TECHNISCHE UNIVERSITÄT WIEN
VIENNA UNIVERSITY OF TECHNOLOGY

MASTERARBEIT

Mashups
a new concept in web application programming

ausgeführt am Institut für
Informationssysteme
der Technischen Universität Wien

unter der Anleitung von
Prof. Dr. Reinhard Pichler

durch

DI (FH) Christoph Klocker
Toldgasse 2 / 16
1150 Wien

Wien, am 08. Februar 2008

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Wien, am 5. Februar 2008

DI (FH) Christoph Klocker

Acknowledgements

Finally I will finish my second studies “Informatikmanagment” with this master thesis. I am glad to finish it after having had an intense time the last 2 years. During this time, when I used most of my spare time for my studies, I still got the full support from my girlfriend. Kathrin, thanks a lot for your support and considerateness.

Another person to mention is Dr. Robert Baumgartner. He gave me helpful advises writing the thesis and inspired my to write about *mashups*.

Last but not least my supervisor Prof. Dr. Reinhard Pichler.

Abstract

In recent years, a new trend in web application development has evolved, namely *Mashups*. These applications are hybrids that integrate content from different sources into something new, thereby adding value. This thesis gives a general overview of the kinds of mashups that exist and what technologies are most commonly used within mashups. It also introduces some tools and platforms for the creation of mashups.

The case study about creating a *Music Mashup* provides insight into the concepts behind mashups, such as web scraping, data mashing and front-end development.

Zusammenfassung

In den letzten Jahren entwickelte sich ein neuer Trend in der Webapplikationsentwicklung, die so genannten *Mashups.* Diese Applikationen sind Hybride, die Informationen von unterschiedlichen Quellen zusammenführen und dadurch neue Applikationen mit einem zusätzlichen Mehrwert kreieren. Die Masterarbeit zeigt einen allgemeinen Überblick der verschiedenen Mashuparten auf, führt in die wichtigsten zu Grunde liegenden Technologien ein und stellt einige Programme und Platformen für das Erstellen von Mashups vor.

Die Fallstudie *fmMusic*, ein Musik-Mashup, gibt einen Einblick in die verschiedenen Konzepte, wie Web scraping, Data mashing und Frontendentwicklung, auf welchen Mashups aufbauen.

Contents

Erklärung ii

Contents vi

List of Figures x

I Introduction 1

1 Introduction 2

1.1 Motivation 2
1.2 Mashups 3
1.3 Definition 3
1.4 Categories 4
1.4.1 Mapping mashups 4
1.4.2 Video and photo mashups 5
1.4.3 Data (News) mashups 5
1.4.4 Shopping mashups 6
1.5 Enterprise mashups 6
1.5.1 Definition 6
1.5.2 Service Compositions and Mashups 8

II Technology 9

2 Technologies 10

2.1 XML 10
2.1.1 A sample XML Document 11
2.1.2 Validation 11
2.2 XPath 12
2.2.1 How to use XPath 12
2.3 XSL Transformations (XSLT) 13
2.3.1 Usage of XSLT 13
2.4 Web Services 13
2.4.1 Where to find web services 14
2.4.2 How does it work 14
2.4.3 Implementations 15
2.5 Representational State Transfer 15
2.5.1 Definition 15
2.5.2 Design principles of REST 16
2.5.3 Methods of REST 16
2.5.4 Using YouTube's REST API 16

2.6 SOAP 17
2.7 SOAP vs. REST 18
2.8 AJAX 19
2.8.1 Introduction 19
2.8.2 Definition 19
2.8.3 AJAX Model 20
2.9 JSON 21
2.10 XUL 23
2.10.1 Features and Benefits 23
2.10.2 Requirements 24
2.11 RSS / ATOM 24
2.11.1 RSS 24
2.11.2 Atom 25

3 Mashup Techniques **27**
3.1 Server-side Mashups 28
3.1.1 Benefits 28
3.1.2 Drawbacks 29
3.2 Client-side Mashups 29
3.2.1 How It Works 30
3.2.2 Benefits 31
3.2.3 Drawbacks 31
3.3 Mashup Targeting 31
3.3.1 Presentation Mashups 31
3.3.2 Data Mashups 32
3.3.3 Logic Mashups 32

4 Web Data Extraction **34**
4.1 Definition 34
4.2 Information Extraction 34
4.3 Web Data Extraction 35
4.4 Extraction Tools 35
4.4.1 Taxonomy for Characterizing Web Data Extraction Tools 36
4.4.2 Deep Web Navigation 37

III Tools and platforms **38**

5 Mashup Enablers **39**
5.1 Dapper 39
5.1.1 Creating a Dapp 40
5.1.2 Limitations 40
5.2 Openkapow 41
5.2.1 Limitations 44
5.3 Conclusion 44

6 Mashup Platforms **45**
6.1 Introduction 45
6.2 What you can expect 45

6.3 What you will get . . . 46
6.4 Example Mashup platforms . . . 46
6.4.1 Yahoo! Pipes . . . 46
6.4.2 Microsoft Popfly . . . 48
6.4.3 IBM's Mashup Starter Kit . . . 50

IV FmMusic - a music mashup 53

7 Specification 54
7.1 Artist/Track . . . 54
7.2 Album, Album cover . . . 55
7.3 Lyrics . . . 55
7.4 Artist Information . . . 55
7.5 Music Videos . . . 56
7.6 Pictures . . . 56

8 Apache Cocoon 57
8.1 Introduction . . . 57
8.2 The Cocoon framework . . . 58
8.3 Separation of Concerns (SoC) . . . 58
8.3.1 Model View Controller . . . 59
8.3.2 Cocoon, MVC and Flowscripts . . . 59
8.4 The pipeline model . . . 59
8.4.1 Generator . . . 60
8.4.2 Transformer . . . 61
8.4.3 Serializer . . . 61
8.4.4 Matchers . . . 61

9 fmMusic backend 63
9.1 Filestructure . . . 63
9.2 Scraping Fm4 . . . 64
9.3 Amazon web services . . . 67
9.3.1 Using the ItemSearch operation . . . 68
9.3.2 Matching and extracting response information . . . 69
9.4 Lyrics extraction . . . 70
9.4.1 Sub-page navigation . . . 70
9.4.2 Linking pipelines using flow logic . . . 71
9.5 Pictures from Flickr . . . 73
9.6 Wikipedia . . . 74
9.7 Musicvideos from YouTube . . . 75

10 fmMusic frontend - a Firefox sidebar 78
10.1 Extending Firefox . . . 78
10.2 XUL Overlays . . . 79
10.3 Extended functionality with JavaScript . . . 81
10.4 Creating a tabbed interface . . . 82

V Future Issues **83**

11 Legal issues **84**
11.1 Copyright 84
11.2 Copyright in the digital age 84
11.3 Data ownership 85
11.4 Conclusion 85

12 Future Trends **86**
12.1 Role of Mashups in the Enterprise 87
12.1.1 Business Intelligence 87
12.2 Mashup tools and platforms 87
12.3 Conclusion 88

VI Appendix **89**

A YouTube API response **90**

B Stylesheets and transformation results **92**

C Extension source code **97**

Bibliography **99**

Bibliography **99**

List of Figures

1.1 Mapping Mashup - California wildfires 4
1.2 Google News - a data mashup 5
1.3 Pricerunner - Showing pricecomparisons for a product 6

2.1 XML document represented as tree 12
2.2 Request/Response web service 15
2.3 One way web service 15
2.4 Web application model comparison 21
2.5 AJAX process flow 22
2.6 XUL Example opened in Firefox 24

3.1 Sample homepage provided by Pageflakes 32

5.1 First step creating a dapp 40
5.2 Collecting sample pages 41
5.3 Selecting the content to extract 41
5.4 Grouping of the extracted content 42
5.5 OpenKapow Robosuite IDE 43
5.6 Extraction with the RoboSuite 43

6.1 Yahoo! Pipes interface 47
6.2 Pipes debugger 48
6.3 Popfly interface 49
6.4 Popfly result module 49
6.5 Damia editor 51
6.6 QEDWiki user interface 51

7.1 Fm4 radio stream 54

8.1 Three concerns of Web page development and management 58
8.2 Model View Controller 59
8.3 Main components within Cocoon 60

9.1 Sources that will be aggregated 63
9.2 Frame with tracklisting 64
9.3 Amazon's Music portal page 67
9.4 AZLyrics.com - Content source for lyrics 70
9.5 Search results when searching after an artist and title 71
9.6 Artist information on Wikipedia 75
9.7 Extracted artist information from Wikipedia 75
9.8 Result showing embedded videos 77

10.1 Music Extension sidebar inside the Firefox Browser 79

10.2 Artist and video tab 79
10.3 Lyrics and pictures tab 80
10.4 Statusbar displaying current song played 80

12.1 Mashup Timeline - New mashups on ProgrammableWeb, last 6 months 88

Part I

Introduction

1 Introduction

1.1 Motivation

In the last few years web applications have significantly extended their use and value from plain information sources into highly sophisticated desktop-like applications. The so-called Web 2.0 introduced, with user-generated content, a new way of acquiring and providing information and paved the way for a revival of Software as a Service (SaaS). Within Web 2.0, new technologies and models evolved or were explicitly provided to mix data from different sources in order to create *something new*. These hybrid applications became known as *Mashups*.

Mashups initially focused on mixing different data sources together, meanwhile developers, especially in the enterprise sector worked to offer a quick and simple way to create situational business applications.

In this chapter I will give a short introduction as to what a mashup is and what you can expect of it. Categorization will help you get a general idea of what types of mashups do exist, and how enterprise can take advantage of mashups.

Part two discusses the technology that are related to mashups. Chapter two is all about common technology standards that most mashups build upon. The third chapter takes a closer look at the different types of mashups within a technical and logical view.

Chapter four deals with data extraction, a fundamental part of mashups when content is not accessible through a public application programming interface (API). The follow up chapter presents tools that offer a fast-forward approach for web data extraction.

Mashup platforms are all about already existing software products or services that offer mashup functionality. I will have a look at three publically available platforms and discuss their functionality.

In the third part, a case study on creating a music mashup *fmMusic* is presented. fmMusic is a service that extracts the artist and track name from a radio station's streaming service and gathers various additional information, like album name, lyrics, video or artist information. The case study uses the XML framework Cocoon[1], therefore chapter eight will introduce the basic concepts of this, before we examine the functionality behind the mashup in chapter nine.

