a number of sources like Excel files, database files, html files, etc., but it doesn't currently support emails.

The mail plugin connects to a mail server through the POP/IMAP protocol and converts the emails into ATOM feeds. The plugin is implemented with the help of Mashuphub APIs4, which provides a plugin mechanism for adding extensions for the support of new feed sources. The mail plugin consists mainly of two parts. The first part deals with the display or user interface part of the plugin, which is implemented in Dojo and html. The second part deals with the processing of the information and generation of feeds that are implemented in Java5. There are four security features in the mail plugin. These features are encryption of the feed, access-control rights of the feed, dynamic addition of a mail server to a trust store and addition of a digital signature to the feed. The first two features (i.e. feed encryption and access control) are provided by the

plugin framework. We have implemented the remaining two security features.

The remainder of this paper is organized as follows. In Section 2, we provide a detailed introduction to mashups and their internal data format. In Section 3, we discuss the MashupHub plugin architecture. In Section 4, we discuss the mail plugin and its features. In Section 5, we conclude with some ideas about the future extensions.

2. MASHUP

One of the goals of Web 2.0 is to make it easy to create, use, describe, share and reuse resources on the web. To achieve that, technologies have flourished around this concept. The capabilities of Web 2.0 are further enhanced by many service providers who expose their applications in two ways. One way is to expose application functionalities via Web APIs such as Google Maps, Amazon.com or YouTube. The other is to expose data feeds such as RSS and ATOM. This approach opened up new and exciting possibilities for service consumers and providers as it enabled the notion of using these services as ingredients that can be mixed and matched to create new applications. To achieve this goal, a new framework called mashup was developed. Mashup is a new application development approach that allows users to aggregate multiple services to create a service for a new purpose.

2.1 Internal Data Model

Data comes from various sources in different formats. The mashup combines them together and gives an output in a different format6. The mashup uses an internal data model that is a unified schema of the entire data. The internal data model of mashup tools can be graph based or object based. The IBM Mashup Center and most of editors use a graph-based approach, which is nothing but an XML representation of data. Microsoft's Popfly editor uses an object-oriented approach. The web contains data in heterogeneous formats, thus we need to convert it into a common format, which in case of the IBM Mashup Center is an ATOM feed format. Our work contributes to this area by adding mail servers to the data source.

3. PLUGIN ARCHITECTURE

MashupHub provides a plugin API that is used to add plugins to the feed generator component of the IBM Mashup Center. The plugin API

allows developers to write components that extend the ability of the MashupHub to store, search, and retrieve catalog entries (feeds, widgets, pages, services, and so on). Plugins can show editors that let users define or upload new entries, show forms that collect user parameters for feeds, generate feeds given zero or more parameters, allow clients to download opaque entries (feeds, pages, and so forth), and define custom handlers for application queries.

The fundamental plugin extension components are the generator editor and collection extensions. The generator extension component is called to generate an instance of a feed from some source. The editor component provides the user interface to define the feed, or process a widget or other object. It is invoked when a user selects the object from the Create New Feed or Upload selection lists in the user interface. The editor can be simple, using a Java Server Page or emitting raw HTML, or can be more sophisticated and provide a rich user interface. The main method for displaying the editor's page in the MashupHub user interface is renderEditor. The plugin model also provides JavaScript APIs for use by the plugin's JavaScript code running in the user's web browser. This allows the plugin to participate in the page flow that is common to all plugins in the New Feed and various Upload user interfaces

3.1 Plugin Development

Plugins can be developed using any Java development environment. MashupHub provides a Java Archive (JAR), mhubapi.jar that contains all the interfaces and classes needed to compile a plugin implementation. The JavaScript functions that constitute the client-side API are in the default installation.

3.1.1 Plugin Installation

Installing a new plugin and making it active in MashupHub is an administrator task. The administrator must place the plugin zip file in the <WebApplication>\WEB-INF\ plugins folder. The <WebApplication> folder is the directory where MashupHub Web applications are installed. In the default installation it is at <install-dir>\installedApps\ MashupHub.ear\mashuphub-enterprise.war\WEBINF\plugins. The plugin is identified by the name of its zip file. The file name must be the same as its java package name, such as com.myco.myplugin.zip. The administrator can install, upgrade and remove the plugin.

3.1.2 Plugin Removal

When the zip file for the plugin is removed from the <WebApplication>/WEB-INF/plugins folder and MashupHub is restarted, the plugin is uninstalled, and all of its files are removed and catalog entries defined using this plugin are deleted. System plugins cannot be uninstalled.

