

IBIS - Intelligent Band Information System

(<http://ibis.3wep.de>)

Roland Kölbel and Matthias Tylkowski
Chair of Internet Technologies
Brandenburgische Technische Universität Cottbus

16. July 2009



Abstract

The IBIS (Intelligent Band Information Service) platform is a mashup of different services, which provide band-related information. Special attention was given to the intelligent aggregation on the one hand and the customization by the user on the other hand. Services currently included are DBPedia - an extraction of semantic data from Wikipedia, YouTube, Amazon, Twitter, Eventful and GoogleMaps. The user has a dashboard and can decide which of these services he wants to add or to modify. On some services such as DBPedia the user can also decide which information he wants to include, such as the current band members or the abstract of the band. The user can also decide how this information should be displayed. Services can be positioned freely on the dashboard and the components size is also changeable. All these adjustments can be saved persistently on the IBIS server or on the own computer in one single HTML file.

1 Motivation

The aggregation of internet services, also known as a mashup, is a subject that has been under intensive development in the last few years. A growing amount of organizations permit access to their data via programmable interfaces and encourage users to develop new applications which, if they are successful, provide a bigger market share to the service providers.

Many users adopted these services and integrated them in their own web applications. Unfortunately most of these integrations lack a useful purpose to the user of the website. More and More implementations focus on the intelligent combination of web services with other services and of course with the own offered content.

The described implementation of a mashup should not only focus on the intelligent combination of services, but also on the modification of this generated combination by the user itself. Thus enabling the user to play an important role in the creation of a mashup.

2 The IBIS Model

The IBIS platform was indented to be a client-side-only mashup programmed in JavaScript, but due to technical issues concerning *Cross Site Scripting*¹ it was not possible to completely achieve this goal. A proxy which connects to the services and passes the data to the client had to be implemented. After several approaches with different kinds of proxies we decided to implement a PHP² proxy. A graphical illustration of the proxy functionality is given in Figure 1.

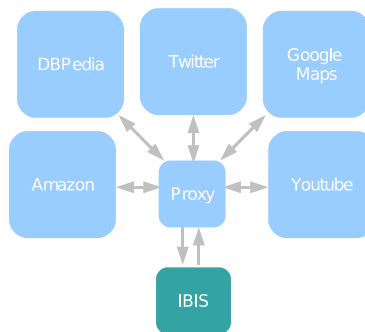


Figure 1: Communication Model

An advantage of this proxy approach is the ability to fulfill additional tasks such as XSLT³ transformations. Some service providers offer their data in Atom⁴ format such

¹ Cross Site Scripting, http://de.wikipedia.org/wiki/Cross-Site_Scripting, July 2009

² PHP, <http://www.php.net/>, July 2009

³ XSL Transformations (XSLT), <http://www.w3.org/TR/xslt20/>, July 2009

⁴ Atom, [http://de.wikipedia.org/wiki/Atom_\(Format\)](http://de.wikipedia.org/wiki/Atom_(Format)), July 2009

as YouTube⁵ and Twitter⁶. Others, such as DBPedia⁷ offers its data in JSON format⁸ while Amazon has its own XML format⁹. All of these return formats can be preprocessed by the proxy if necessary. As an exception, the JSON format can be processed in JavaScript as it is. The class diagram in Figure 2 shows the UML model of the JavaScript pseudo classes.