The frontend that is used for the mashup is an extension for the Firefox browser. Chapter ten explains how to extend the browser and add the mashup functionality to it. The mashup could have been implemented in HTML or something else too, but I favored this way, because I think an extension offers better usability to the user.

The last part discusses future issues related to mashups. The rising popularity of mashups will make them widely available in the enterprise sector. I will have a closer look at enterprise's current role and how these mashup tools will fit within enterprises. The last chapter tries to point out the role of mashups in relation to current copyright law and how mashups are dependent on the content owner, especially if some kind of web scraping mechanism is involved.

[1]http://cocoon.apache.org/2.1/

1.2 Mashups

Mashups are a new sort of interactive web application that use existing content or data sources to create new services or applications. Mashups can be seen as a follow-up involvement to the so-called Web 2.0, where building desktop-like web applications and social interaction was the main focus. Tim O'Reilly defined the principles of Web 2.0 as *simple, low-barrier and fast* and *every user himself is the center of the Internet* [ORei05]. Mashups not only build up on these technologies they go further and integrate different data sources that are either explicitly provided by content owners through APIs, or use web scraping techniques to access content.

The term *mashup* was borrowed from the music scene, where in beginning of this century a new type of remixing evolved. Vocals and instrumental tracks, mainly from different genres, e.g. hiphop and rock, were mixed together to create something new. This quickly became popular as *bastard pop*[2] and is now well known as *mashup*. In an analogy to the music scene, web mashups use existing content, most of the time just parts of it, and mixes it with content from at least one other data source.

Google Maps[3] was, at the time of its release, the most outstanding application, as it showed off what a web application is capable of doing. It changed all the perceptions of how a web application operates in comparison to a desktop client. The spotlight shifted to developing new, AJAX [Garr05] based, applications. With the emergence of AJAX use in these rich Internet applications (RIAs) the usability experience improved and led to a more comfortable user interaction. As more and more web applications started to use these new possibilities and started to motivate users to enhance the information provided, the so-called social networking sites appeared on the scene. A well known example is Flickr[4], a popular photo sharing site. It offers tools to users to tag, manipulate, comment or share photos in a quick and convenient way.

Right after the release of Google Maps, Google provided an application programming interface (API) to access its features. Everyone could integrate data within Google Maps. Los Angeles Times[5] used Google Maps to create a wildfire tracking map[6]. Lots of developers and hobbyists created their own maps. Mapping mashups became the most popular type of available mashups.

1.3 Definition

In my introduction, I defined a mashup as a mix of different content sources to create something with new value, maybe even disconnected from its intended uses and meanings. If I look at this definition, I could easily say that an integration of *Google Adsense*[7] links makes a website a mashup. Placing an ad an a website is not a way to extend the value of a website. Therefore I come to the conclusion that the definition must also include reference to creating something new out of the different content sources.

Someone could ask, is the mapping of a data source onto a map a mashup? As this is one of the initial applications used, of course it is; it's even a special type of mashup,

[2]http://de.wikipedia.org/wiki/bastardpop
[3]http://maps.google.com
[4]http://www.flickr.com
[5]http://www.latimes.com/
[6]http://www.latimes.com/news/local/la-firemap,0,6179739.htmlpage
[7]https://www.google.com/adsense/login/de/

a mapping mashup. Thus what other kinds of mashups are there, and can we define categories for mashups?

1.4 Categories

1.4.1 Mapping mashups

Visual representation and interaction is always a big issue when displaying data and information. Spreadsheets have, for a long time now, provided different types of graphs to visualize data. Mapping data to a map was previously a time-consuming task, as mapping onto the map was done by hand. Interactive maps now offer a new, convenient way to this automatically.

Let's use the example from above, the wildfires in California (Figure 1.1). The common approach would have been to show the places with wildfires in a table or list. If there were a lot of wildfires, there may have been some kind of sorting or filtering mechanism. Additionally there may have been a map where some of the most interesting spots were marked. Depending on the resolution of the image, more or less exact positions could be mapped.

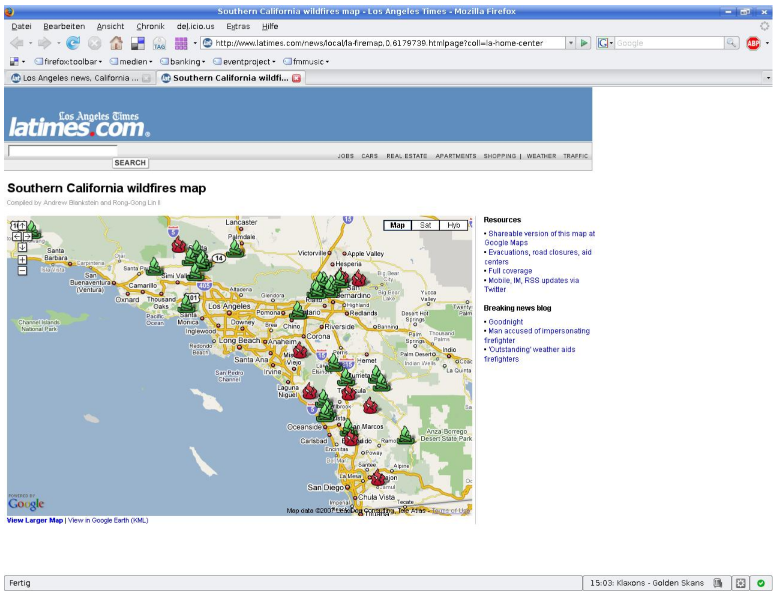

Southern California wildfires map - Los Angeles Times - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik del.icio.us Extras Hilfe
http://www.latimes.com/news/local/la-firemap,0,6179739.htmlpage?coll=la-home-center
latimes.com
SEARCH
JOBS CARS REAL ESTATE APARTMENTS SHOPPING WEATHER TRAFFIC
Southern California wildfires map
Compiled by Andrew Blankstein and Rong-Gong Lin II
Map Sat Hyb
Resources
Shareable version of this map at Google Maps
Evacuations, road closures, aid centers
Full coverage
Mobile, IM, RSS updates via Twitter
Breaking news blog
Goodnight
Man accused of impersonating firefighter
'Outstanding' weather aids firefighters
View Larger Map | View in Google Earth (KML)
Fertig
15:03: Klaxons - Golden Skans

Figure 1.1: Mapping Mashup - California wildfires

Let's examine the advantages of a mapping mashup. If you use a static image, instead of a dynamic map, you always have to update that image, most commonly by hand, to add a new spot to the map, and then upload it. Dynamic maps used in a mapping mashup connect to a web service that delivers the coordinates for each spot that is rendered on the map. A new spot is simply added to the database to which the web service connects and then rendered automatically. Furthermore, these maps offer interactive tools that let the user zoom in/out, navigate in the map, and even find additional information for each spot inside small popups.

The combination of interactivity and information display is what attracts a user. The user is offered a direct connection between what the information is all about and where it is located. The simplicity of using a mapping mashup, its interactivity and the users attraction to visual representations is the power of these maps.

1.4.2 Video and photo mashups

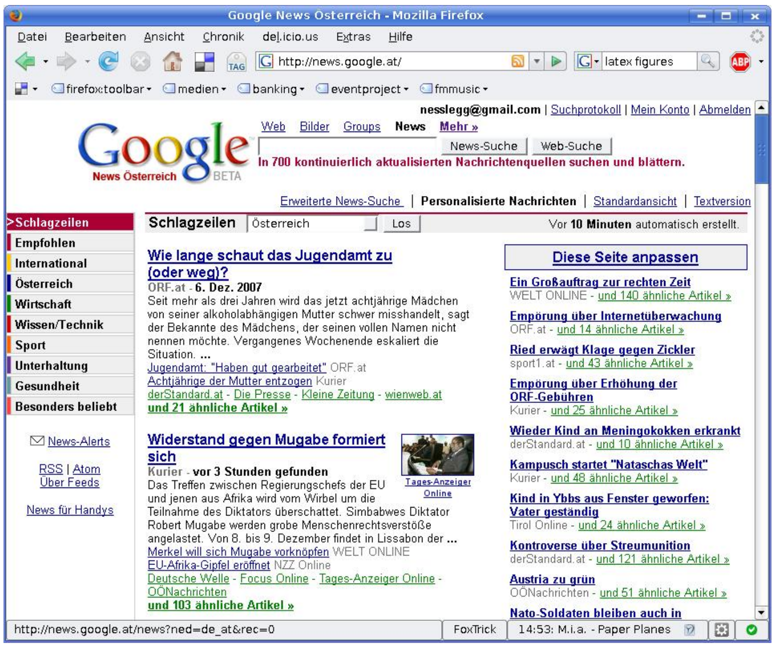
Photo hosting and sharing sites like Flickr introduced the addition of of meta information (creation date, place taken, name of photographer) and especially tagging to pictures. If the image is set to be accessed by the public, it can be accessed and viewed by anyone. Flickr offers a REST (2.5) service where you can search all the meta information and get back a list with links to these pictures.

One can imagine a mashup that could parse the feed of traveller, analyze the meta descriptions to extract the various places the traveller visited (e.g.. London), and mix these with pictures from Flickr that are tagged with the location's name.

1.4.3 Data (News) mashups

Really Simple Syndication (RSS) feeds evolved during the beginning of blogging. RSS allowed people to stay up to date with new entries posted on a users blog. Today most online newspapers and magazines implement news feeds for their recent articles, often categorizing them into different topics like politics, economy, sports, and so on. A news mashup acquires different feeds and joins them together into a custom feed. On this new feed a filter could be applied that filters the links depending on keywords in its description. An example for this could be a selection of news feeds from different sport portals, where the filteris for articles about tennis.

Google News (Figure 1.2) is an example on how a data mashup can be represented.


Google News Österreich - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik del.icio.us Extras Hilfe
http://news.google.at/
latex figures
firefox:toolbar medien banking eventproject fmmusic
Google News Österreich BETA
Web Bilder Groups News Mehr »
News-Suche
Web-Suche
Los
FoxTrick
14:53: M.i.a. - Paper Planes

nesslegg@gmail.com | Suchprotokoll | Mein Konto | Abmelden

In 700 kontinuierlich aktualisierten Nachrichtenquellen suchen und blättern.