3.1.3 Plugin Upgrades

To upgrade a plugin, place a newer version of the plugin package in the <WebApplication>/WEB-INF/plugins directory. MashupHub will check the time stamp of the file to determine if the zip file is newer. The MashupHub Web application must be restarted to activate the new version.

4. MAIL PLUGIN

Mail Plugin is an external plugin for the Mashup Center that connects to mail server using POP3/IMAP standard protocols. It is classified as an enterprise plugin.

4.1 Flow-Control of Mail Plugin

The mail plugin flow-control diagram is shown in Figure 2.

As specified in the meta file, the MailEditor class provides the plugin editing function. Its implementation extends BaseEditor, a base class that requires implementation of the renderEditor method. The renderEditor method is called by the framework when users make the selection to create a new feed using this plugin, or when users edit an existing feed previously created by this plugin. The renderEditor method takes a single parameter: IEditorContext. From the context, we get the IRequestContext interface containing information sent from the browser, and IEntry containing all the information maintained by the framework for this feed instance. The parameter is common to all methods invoked by the framework in response to user editing actions. The renderEditor method returns an instance of the type ViewBean. The main purpose of the ViewBean is to specify the JSP used by the feed generation framework to create an HTML fragment for the plugin-specific editor.

4.2 Mail Plugin Security

Security for the mail plugin has four main parts, which are as follows:

1. The generated feed is encrypted by using one of keys from the keystore. This feature is already implemented in the framework.
2. The access control features of the feed are given at feed creation time. This feature is also already implemented in the framework.
3. Dynamically adding untrusted mail servers to the trust store. We have implemented this feature.
4. If a widget doesn't support a secure connection, then there is the option of adding a digital signature to the feed. We have implemented this feature.