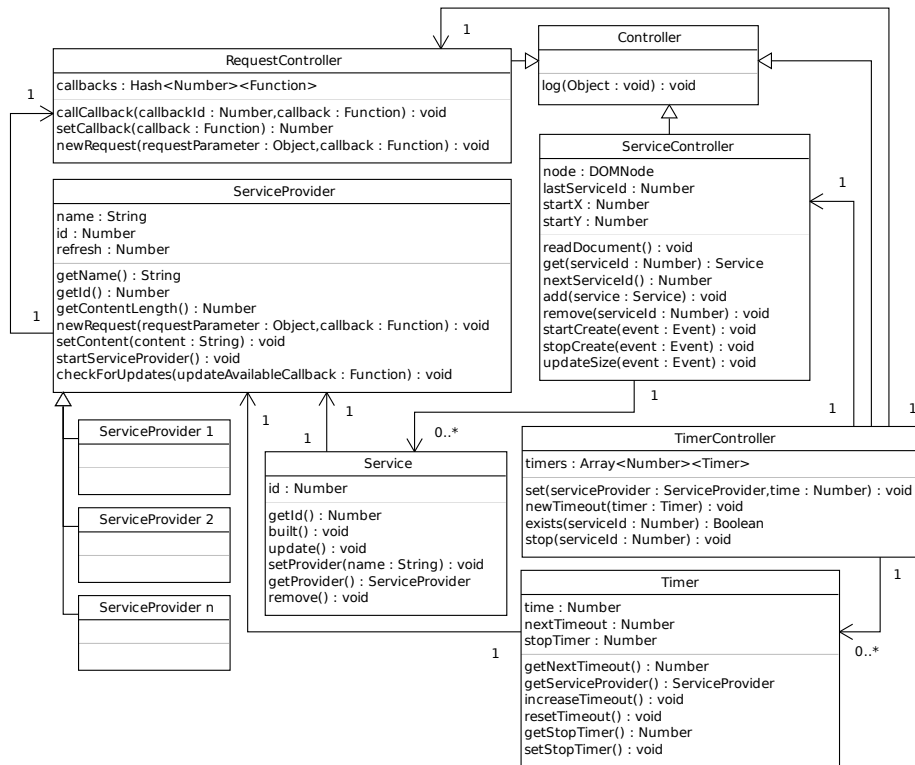


Figure 2: JavaScript UML Model

The mashup described in this document combines services by a given band or an artist name. After the automatic creation of the site, the user is able to make adjustments and refinements by changing the style of the website, by changing a specific service or adding additional services to the mashup.

These changes can be done without having any knowledge about computer programming at all, but simply by the use of mouse actions. Performed changes can be persistently saved and made accessible to other users searching for the same keyword. An other way of saving is the download of the source code of the created mashup without losing any of its functionalities.

To make the service customization as intuitive as possible a user interface consisting

⁵ YouTube, <http://www.youtube.com>

⁶ Twitter, <http://www.twitter.com>

⁷ DBPedia, <http://www.dbpedia.org>

⁸ JavaScript Object Notation, <http://de.wikipedia.org/wiki/JSON>, July 2009

⁹ Amazon Web Services, <http://aws.amazon.com/>, July 2009

of drag-able windows has been chosen (see Figure 3). These windows can be dragged freely across the website and changes can be performed by navigating through a window specific navigation bar located at the top of the window. This navigation bar is shown if the user moves the mouse over a service window.

3 Implementation issues and challenges

The components of the mashup implementation involve a HTML document for the basic structure, a JavaScript application for the whole service aggregation and customization as well as an internet proxy written for accessing the external services. Without the internet proxy a communication would not be possible due to cross domain restrictions in current web browsers.

There were several challenges when implementing the mashup. The goal of the mashup was to have everything in one single website, so that the visitor has the possibility to just include the offered code into his own web page and still obtains the full functionality of the mashup.

Most challenging was the Same Origin Policy¹⁰ used in current web browsers. It prohibits the sending of a request through JavaScript to a different domain than the current one. This is a problem because it prevents the mashup from running only on the client side.

To solve the problem a flash proxy was used to make the request calls. Flash did not have the same restrictions as JavaScript - before Flash 9. For cross site scripting to work the requested server now has to offer a `crossdomain.xml` file in its root directory explicitly allowing cross domain communication. Therefore we used the proxy server to redirect the communication and offer a valid `crossdomain.xml` to the flash client. Unfortunately the described solution violates partially the goal of having the whole mashup in one single file so another method was established.

Currently a new script tag with the URL of the proxy server and the target web page as parameter is passed directly into the head of the html document. The returned JavaScript executes a temporarily saved callback function with the desired content and thus bypasses the Same Origin Policy of the web browser.

All the requests are REST¹¹, based queries. The DBPedia service uses SPARQL¹² endpoints.