Erweiterte News-Suche | Personalisierte Nachrichten | Standardansicht | Textversion

>Schlagzeilen
Empfohlen
International
Österreich
Wirtschaft
Wissen/Technik
Sport
Unterhaltung
Gesundheit
Besonders beliebt

News-Alerts

RSS | Atom
Über Feeds

News für Handys

Schlagzeilen Österreich Los Vor 10 Minuten automatisch erstellt.

Wie lange schaut das Jugendamt zu (oder weg)?
ORF.at - 6. Dez. 2007
Seit mehr als drei Jahren wird das jetzt achtjährige Mädchen von seiner alkoholabhängigen Mutter schwer misshandelt, sagt der Bekannte des Mädchens, der seinen vollen Namen nicht nennen möchte. Vergangenes Wochenende eskaliert die Situation. ...
Jugendamt: "Haben gut gearbeitet" ORF.at
Achtjährige der Mutter entzogen Kurier
derStandard.at - Die Presse - Kleine Zeitung - wienweb.at
und 21 ähnliche Artikel »

Widerstand gegen Mugabe formiert sich
Kurier - vor 3 Stunden gefunden
Das Treffen zwischen Regierungschefs der EU und jenen aus Afrika wird vom Wirbel um die Teilnahme des Diktators überschattet. Simbabwes Diktator Robert Mugabe werden grobe Menschenrechtsverstöße angelastet. Von 8. bis 9. Dezember findet in Lissabon der ...
Tages-Anzeiger Online
Merkel will sich Mugabe vorknöpfen WELT ONLINE
EU-Afrika-Gipfel eröffnet NZZ Online
Deutsche Welle - Focus Online - Tages-Anzeiger Online - OÖNachrichten
und 103 ähnliche Artikel »

Diese Seite anpassen

Ein Großauftrag zur rechten Zeit
WELT ONLINE - und 140 ähnliche Artikel »
Empörung über Internetüberwachung
ORF.at - und 14 ähnliche Artikel »
Ried erwägt Klage gegen Zickler
sport1.at - und 43 ähnliche Artikel »
Empörung über Erhöhung der ORF-Gebühren
Kurier - und 25 ähnliche Artikel »
Wieder Kind an Meningokokken erkrankt
derStandard.at - und 10 ähnliche Artikel »
Kampusch startet "Nataschas Welt"
Kurier - und 48 ähnliche Artikel »
Kind in Ybbs aus Fenster geworfen: Vater geständig
Tirol Online - und 24 ähnliche Artikel »
Kontroverse über Streumunition
derStandard.at - und 121 ähnliche Artikel »
Austria zu grün
OÖNachrichten - und 51 ähnliche Artikel »
Nato-Soldaten bleiben auch in

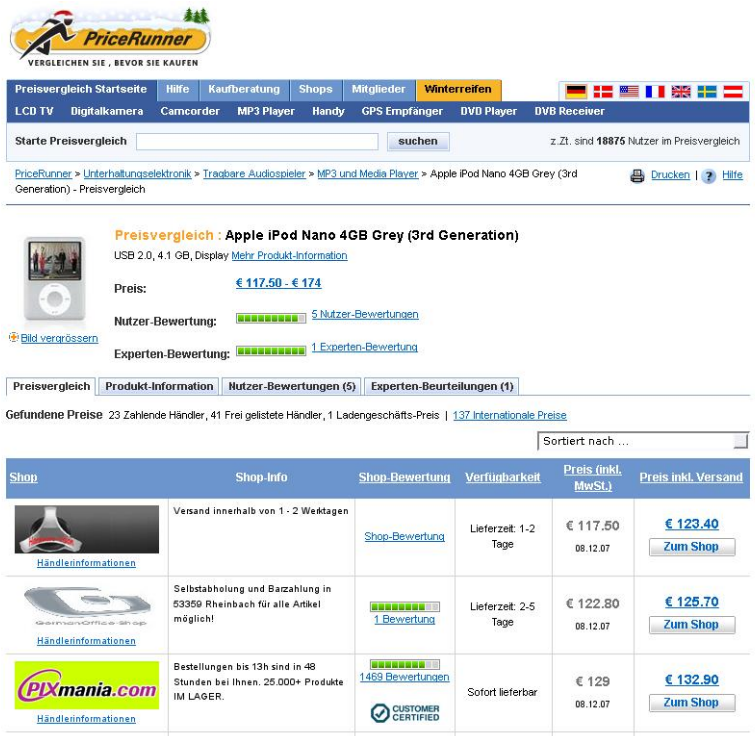
http://news.google.at/news?ned=de_at&rec=0

Figure 1.2: Google News - a data mashup

1.4.4 Shopping mashups

Shopping mashups, like price comparison sites, existed a long time before the term mashup was commonly used. The main concept behind a shopping mashup is to compare prices from different dealers to get the cheapest offer available. Famous examples of such sites are Pricerunner[8], Geizhals[9] or Froogle[10]. Long before standards like SOAP or REST evolved, these sites used various screen scraping techniques, proprietary formats or input forms to acquire data for its comparisons.

Meanwhile, companies like Amazon[11] or Ebay[12] are offering APIs to allow access to their product data. Of course they do not do this without reason. They look for additional revenue generated by web sites that use their data and link back to them. The linking sites then get a fraction of the revenue as commission. Figure 1.3 shows a search result page of Pricerunner.



Starte Preisvergleich

z.Zt. sind **18875** Nutzer im Preisvergleich

PriceRunner > Unterhaltungselektronik > Tragbare Audiospieler > MP3 und Media Player > Apple iPod Nano 4GB Grey (3rd Generation) - Preisvergleich

Preisvergleich : Apple iPod Nano 4GB Grey (3rd Generation)

USB 2.0, 4.1 GB, Display Mehr Produkt-Information

Preis: € 117.50 - € 174

Nutzer-Bewertung: 5 Nutzer-Bewertungen

Experten-Bewertung: 1 Experten-Bewertung

Gefundene Preise 23 Zahlende Händler, 41 Frei gelistete Händler, 1 Ladengeschäfts-Preis | 137 Internationale Preise

Shop	Shop-Info	Shop-Bewertung	Verfügbarkeit	Preis (inkl. MwSt.)	Preis inkl. Versand
Händlerinformationen	Versand innerhalb von 1 - 2 Werktagen	Shop-Bewertung	Lieferzeit: 1-2 Tage	€ 117.50 08.12.07	€ 123.40 Zum Shop
GermanOffice-Shop Händlerinformationen	Selbstabholung und Barzahlung in 53359 Rheinbach für alle Artikel möglich!	1 Bewertung	Lieferzeit: 2-5 Tage	€ 122.80 08.12.07	€ 125.70 Zum Shop
PIXmania.com Händlerinformationen	Bestellungen bis 13h sind in 48 Stunden bei Ihnen. 25.000+ Produkte IM LAGER.	1469 Bewertungen CUSTOMER CERTIFIED	Sofort lieferbar	€ 129 08.12.07	€ 132.90 Zum Shop

Figure 1.3: Pricerunner - Showing pricecomparisons for a product

1.5 Enterprise mashups

1.5.1 Definition

Enterprise application development is increasingly becoming involved in mashups. Mashups for the enterprise aim to integrate data from internal and external sources. You

[8] http://www.pricerunner.at/
[9] http://www.geizhals.at
[10] http://froogle.google.com/
[11] http://www.amazon.com
[12] http://www.ebay.com

may think this is nothing new, as this has been done or a long time within the Enterprise Information Integration (EII) context. *EII is the integration of data from multiple systems into a unified, consistent and accurate representation geared toward the viewing and manipulation of the data* [InCo04]. This traditional approach does acquire data from an external content provider, transforms and stores the data inside the company's databases for internal processing. Most commonly this is done on a periodical request base. This process can lead to heavy data and/or information processing and easily lead to outdated information.

When using enterprise mashups, the external data will not be handled internally anymore; web services connect to the external provider and leave the processing there. The enterprise itself can focus on its core functions. You can think about services like a geocoding or global maps, where it would never will be profitable for a company to develop such services by itself.

The need for *Situational Applications*, (i.e applications that come together for solving some immediate business problems) are one set of tasks [Jhin06] that currently are not well served by EII. Mashups for the enterprise try to address issues like creating applications that address a specific problem and are used for just a short time. Traditionally such applications would not be developed by the IT department because the cost for development would be much higher than its usefulness. Mashup platforms allow a business user to create such situational applications as they come with modular modules that can simply be connected when necessary.

Meanwhile you can find more and more mashup enablers/providers trying to enter the enterprise section with their solutions. Services like Yahoo! Pipes[13], Snap Logic[14], Open Kapow[15] or Dapper[16] offer functionality to convert (web) data into structured XML, feeds, widgets or even offer special primitives for further processing.

The concept of a three layer mashup composition for the enterprise is proposed by A. Jhingran [Jhin06] to cover aspects of data, aggregation and presentation.

1. **Ingestion**: The layer that accesses all structured and unstructured data and provides services to access these data. Common screen scraping technologies or products or predefined webservices would fall into this layer.

2. **Augmentation**: Here some basic operations like union, fusion or standardization take place. You can think of filtering a sports feed to return only news stories about tennis.

3. **Presentation**: The final presentation for the end user wether a HTML page, Atom/RSS or web service.

The online compendium ProgrammableWeb.com[17] is a good resource to find APIs that can be used within a mashup. Looking at the continuous growth of available services it is not a matter of *if* companies start using enterprise mashups, but when.

[13]http://pipes.yahoo.com
[14]https://www.snaplogic.org/
[15]http://www.openkapow.com
[16]http://www.dapper.net
[17]www.programmableweb.com

1.5.2 Service Compositions and Mashups

Service oriented architecture (SOA) has become very popular in the last few years. The paradigm is to use services as basic constructs of applications. It changed the development approach from traditional *product-centric* manufacturing, to *consumer-centric* service composition. This allowed users to build new business processes, applications or solutions in a rapid and low-cost way across heterogeneous environments.

[LiuX07] lists common drawbacks of current SOA technologies and points out where mashups for the enterprise needs to lead to overcome those. Current service composition programming is mainly designed for professional SOA developers to solve complex business problems in the enterprise's IT environment. Though these technologies are powerful, there are still some important issues:

- These technologies involve requirement overhead with respect to developer's skill and supporting infrastructure. Major effort is spent to master many different SOA technologies (BPEL, WSDL, UDDI), as well as tools and runtime servers.
- Service compositions cannot be done on the fly. Mostly IDE tools are needed for customization of those services and they need to be deployed to runtime servers. After deployment, composition logic is hard to customize.
- These technologies cannot support the composition of legacy or existing web applications which don't or can't provide web service interfaces.