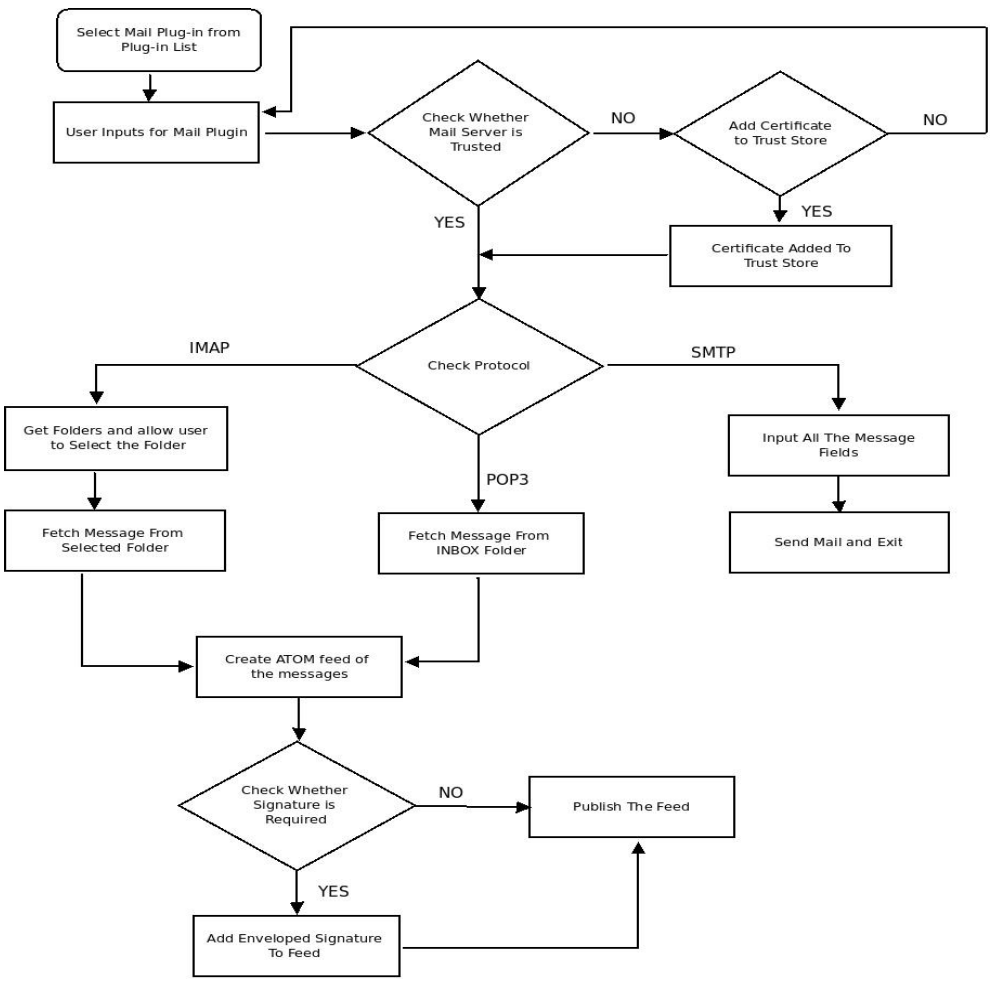

Figure 2. Flow Control of Mail Plugin

A set of certificates for the user is created when his account is created at the server. The feed is encrypted using one of the randomly selected keys from the key set. This feature is provided by the framework itself, so the mail plugin feed can also use this feature. Access control details are given with the other details of the feed at feed creation time. This feature is also provided by the framework for all plugins. It supports three types of access control. The feed can be made private (i.e. accessible to this user only), public (i.e. accessible to all) or protected (i.e. accessible to a group of users only). We have implemented the other two security features, which are discussed in the next section. When the mail server is untrusted, the user is given an option to add the server to the trust store by showing necessary details like issuer, signer, valid up to, etc., If the user accepts it then it is automatically added to the trust store.

4.2.1 Dynamically Adding Untrusted Mailserver

This is the first step done after saving the values entered by the user. The mail server connection is checked on the specified port. If it gives a SSL Handshake Exception, then it indicates that the server is not present in the trust store of the client. This checking is implemented in checkCertificate() method of the CheckAndInstallCert class. If there is no exception, then the mail plugin proceeds with its normal function.

Once we get the SSL Handshake exception, we get the certificate chain from the mail server. From the certificate we extract the certificate information and display it to the user, if the user accepts, then the installCertificate() method of the CheckAndInstallCert class is called, which adds the certificate to the Java trust store. In this implementation, it is assumed that the "Java home" variable is set to the Java used by the IBM Mashup Center. The passkey is assumed to be "changeit" which is the default passkey of the Java trust store. The certificate is added using the alias "hostname_0". After this step, the mail plugin proceeds with its normal function.

4.2.2 Adding a Digital Signature to the Feed

This feature is useful when the widget doesn't support https, i.e. when the feed is accessed in plain format. This is an optional feature which can be enabled from the initial user interface page. The mail plugin adds an enveloped signature to the feed. The enveloped signature is included in the feed as a new element. The signature is calculated over the feed excluding the signature part. The validation of the signature should be implemented at the widget or receiving level.

The keystore and key secret is given by the user which will be used to sign the feed. Key alias is an option field that is compulsory only when the keystore contains more than one key pair. First, all of the message digest of the feed is calculated using the SHA1 algorithm, which converts the feed into a 160- bit hash. Then it is encrypted using the RSA algorithm, which takes the private key from the keystore provided by the user. At the receiving end, again a hash is calculated and is compared with the decrypted hash. If both are same, then the feed is accepted, otherwise it is rejected. This provides data integrity.

5. CONCLUSION AND FUTURE WORK

The mail plugin adds one more data source to the IBM Mashup Center feed generation component. This feature will enable the mashup center user to use a single widget to get mail updates from different mail servers. It will also allow filtering of the mail to receive mails from a particular email id. The dynamic addition of a certificate to the trust store simplifies the use of the mail plugin. The digital signature feature makes it possible to check the integrity of the feed when accessed in an unencrypted format. This feature is currently only available in the mail plugin. In the future this can be made a part of the plugin framework so that other plugins can also take advantage of this feature. A search feature can also be added to the mail plugin, which will enable a widget to send queries and get their reply. Currently, the mail plugin only supports the secure version of protocols. Looking forward, features like dynamic detection of the mail server port and support for non-secure versions of the protocol can be provided.

6. REFERENCES

[1] A. Taivalsaari and T. Mikkonen, Mashups and modularity: Towards secure and reusable web applications," in Automated Software Engineering - Workshops, 2008. ASE Workshops 2008. 23rd IEEE/ACM International Conference on, 2008, p 25-33

[2] F. De Keukelaere, S. Bhola, M. Steiner, S. Chari, and S. Yoshihama, “Smash: secure component model for cross-domain mashups on unmodied browsers," in WWW '08: Proceeding of the 17th international conference on World Wide Web. New York, NY, USA: ACM, 2008, p 535-544

[3] MashupHub API. [Online]. Available: download.boulder.ibm.com/ibmdl/pub/software/mashups/mhapiref.pdf

[4] Dojo Tool Kit. [Online]. Available: http://www.dojotoolkit.org
[5] Java Mail API. [Online]. Available: http://java.sun.com/products/javamail/javadocs/index.html
[6] G. Di Lorenzo, H. Hacid, H. young Paik, and B. Benatallah, Data integration in mashups," SIGMOD Rec., 38(1), p 59-66, 2009.