For the JavaScript components the class concept of the PROTOTYPE¹³ framework was used. It offers amongst others inheritance, which moves the pseudo classes of JavaScript closer to real classes.

¹⁰ Same Origin Policy, http://de.wikipedia.org/wiki/Same_Origin_Policy/, July 2009

¹¹ REST, http://de.wikipedia.org/wiki/Representational_State_Transfer, July 2009

¹² SPARQL, <http://www.w3.org/TR/rdf-sparql-query/>, July 2009

¹³ PROTOTYPE, <http://prototypejs.org/>, July 2009

A timer class was implemented which lets specified services update in defined intervals. When the timeout fires and the service has no updated content the time to the next timeout will be doubled. When the content was updated the initial time will be set again. Through this mechanism the number of requests will be reduced. To achieve this functionality the JavaScript function `setTimeout` was used.

4 Usage Example

In case the user does not open a URL with a specified keyword he will be directed to the mashup welcome page. From that point he can open some of the predefined examples and will be directed to the *IBIS Dashboard*. On this dashboard windows can be created by holding the left mouse button and drag a rectangle. On releasing the button a window will be created. This window has a title bar with two buttons, the right one closes the window and everything inside will be deleted. On hovering over the other button a drop down menu opens where the different services can be selected. The windows define the size which the content of the service can use. The windows can be moved freely by clicking on the title bar and dragging it across the dashboard. Resizing can be done in the same way by clicking on the dotted border of a window and dragging it. The title bar and the borders will only be shown when the user is hovering with the mouse over one window. A typical dashboard representation is shown in Figure 3.

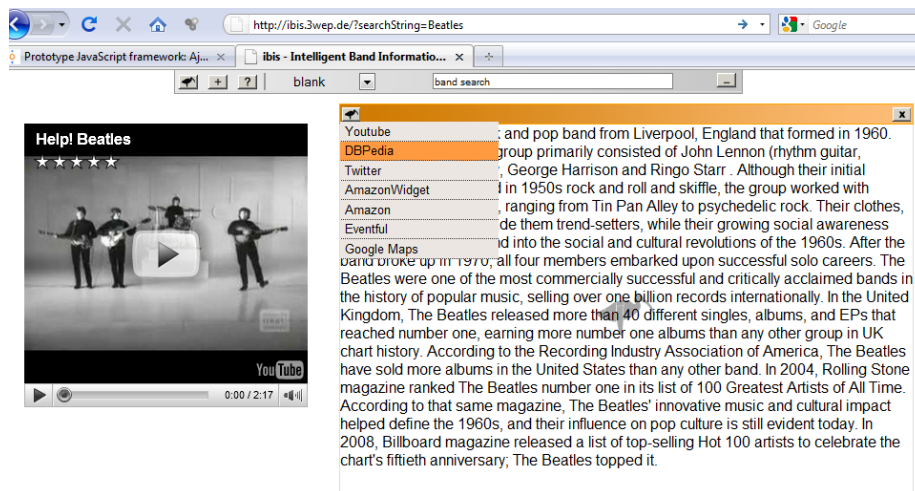


Figure 3: The IBIS graphical user interface

5 Conclusions and Future Work

With the current security policies of the browsers it is not possible to achieve a client side only mashup that is not a browser plug-in or does not need any kind of proxy mechanism. The implementation of the IBIS mashup allows the user to interact with the web page and get the information he really wants to have.

The IBIS System will be enriched with more services to give the user more freedom when combining the services of his choice. It is now possible to include a ticket booking service which shows the concert locations of the specified band on Google Maps, reduced on the locations around the position of the user. Location-aware browsing which was introduced with Mozilla Firefox 3.5 makes this feature available. This example shows the possibility of a connection between several services to enrich the usage of their data.

With the upcoming HTML 5¹⁴ it will be possible to have a client-side-only mashup, because it includes mechanisms where CSS (*Cross Site Scripting*) is not an issue anymore. Therefore the effort of creating a mashup will be significantly reduced.

¹⁴ HTML 5, <http://dev.w3.org/html5/spec/Overview.html#crossDocumentMessages>, July 2009