These issues are the barriers to wider adoption of SOA in the enterprise. Mashups in the sense of a Web 2.0 paradigm should allow **the consumer** to create, add and adapt services. What is already possible on the world wide web should be incorporated inside enterprises as well. Users with low programming skills should address end-users needs for flexible composition and customization.

The new technologies should provide support for:

- The browser as design-time and runtime tool for service composition, reducing the mentioned overhead of tools and server runtimes.
- Flexible customization and deployment to overcome strict IT deployment rules and allow quicker response to business needs.
- Ease of reuse and composition of already existing services and external accessible data like web services.

Part II

Technology

2 Technologies

2.1 XML

XML- eXtensible Markup Language is a W3C-endorsed [W3Co08] standard for document markup. The standard defines a generic syntax to provide a human readable presentation of data. It is a general purpose format on which other many other standards build up on. XML is used in a broad sense for:

- interchange format
- vector graphics
- document format
- syndication format
- remote procedure calls
- configuration files
- voice mail systems
- user interface description
- and lots more....

The following citation out of [Haro01] describes what someone can expect of XML.

> *Most importantly, XML is a meta-markup language. That means it doesn't have a fixed set of tags and elements that are always supposed to work for everyone in all areas of interest. .... Instead, XML allows developers and writers to define the elements they need as they need them.*

An XML document consists of elements and attributes and is structured in a hierarchical structure. The topmost element is called the root node that can exists of as many child elements. Each element can have n-attributes and n-child elements. An attribute is a name-value pair to an element.

A short introduction to XML, what its defined for, its technology and its possibilities comes from the World Wide Web Consortium (W3C) in it's article XML in 10 points [W3CC99].

1. is for structuring data
2. XML looks a bit like HTML
3. XML is text but isn't meant to be read

4. XML is verbose by design
5. XML is a family of technologies
6. XML is new, but not that new
7. XML leads HTML to XHTML
8. XML is modular
9. XML is the basis for RDF and the Semantic Web
10. XML is license-free, platform-independent and well-supported

2.1.1 A sample XML Document

Let's have a look at a example document. In listing 2.1 you find the description of an album by a band. Looking at the XML you can easily identify whats the band name, which title the album has and what year it was created.

Listing 2.1: XML example for an album description

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<album @year="2004">
    <artist>Metallica</artist>
    <title>St. Anger</artist>
</album>
```

One of the requirements to an XML document is that is has to be well-formed, what means it has to fulfill following criteria [W3Co06]:

1. Taken as a whole, it matches the production labeled document.
2. It meets all the well-formedness constraints given in this specification.
3. Each of the parsed entities which is referenced directly or indirectly within the document is well-formed.

Second it has to be well structured. Unlike HTML, each element has to consist of a start and an end tag. Empty tags may be used for elements without any contents.

2.1.2 Validation

The structure of an XML file can be defined externally to allow a validation of its structure and content. A document type definition (DTD) was the first way to provide a description of an XML file and comes with a proprietary syntax. Figure 2.2 represent the example from above. Meanwhile a second standard to describe an XML evolved, XML Schema .

XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents. in more detail. [Cmsp00]

XML Schema does not use it's own syntax, but uses XML. Schema extends the possibilities of DTD's and allows much stricter validation. E.g.. schema allows to define datatypes or the use of regular expressions on element content.

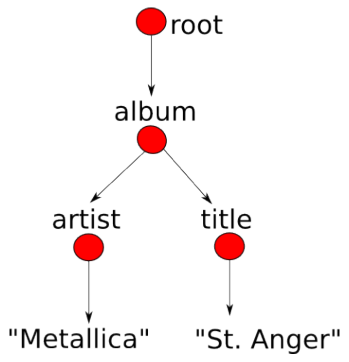


Figure 2.1: XML document represented as tree

Listing 2.2: DTD for the album example

```
<!ELEMENT album (artist, title) >
<!ELEMENT artist (#PCDATA) >
<!ELEMENT title (#PCDATA) >
<!ATTLIST album year CDATA #REQUIRED>
```

2.2 XPath

When you are dealing with XML files you most probably want to do something with it, most probably access some information that is stored in the document. That's when XPath comes in. The standard recommendation [W3Co99a] defines XPath as follows:

> The primary purpose of XPath is to address parts of an XML document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document.

XPath is capable to do a vary of tasks [Poor06]:

- Location of parts of an XML document.
- Navigating through an XML document and selecting parts of it.
- Performing complex node manipulation through built in functions.

The principle of XPath is its view of the XML document as a tree [Chau2002] with branches called nodes. Figure 2.1 represents the album example from above.

2.2.1 How to use XPath

Looking at the example above, following XPath selects the artist name of the defined album.

```
/album/artist
```

Next statement selects the title of the album where the artist is Metallica:

```
/album[artist='Metallica']/title
```

2.3 XSL Transformations (XSLT)

The eXtensible Stylesheet Language (XSL) consist of two parts: The XSL Transformation (XSLT) and XSL Formating Objects (XSL-FO). XSLT is an XML application that specifies rules by which one XML document is transformed into another. XSL-FO describes the precise page layout for rendering data to the portable document format (PDF).

2.3.1 Usage of XSLT

A common task when handling XML documents is to transform it from one form into another. The resulting format may again be XML but any other text based format (e.g CSV, JSON) is possible as well. XSLT is used in various ways, some examples would be

Common situations when you would use XSLT [Frit03]

- Transforming an XML document into an HTML or XHTML document for display in a web browser.
- Converting from one markup vocabulary to another, such as from Docbook (http://www.docbook.org) to XHTML
- Extracting plain text out of an XML document for use in a non-XML application or environment
- Building a new German language document by pulling and repurposing all the German text from a multilingual XML document

In common business-to-business (B2B) situation data from a business partner needs to be transformed to match the internal document structure or database schema. Within modern web browsers XML documents with attached XSL Stylesheets are rendered on the fly as HTML documents. Statistical data can be transformed into SVG and then be rendered on SVG-supporting applications or e.g. rendered to images (with the open-source library Batik[1]).

A lot of APIs provide a REST (Section 2.5) interface. In the case study we will transform the data to another structure with XSLT.

2.4 Web Services

Over the last years web services draw amendous attention to it. What once was a hype now is commonly used all over. Nearly all big players in the IT-business integrate web services into their products. If you search for a definition of web services you will find a lot of different definitions, however web services mainly provide a systematic and extensible framework built on top of existing protocols and based on open XML standards for system independent interaction [Curb02]. The W3C describes it as follows. [Lafo07]

[1]http://xmlgraphics.apache.org/batik/

Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions thanks to the use of XML. They can be combined in a loosely coupled way in order to achieve complex operations. Programs providing simple services can interact with each other in order to deliver sophisticated added-value services.

More and more companies publicly offer access to some functionality through web services. Companies like Google [2], YouTube [3] or Amazon [4] provide access to their services. Amazon for example not just provides access to their product lines, it extended its business area from a pure e-commerce company to a an IT service provider. Amazon offers a service to store and retreive data (Amazon S3, a queue for storing message in queues (Amazon Simple Queue Service) and a few others. With these services Amazon allows developers to use their massive infrastructure to build scalable applications with it. Amazon is a good example on how powerful web services can be used.

2.4.1 Where to find web services

Most companies who expose web services place a link most commonly called API or Developer to the descriptions on what services they offer and how to use them. Another good place to discover web service providers is programmableweb[5]. At the time of writing this, there are 2555 mashups and 559 APIs in about 50 categories listed. Web services play an important role, as they are the building blocks of mashups.

2.4.2 How does it work

Request/Response Messaging

Most commonly web services are used to acquire information from another system. The client is sending a request to a web service provider and waits for the response of it (Figure 2.2).

A common example is a e-commerce site that needs the credit card number to be verified. The site is requesting the credit card company, if the given values (name, number, valid to) are correct, and the customer can be charged on his card. The e-commerce site, as requestor, is sending the values to the web service of the credit card company. The credit card company verifies these values and then sends back a response, either if the customer can be charged or not.

One way Messaging

Web services do not automatically expect a response from the web service provider. There are situations when you are not interested in receiving a response. So the client just emits a request (Figure fig:request) An example for such a situation would be if a service A is dependent of the processsing of another service B before it can start processing. When service B is at the stage where service A can start processing, it can

[2]http://www.google.com/apis
[3]http://www.youtube.com/dev
[4]http://aws.amazon.com
[5]http://www.programmableweb.com

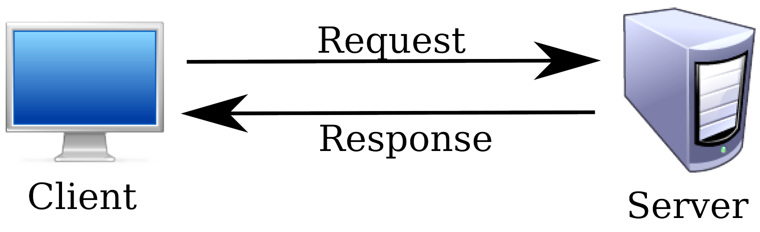


Figure 2.2: Request/Response web service

send a request to a listener, that is implemented as web service, to inform service A that it can start.

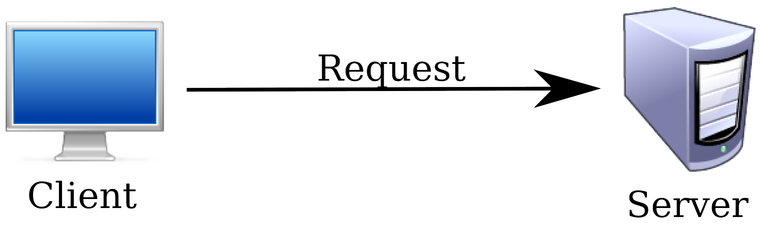


Figure 2.3: One way web service

2.4.3 Implementations

Most common web services are directly associated with Simple Object Access Protocol (SOAP), a widely used standard for exchanging structured information, however Representational State Transfer (REST) is getting more and more attention in developing web applications. It is not a standard but an architectural style for building network-based systems. Tim O'Reilly blogged that within Amazons Web Services, while it offers SOAP and REST services, 85 percent use REST as the preferred web service interface [ORei03].

2.5 Representational State Transfer

In the music mashup a few content sources,like Amazon, Flickr and Youtube provide APIs to access their information. These APIs provide Representation State Transfer (REST) interfaces to query their database and we are going to use these.

2.5.1 Definition

Representational State Transfer is an architectural style for building network-based systems described by FIELDING [Fiel00]. REST does not focus on the implementation and syntax, but on the roles of components, the constraints of interaction and the interpretation of the data. It is an abstraction of architectural elements: data, connectors and components. This goes back to the beginning of the early web architecture, prior to 1994, where client-server architecture was mainly a stateless exchange of static data.

Unlike transactional systems, in which servers maintain session handling between server and client, the web interaction mainly exists of a request, where the response is a pre-computed set of data in a representational state. The client can easily navigate between resources by following links from one resource to the next.

In [Rich07] three kinds of resources are distinguished.

- predefined one-off resources such as a service's homepage or a static list of links to resources.
- a large (possibly infinite) number of individual items of data, this might be an object in an object oriented system, or a database row in a database system.
- a large (probably infinite) number of resources corresponding to the possible outputs of an algorithm, e.g. the outcome of a database query.

2.5.2 Design principles of REST

- separation from user interface concern from data storage concern, what improves scalability
- resources are identified by URIs
- interactions are stateless, requests must contain all information's that can be processed.
- cachability of response to improve network efficiency
- data is selfdescriptive, containing descriptive information
- emphasis on a uniform interface, information is transferred in a standardized manner
- hypermedia as the engine of application state, there are no services, just resources, navigation through hyperlinks

2.5.3 Methods of REST

Http 1.0 comes with methods GET, PUT, POST and DELETE, while HTTP 1.1 allows extensions. This is similar to SQL, where SELECT, UPDATE, INSERT and DELETE are the main commands used to manipulate data in a table. The rest is a set of filters and transformers to manipulate the data. In a web context this methods are usually enough to interact with a server (data manipulation can e.g. made with JavaScript). In REST a request to an URI results in a representation of an object. This document provides the client with the opportunity to navigate to a different URI to change the state of the object.

2.5.4 Using YouTube's REST API

YouTube allows to query their content through a REST API, that we will use. Lets have a look on how YouTube has implemented REST. What we need is to search for a music video of the track that is currently played.

Searching for videos

The API comes with a set of options to limit the results to match a specific criteria. It allows to query a specific category, search for tags or query after keywords.

To search for a videos you use submit a request to:
http://gdata.youtube.com/feeds/api/videos

The following is a list of the most common query parameters used in searches [YouT07]

- *alt* - The format of feed to return, such as atom (the default), rss, or json.
- *orderby* - The order in which to list entries, such as relevance (the default for the videos feed) or viewCount.
- *start-index* - The 1-based index of the first result to be retrieved (for paging).
- *max-results* - The maximum number of entries to return at one time.
- */-/categories* - The categories and/or tags to use in filteA.1ring the feed results. For example, feedURL/-/fritz/laurie returns all entries that are tagged with both of the user-defined tags fritz and laurie.
- *vq* - A search query term. Searches for the specified string in all video metadata, such as titles, tags, and descriptions.
- *format* - A specific video format. For example, format=1 restricts search results to videos for mobile devices.

So if we want to query after the track Karma Police from artist Radiohead in category music and limit it to the first 5 result we would use following request. The response is of the request is listed in listing A.1.

```
http://gdata.youtube.com/feeds/api/videos/-/Music?vq=Karma+Police+
Radiohead&max-results=5&orderby=viewCount&alt=rss
```

For more information about query parameters, see the YouTube Data API Reference Guide and the Google Data APIs Reference Guide.

2.6 SOAP

The Simple Object Access Protocol (SOAP) is a lightweight protocol that for exchanging information over a network. SOAP is a specification standardized by the W3C and is available in Version 1.1 and 1.2 and is now the most widely supported protocol for use with XML Web Services, hence the SOAP acronym is frequently referred to as the Service-Oriented Architecture Protocol, instead of the Simple Object Access Protocol.

The SOAP specification is built on XML technologies and defines a messaging framework to provide a messaging construct that can be used on a variety of frameworks, however at the moment just HTTP is widely used. It establishes a standard message format that consists of an XML document capable of hosting RPC and document-centric data.

Web services are a fundamental concept in the service-oriented architecture (SOA). In the enterprise sector most of the integration applications support SOAP and WSDL for composing web services. Service-oriented architectures are centered around services.

According to [ErlT04] services are:

- *loosely coupled* - services are self-contained and self-managing.
- *abstract* - the functionality of the underlying service is irrelevant to the outside
- *composable* - services may compose other services, what promotes reusability and granularity.
- *autonomous* - control and function of the services are within an explicit boundary, and is not dependent on other services.
- *stateless* - services should not mangage state information to remain the ability to be loosly coupled.
- *discoverable* - the descriptions of services should be found by external service requestors

2.7 SOAP vs. REST

We now introduced REST and SOAP, which are two opposing standards of web services. Even if these two standards are not related, SOAP is a general protocol for exchanging messages and REST is an architectural style [Zhan04], both are often compared directly and often end in a serious debate which approach is the better one.

[Mueh04] classify upcoming workflow standards like BPEL4WS into two categories: REST-oriented and SOAP-oriented standards.

- REST-oriented integration is abstracting the principles the makes the World Wide Web scaleable, and unlike transactional systems does not need to maintain complex sessions, but rely on a simple response.
- SOAP-oriented integration however relies on WSDL to describe the endpoints of the communication and SOAP to provide the messaging standard. Unlike REST pure SOAP solutions do not distinguish between the process factory and process instances.

In these debate many REST campaigners argue with FIELDING's critique of SOAP

> *In order for [the next generation SOAP protocol] to succeed as a Web protocol, it needs to start behaving like it is part of the Web. That means, among other things, that it should stop trying to encapsulate all sorts of actions under an object-specific interface. It needs to limit its object- specific behavior to those situations in which object-specific behavior is actually desirable.*

2.8 AJAX

2.8.1 Introduction

Ajax is one of the main technologies, or better said a composition of different technologies that mashups build upon on. Again, even before the mashup hype, Google Maps or GMail[6] were the initiator of the hype that started around AJAX [Cran05]. Everyone was asking himself: "How do they do it". Someone could click on the map, drag it around, and the missing parts of the maps were immediately downloaded, without the need of refreshing the whole page. This was a whole new way of working with web applications. The web started to behave like a desktop application. Even if Google Maps was not the first one to integrate the technology, it was the first who pushed the limits of web applications. It then became the underlying technology of the so-called Web2.0, of what we will here more later on.

2.8.2 Definition

AJAX is an acronym that stands for Asynchronous JavaScript And XML and became well know after appearing in an article by Jesse James Garret. In this article [Garr05] Garret defined Ajax as follows:

- standards-based presentation using XHTML and CSS;
- dynamic display and interaction using the Document Object Model;
- data interchange and manipulation using XML and XSLT;
- asynchronous data retrieval using XMLHttpRequest;
- and JavaScript binding everything together.

AJAX, even if it is often referenced as technology, is not one. It is using well established technologies previously known as Dynamic HTML (DHTML) and remote scripting [Mesb06], and integrates these to a full messaging and presentation framework for a better user experience [Wild07]. Lets have a look at the parts of AJAX.

(X)HTML HypeText Markup Language (HTML) defines the structure and elements of a webpage and is the most common standard used to represent data and text on the world wide web. Extensible HyperText Markup Language (XHTML) is the successor of HTML. It was created as a reformulation of HTML 4 in XML 1.0 [W3C02]. Whereas HTML allowed inconsistent programming and led to representation differences in user agents, XHTML comes with stricter rules to constrain to more uniformity.

CSS Cascading Style Sheets (CSS) was created for adding style to web documents. It was an approach to separate the style of a document from its structure. Through this content and style can easily edited separately. In the context of AJAX CSS is used for the representation of the data. Usually parts of the document are adapted or exchanged. With the use of CSS, there is no need to submit the style in the request, it therefore limits it to the actual content. This limits network traffic and leads to better rendering performance.

[6]http://gmail.google.com

XML We already introduced XML before. XML is used for the data exchange between client and server and is well supported on most clients. Meanwhile JSON, a more lightwight format is also becoming popular.

XSLT XSLT is an optional part of AJAX, but often used too. XSLT has its right to again limit the network traffic and processing resources on the server. Lets have a quick look on how this can be achieved. Example

DOM The Document Object Model (DOM) presents the structure of a HTML or XML document as a tree of objects that can be manipulated through JavaScript.

XMLHttpRequest The XMLHttpRequest-Object [Kest07] is one of the fundamental parts of AJAX. It allows the client to communicate with the server in an asynchronous way (synchronous is possible too) to retrieve data as a background activity. Through this object HTTP-requests can be sent and received without having the client to refresh the whole page. This provides more interactiveness in the client application.

JavaScript JavaScript is a client side scripting language that was introduced by Netscape in the year 1996. It was initially created for manipulating web sites within the web browser dynamically. Meanwhile JavaScript is implemented in few other applications, Mozilla Firefox, Adobe Acrobat or Yahoo! Widgets just to name a few. We will see that it Rhino, another open source JavaScript engine comes with Apache Cocoon Withing AJAX JavaScript is what holds all together. It is used to control application logic and manipulates the content and representation. It is used to control the XMLHttpRequest object, manipulate the DOM elements, do XSL transformations and react on the user generated events.

2.8.3 AJAX Model

The introduction of AJAX in modern web applications changed the classic web application model. Figure 2.8.3 [Garr05] tries to illustrate the difference between the functionality of the two models.

In the traditional model every interaction is generating an HTTP-request that requests a new HTML page from the server and renders a new page on the browser. In an AJAX enabled application each interaction is routed to the AJAX engine, that decides if new data has to be loaded from the server, if yes, starts an asynchronous request using the XMLHttpRequest-Object. The data received from the server is then rendered directly within the current page. This procedure allows to add or update parts of a document without refreshing the whole document. As the request is done asynchronously in the background, the user is not limited to wait for the response, what results in a far better user experience.

In figure 2.5 [Garr05] the synchronous process flow of the traditional concept is compared to the asynchronous. In the synchronous process the client requests a site from the server. The server is processing the request and sends back the response. Meanwhile the user interactivity is stopped until the response again is rendered on the client. In the asynchronous process flow the requests are processed by the AJAX engine which processes the request to the server. In comparison to the traditional model, not the whole page is refreshed, but just the actual needed data is requested and placed inside the current web site. The user is during this request not limited in his activities.

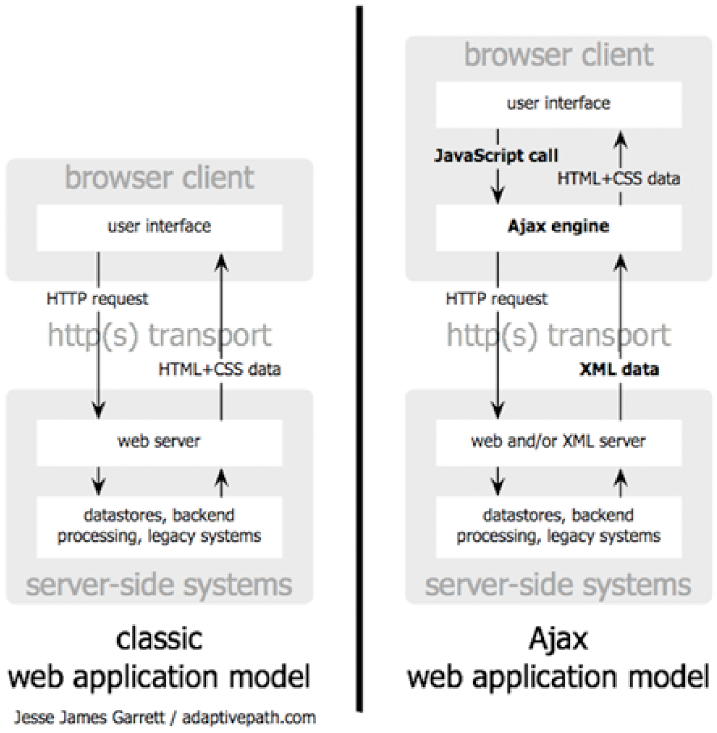


Figure 2.4: Web application model comparison

2.9 JSON

Javascript Object Notation (JSON) [Croc06] is a language independent data format for specifying JavaScript objects so that they can easily be transported over the network. JSON is a was created in 2002 as a cleaner and lighter alternative to XML [Mahe06]. It was derived from the ECMAScript Standard. JSON is commonly supported by JavaScript it is more and more used within AJAX application as replacement of XML.

JSON is built o two structures [7]

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

So how does a JSON Object looks like.

An object is an unordered set of name/value pairs. An object begins with (left brace) and ends with (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma). [8]

Listing 2.3: JSON Example

```
{
    "album": {
```

[7]www.json.org
[8]www.json.org

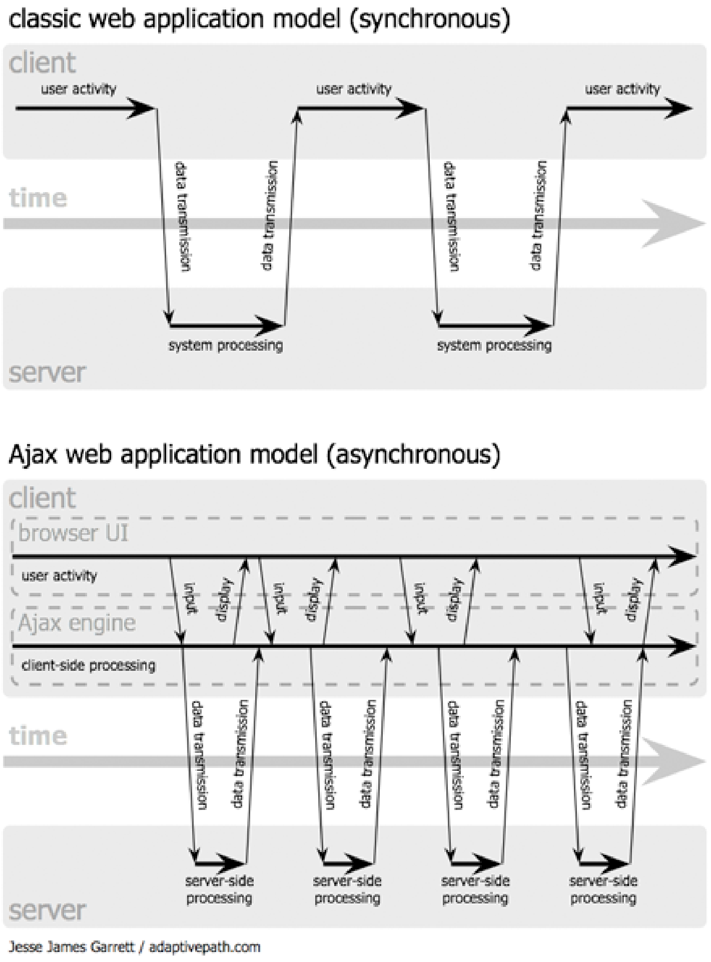


Figure 2.5: AJAX process flow

```
          "year": "2004",
           "artist": "Metallica",
          "title": "St. Anger"
     }
}
```

again the same presented in XML.

Listing 2.4: JSON Example presented in XML

```
<album @year="2004">
    <artist>Metallica</artist>
    <title>St. Anger</artist>
</album>
```

JSON, even it is quite new, is getting more and more attention by developers. In comparison to XML, JSON is maybe a bit harder to read (at least in my personal opinion), but has its clear advantage in using much less bandwith on the network. Using less bandwith directly leads in a better performance of the application, and of course less costs for traffic on the server. The drawback of JavaScript is the inferior support in development IDEs, no simple validation option or client side XSL-transformation.

2.10 XUL

The XML User Interface Language (XUL)[9] is a language to describe the structure of an application. In relation to XML, where XML is used to describe a document structure, XUL can be described as using XML in the field of graphical user interfaces. [Prot07]

XUL derived of the development of the Netscape Navigator and is now most well known as GUI for the Mozilla browser Firefox and mail client Thunderbird. The advantages of XUL is the abstraction of the user interface from the application logic. Microsoft introduced with the Extensible Application Markup Language (XAML) [10] a similar concept. Within the context of mashups, it is not a technology on which a mashup relies on, it is one way to present the mashup on a client. Whereas most mashups rely on HTML as their rendering language, the music mashup will be used as an extension for the Mozilla Firefox web browser.

XUL itself does not implement any functionality, it is just a way to describe where and how elements in an application are placed. In a way it is probably best to compare it with HTML. In HTML you use HTML-Tags and set their content, place and style it through CSS and manipulate it through JavaScript. The same principle lies behind XUL. You define the structure of the application and use CSS and JavaScript for the design and application logic. Extended functionality is provided through XPComm technology.

2.10.1 Features and Benefits

XUL and its related technologies offer a wide functionality and that makes it possible to build powerful cross-platform applications. Following we point out an overview of the features and benefits of XUL [MozD07].

Widget based markup language XUL is a markup language that is designed towards building cross-platform applications. It provides elements, such as windows, labels, buttons, etc usually used in applications. It targets at developers that try to developers that try these typical application functionality with DHTML, at the costs of performance and complexity.

Based on existing standards XUL builds up on XML 1.0. Other standard technologies are HTML 4.0, CSS, Document Object Model (DOM) and Javascript.

Platform independent XUL comes with the promise of write-once, run anywhere. Currently Mozilla applications run on nearly all common platforms available and therefore XUL applications using standard XUL components can be run there too.

Separation of presentation from application logic XUL implements a clear separation of programmatic logic ("content" consisting of XUL, XBL and JavaScript), presentation ("skin" consisting of CSS and images) and language-specific text labels ("locale" consisting of DTDs and string bundles in .properties files).

[9]https://www.mozilla.org/projects/xul/

[10]http://msdn2.microsoft.com/en-us/library/ms752059.aspx

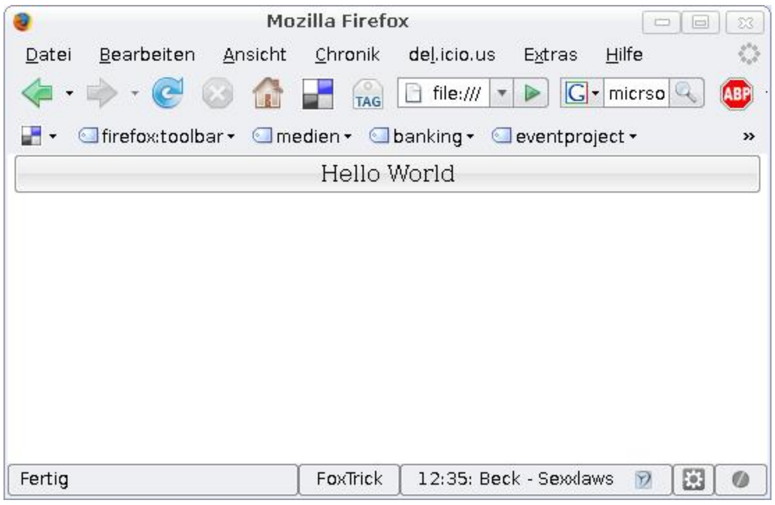


Figure 2.6: XUL Example opened in Firefox

2.10.2 Requirements

If you want to run a XUL applications you either need to install one of the Mozilla products like Firefox or Thunderbird and associate the XUL-file with one of the programs, which work on nearly all platforms.

Lets have quick look on a *Hello World* XUL example. A more advanced example reflst:sidebarXUL is shown in the case study, where a tabbed interface is created.

Listing 2.5: Hello World Example

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/xul.css" type="text/
    css"?>

<!DOCTYPE window>
<window id="main-window" xmlns:html="http://www.w3.org/1999/xhtml
    "
          xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.
              is.only.xul">
<button value="Hello World"/>
</window>
```

Saving the file and open it with Firefox will render a Button with the value "Hello World" shown in Figure 2.6..

2.11 RSS / ATOM

2.11.1 RSS

RSS (Really Simple Syndication) is a simple XML-based data used for content syndication. Content syndication, or *feed* within the scope of Internet is to make parts or all content of a site available in a standardized way [Hamm03]. Content such as news headlines, podcasts or blog entries are common examples that are available as such feeds. Different versions of RSS exists, however RSS 2.0 is now widely adopted, as it a simplyfication of its predecessors. Feeds are meanwhile more than just a specification for content syndication, people start to building services that only output to a feed without refering to a existing site. *Yahoo! Pipes* (section 6.4.1), a mashup platform that we discuss later on, uses the feed standard for internal processing and main output. Modules allows to extend the basic XML schema without modifying the core RSS specifications, this is accomplished through declaring an additional namespace. OpenSearch[11] is an example of such an extension, Podcasts are another format, that extends RSS.

Listing 2.6 lists an example of an RSS feed.

Listing 2.6: Example of an Atom 1.0 Feed

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<rss version="2.0">
  <channel>
    <title>Some title</title>
    <link>http://linktoFeed</link>
    <description>description of feed</description>
    <language>en-us</language>
    <pubDate>Tue, 10 Jun 2003 04:00:00 GMT</pubDate>
    <lastBuildDate>Tue, 10 Jun 2003 09:41:01 GMT</lastBuildDate>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <generator>some generator</generator>
    <managingEditor>editor@example.com</managingEditor>
    <webMaster>webmaster@example.com</webMaster>

    <item>
      <title>Title of first article</title>
      <description>Short description</description>
      <link>Link to article</link>
      <author>Author name</author>
      <guid>unique identifier</guid>
    </item>
  </channel>

</rss>
```

2.11.2 Atom

The Atom Syndication Format and Publishing Protocol (Atom) initially was created by Sam Ruby, a programmer at IBM as a result of the continous in-fighting within the RSS fraction. He started a fresh approach on what a syndication feed should be and started an open discussion on a wiki [Atom08]. Meanwhile the standard is released under the Creative Commons Attribution/Share Alike license.

Atom applies to a pair of related standards. One for syndication and one for publishing and editing Web resources. Listing 2.7 lists an example document of Atom 1.0.

[11] http://www.opensearch.org/Specifications/OpenSearch/1.1

Listing 2.7: Example of an Atom 1.0 Feed

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

 <title>Example Feed</title>
 <subtitle>A subtitle.</subtitle>
 <link href="http://example.org/feed/" rel="self"/>
 <link href="http://example.org/"/>
 <updated>2008-12-13T18:30:02Z</updated>
 <author>
   <name>Max Mustermann</name>
   <email>max@mustermann.com</email>
 </author>
 <id>urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6</id>

 <entry>
   <title>Atom-Example</title>
   <link href="http://example.org/2003/12/13/atom03"/>
   <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
   <updated>2003-12-13T18:30:02Z</updated>
   <summary>Some text.</summary>
 </entry>

</feed>
```

3 Mashup Techniques

First, we'll take a closer look at the different kinds of mashups. After we do that, I will investigate a few strategies to keep in mind when creating a mashup and what the advantages and disadvantages of these are.

First let's have a quick look at a simple mashup. We have already talked about mashups like Google Maps in Chapter 1 - these mashups project geo-tagged information onto a map. Creating such an application does not sound like a very sophisticated application and definitely does not require a lot of programming, although success does depend on having well-prepared data. To get this application to work, you need the latitude and longitude coordinates for your data. If you don't already have it, you will do some additional processing first, using a geo-tagging service, like Geonames[1], to supply the data.

Later in the book, we will be developing the *fmMusic* mashup. Let's see what we have to do to prepare our data for that project. First I need to extract the artist and track name from the radio station. Having acquired this, I'll use the new information to query Amazon's[2] database using its e-commerce web service, and get the album the track appears on. Here I will need to loop through all albums of the artist, and compare the tracks on each CD with the target track name, to get the CD which contains it. Next I want to acquire the lyrics and will be faced with having to do some deep web navigation and perhaps a web scraping mechanism. Maybe I will have to deal with some kind of authentication, too.

We can see that a mashup's complexity can vary quite a lot, from a simple projection of data on a map up to an advanced aggregation of various different sources using web data extraction tools and web services. Such applications might require a lot of computing power when many people use the mashup, or they require advanced caching technologies to solve performance issues.

Shanahan refers to three different types of mashing techniques in [Shan07].

- mashing on web server
- mashing using a rich user interface like AJAX
- mashing using JSON

I do not agree one hundred percent with this author. Shanahan points out that when using XmlHttpRequest, you cannot retrieve data from servers other than the serving one. This is commonly known as *same-origin policy* [Howe07], a security restriction that operates when browsers use cookies to authenticate unique users. The browser will only send these cookies to the same site that originally set them.

[1]http://www.geonames.org/
[2]http://www.amazon.com

Mashups most commonly refer to browser-based applications, even if more and more runtime environments, like Prism[3], Adobe's AIR[4] or Apple's Dashboard[5] are being used for mashups. The runtime environments are not limited to the same-origin policy. Therefore I will distinguish between server and client side processing techniques for building mashup applications.

A difference between these mashups does not imply that you only will use server OR client side processing. More often you will use client side mashing when you want to provide a better user experience by limiting start-up latency for the initial application, or to limit traffic through the mashing server. On the other side, you might use the processing power of the server to achieve more powerful or maybe simpler processing, using higher level programming languages. I will take a closer look at the pros and cons of each of these two possibilities.

3.1 Server-side Mashups

This is a common and well-established approach to achieve a mashup, where a server does all of the mashing. This has been around a long time, since long before the term mashup was used in this context. Common price comparison sites use this approach to aggregate product data for comparison. How does this work? The following illustrates two parts of the fmMusic mashup that use server side procedures to request data from YouTube and Amazon.

1. The client, sends a request (usually HTTP GET) to the server
2. The server contacts YouTube, using its REST interface to search for a video.
3. YouTube delivers an XML file in response.
4. Another request is sent to Amazon using SOAP.
5. A Soap response is received by the server.
6. The server uses both responses and mashes them into a web site. At this stage, all kinds of transformations, sorting, mapping or additional requests can be processed.
7. The client receives the mashed up response.

We can see that all processing is done on the server before the client ever starts to see something. The server is the single point where all information is gathered and the data is prepared for the client.

3.1.1 Benefits

- **Powerful Processing** If you have access to a server, you do not have any limitations on your choice of tools. You can use any higher level programming language (JAVA, C++, Perl, Python....), libraries (scrubyt, AXIS....) or other technologies for processing. By comparison, the client side application is usually limited in its functionality.

[3]http://labs.mozilla.com/2007/10/prism/
[4]http://labs.adobe.com/technologies/air/
[5]http://www.apple.com/downloads/dashboard/

- **Different Client Implementations** If you use client side programming, you often are confronted with situations when one implementation works well on this client (lets say Firefox) and not on the other (lets say Internet Explorer). Using libraries like *Prototype*[6] that provide you with wrapper functions help you to avoid messing with different client implementations.
- **Multi-channel Coding** Having all the collected data in a single location allows you to prepare the data in different output formats. Since it's easy to identify a client through its HTTP-Header information, you can send all data to web browsers but strip out the video part when sending to mobile devices.
- **Caching Possibilities** If you acquire data from several external sites, it is better to process it once and keep it cached, which helps you avoid being banned from the provider for high traffic.

3.1.2 Drawbacks

- **High Through-traffic** on the mashing server. Often when you request data from a source, you only need to do a minimal transformation of the data, so you can more or less just route the data through the server, which results in high traffic cost for you. (It's always better to leave the traffic costs to the content provider, if possible). Transformation often can be done on the client as well. Amazon, e.g., allows programs to include a reference to an XSLT file in the request and performs the transformation on its servers, so you just get the data you want and limit the traffic.
- **Performance Costs**. If lots of clients send requests to the server at the same time, it can result in high load and reduced performance. Swapping one transformation to the client would probably not be noticed by the user, but could lead a significant reduction in processing on the server.
- **Application Performance**. Mashing all the data together on the server for a single response to the client inevitably leads to latency until all the data is collected. If one of your sources has a delay in its response, all other data is kept waiting too, until it can be sent back to the requestor. This can be a serious problem, since most users are not willing to wait a long time and may abort the request.

3.2 Client-side Mashups

With the evolution of Web2.0, or more accurately, with the emergence of AJAX, asynchronous client-side processing has found a place in web development. Interestingly the asynchronous XML request functionality was implemented years before anyone conceived of mashups, but it got very little notice until the launch of Google Maps and Gmail. These applications brought web applications to another level by using the web browser's ability to make requests and exchange parts of the content without refreshing the whole page. So called Rich Internet Applications (RIAs) evolved and allowed a far richer user experience and started to compete with desktop applications.

[6]http://www.prototypejs.org/

3.2.1 How It Works

A browser mashup

1. The client initiates a request to the web server.
2. The web server sends an HTML response to the client. At this stage no mashing is done on the server, and the request is computed with casual web processing techniques like PHP or ASP.
3. The client now renders the received HTML and starts processing the JavaScript instructions. A function invokes an asynchronous call to server to request additional content, usually done through the XMLHttpRequest-object, but could also be a SOAP or REST request.
4. The server receives the second request and initiates a request to YouTube.
5. Youtube responds with an XML file.
6. The server requests data from Amazon.
7. Amazon responds with data.
8. Now the server mashes the content.
9. The final content is sent as response back to the client.
10. The client's request handling function can now process the response. Mainly it will place, replace or add the received content in the page being displayed. There could be some additional processing (e.g. an XSL-transformation) done before placing the content. Additionally some new processing instruction from an included JavaScript fragment could be evaluated and initiated.

A client mashup - disregarding same-origin policy

In the introduction, I talked about the security restrictions in web browsers that limit XMLHttpRequests to the serving domain. However if you implement a client mashup, such as a Firefox extension, the application sits in the Firefox browser, and therefore is not bound to any web server and will not have any security restrictions. Keeping this in mind I will adapt the process flow a bit, to limit through traffic and optimize performance.

1. The client requests the main page from the web server.
2. The response is sent to the client
3. The client renders the response and processes the Javascript. The response instructs the client to invoke two different calls, one to YouTube and one to Amazon. Both calls are done asynchronously - otherwise the client will block all user interaction on the user's machine until the calls are finished.
4. Amazon responds first.
5. The client does a transformation on the received XML.

6. It then mashes the content into the main page.
7. Now the response from YouTube is received.
8. A transformation for correct rendering is done.
9. The content is mashed into the display of the site.

3.2.2 Benefits

- **Better user experience** The user can view and interact with the main page and has no latency before the first item on the page is rendered. Of course this restricted on how the application is dependent on the mashed content.
- **Processing is mainly done on the client** Even as more and more clients access your application, if the processing is done on those clients, your server only has to deal with simple requests, and you will extend the lifetime of your current hardware.
- **Less through-traffic costs**. If you do not have to route everything through your server, you will have lower costs for the traffic you do see.

3.2.3 Drawbacks

- **Limited processing** Client processing is usually limited to scripting languages like JavaScript that are inferior to higher programming languages.
- **Multiple coding** If the client does the processing, you may need to have different sets of code to interface with different browser. This increases your coding costs.
- No cross-domain requests because of security restrictions using browsers.
- Maintenance of the application. If a content provider changes its API you may need to roll out a new version of your client, while if you are using a server mashup, you just change the implementation once on the server for all users.

3.3 Mashup Targeting

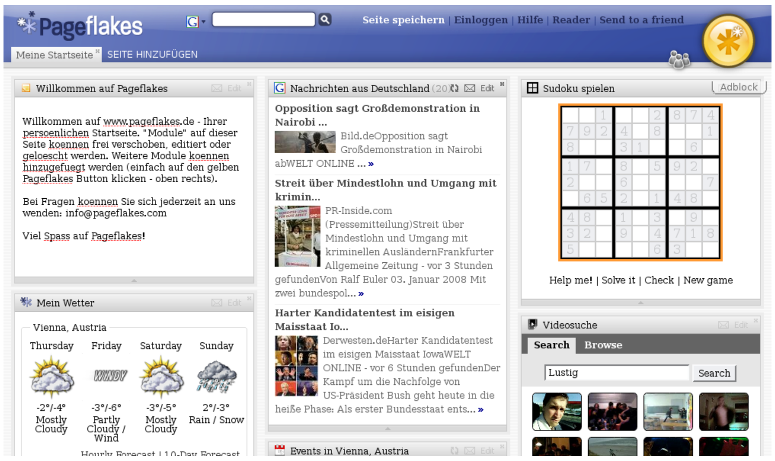
Mashups cannot only categorized in the technical view, where the processing is happening, but also orthogonally what functionality mashups target at. Dornan [Dorn05] defines three types of mashups: *presentation*, *data* and *logic*.

3.3.1 Presentation Mashups

These are the simplest types of mashups. The aim is to bring information from different sources into a common user interface for better user experience and easier handling. This type of mashup has been around for a long time. Web portals can be thought of as presentation mashups, although they act in a very static way. The JAVA Portlet Specification JSR 168[7] is an example of a common way to integrate different information

[7]http://developers.sun.com/portalserver/reference/techart/jsr168

into one portal. Websites like Pageflakes[8] (Figure 3.1 or iGoogle[9] are much more flexible and easier to handle, and users can choose from different widgets and define their own dashboards. In the enterprise so called *monitoring dashboards* are commonly used to present incoming information in a way that is easy to read. The generation of such widgets is much easier than creating a JSR-Portlet, which has enabled many people to quickly create such widgets, so that now you can choose from thousands of available widgets.


Pageflakes
Seite speichern | Einloggen | Hilfe | Reader | Send to a friend
Meine Startseite
SEITE HINZUFÜGEN
Willkommen auf Pageflakes
Edit
Nachrichten aus Deutschland (20)
Sudoku spielen
Adblock
Help me! | Solve it | Check | New game
Mein Wetter
Videosuche
Search
Browse
Lustig
Search
Events in Vienna, Austria

Willkommen auf www.pageflakes.de - Ihrer persoenlichen Startseite. "Module" auf dieser Seite koennen frei verschoben, editiert oder geloescht werden. Weitere Module koennen hinzugefuegt werden (einfach auf den gelben Pageflakes Button klicken - oben rechts).

Bei Fragen koennen Sie sich jederzeit an uns wenden: info@pageflakes.com

Viel Spass auf Pageflakes!

Opposition sagt Großdemonstration in Nairobi ...
Bild.deOpposition sagt Großdemonstration in Nairobi abWELT ONLINE ... »

Streit über Mindestlohn und Umgang mit krimin...
PR-Inside.com (Pressemitteilung)Streit über Mindestlohn und Umgang mit kriminellen AusländernFrankfurter Allgemeine Zeitung - vor 3 Stunden gefundenVon Ralf Euler 03. Januar 2008 Mit zwei bundespol... »

Harter Kandidatentest im eisigen Maisstaat Io...
Derwesten.deHarter Kandidatentest im eisigen Maisstaat IowaWELT ONLINE - vor 6 Stunden gefundenDer Kampf um die Nachfolge von US-Präsident Bush geht heute in die heiße Phase: Als erster Bundesstaat ents... »

Vienna, Austria

Thursday	Friday	Saturday	Sunday
-2°/-4°	-3°/-6°	-3°/-5°	2°/-3°
Mostly Cloudy	Partly Cloudy / Wind	Mostly Cloudy	Rain / Snow

Hourly Forecast | 10-Day Forecast

Figure 3.1: Sample homepage provided by Pageflakes

A lot of enterprise platforms offer presentation functionality, but they are not compatible or are hard to integrate with other services. Mashups will provide a solution as single dashboard frontend in near future. IBM [Webe08] integrates widgets compatible with Google Gadgets in its upcoming Lotus Notes version.

3.3.2 Data Mashups

The next step is to combine data from multiple sources into a data mashup. The aim is to provide easier access. Instead of requesting information separately from different databases and then combining it, the user can query different databases at once. This saves time and allows easier comparison and/or interaction, and most often leads to better decision making.

A common example of a data mashup would be a site that mixes geographical data with other statistics, like common mapping mashups do. Data mashups are harder to create than just mapping content into a common presentation layer. Most often you need to do some additional programming, like adding geographical coordinates to data when you try to display it on a map.

3.3.3 Logic Mashups

Logic mashups are the most complex applications, and always involve programming. They connect to different applications, automate tasks involving them and are aware of the workflow of these applications. Sometimes these applications are actually tasked

[8]http://www.pageflakes.com
[9]http://www.google.com/ig?hl=de

with imitating human behavour instead of typical automated processing, because some sites do not permit automated processing and consequently will block regular mashup requests. Common examples are travel portals that check different flights, or product price comparison sites.

In the enterprise, logic mashups compete with traditional workflow applications. Whereas the traditional models tend to focus on a single task to be automated, mashups enable rapid customization and adaptation, not just for the programmer, but for the business user as well.

Mashups are an easy way to orchestrate different services into a common interface and therefore could be seen as a new type of a presentation layer for SOA services. With their ability not only to combine internal services, but also to integrate data from outside, they offer the enterprise a new type of information integration.

4 Web Data Extraction

The World Wide Web can be thought of as the biggest information source available. This makes it increasingly attractive to extract data from it for further processing by end users and applications. Data from Web resources can be used for various tasks including information retrieval, monitoring or comparison. Using this data can deliver *Instant Awareness* of the competitors moves and therefore allows a quicker response in business decisions, which can be crucial in the fast moving world we live in.

The Web however consists mostly of unstructured data mainly in form of HTML documents built to be viewed by humans. These documents are written for presentation and not for automated extraction and often are ill-formed, meaning they do not conform to an existing standard. However browsers usually fix the errors when rendering the page.

Various techniques, languages and tools have evolved, collectively known as Web Data Extraction, to bypass these limitations and allow automated or semi-automated extraction of web data.

4.1 Definition

When defining the term Web data extraction Baumgartner [Baum06] begins by distinguishing *Information Retrieval (IR)* from *Information Extraction (IE)*.

IR puts the focus on finding the most relevant documents in a collection. This could even be the Web. The entities are the documents itself, which are analysed, evaluated and categorized. However no content is extracted out of these. You can think of a search engine like Google that lists links to documents that might relate to the search term as an example of IR.

4.2 Information Extraction

IE on the other hand, involves locating and extracting specific content from within a document. IE is a mature field that has been around as long as databases of documents have existed. IE extracts relevant content from collections and is situated at a lower level than IR. The extracted information consists of facts like prices, offers, or abstracts of a paper. IE enables companies to obtain structured information from unstructured documents for further internal use in business applications.

IE can be generally classified into two approaches [Eikv99]:

- **knowledge engineering** Grammars expressing rules for the systems are constructed by hand using the knowledge of the application of the domain. The skill of the knowledge engineer defines the outcome.

- **automatic training approach** Someone with sufficient knowledge annotates a set of documents. Then a training algorithm is used to train the system to run on novel texts. This approach is faster than the knowledge engineering but requires a sufficiently large set of training data.

IE addresses two main disciplines: *Analysis of text content* and *structuring of semi-structured text.*

Analysis of text content builds upon the field of Natural Language Processing (NLP). Short parts of texts are analysed to extract key pieces. An abstract of a newspaper could be analysed for the main subject and the location the article is written about.

Structuring of semi-structured text is used with textual documents that exhibit limited structure such as tables or lists, but do not follow any grammatical rules. Such documents are often based on tokens or delimiters like HTML-tags, for instance. While NLP techniques will not work on such documents, knowledge of the domain enables easy identification of semantic meaning. I.E from a newspaper article most often consists of a headline, an abstract and the main content.

4.3 Web Data Extraction

The Web provides a vast of information that is generally semi-structured. The information is partly dynamic, contains hyperlinks and/or can be split into different documents. Hence extraction from the Web can be seen as a topic in its own right.

Generally a Web page is unstructured if linguistic knowledge is required to extract the information. Manually written Web pages are usually structured or semi-structured, whereas dynamically-generated Web pages (i.e. the content comes from a database) are more structured.

As with Information Extraction, NLP techniques are not suited for acquiring semi-structured content, as they tend to be slow. Web content is usually structured in a repetitious or itemised way, similar to how search engines present their results.

The organisation of the data on the Web is important when you want to extract data from it. The use of hyperlinks, which allows a special structuring and linking of the data, also makes it more complicated to access all of the data. Considering deep web navigation causes to a more complex rule definition. [Chid97] classifies three types of semi-structured Web pages:

- **one-level one page**: One page contains all the information to be extracted.
- **one-level multi-page**: Several links have to be followed to reach all the information.
- **two-level pages**: Several items exists on the first level, and for each, a link must be followed to navigate to a page to access all the information

4.4 Extraction Tools

Several approaches exist to address the problem of Web data extraction and most borrow techniques from areas such as NLP, languages and grammars, machine learning, information retrieval, databases and ontologies. Therefore they present very distinct features and capabilities.

4.4.1 Taxonomy for Characterizing Web Data Extraction Tools

[Laen02] presents a taxonomy for characterizing Web data extraction tools. It is based on the primary technologies each tool uses for wrapper generation. A reference to the tools mentioned can be found in this paper.

Languages for Wrapper Development

One of the first initiatives to address the problem for extracting Web data. A new language is developed for this specific purpose in contrast to general purpose languages like JAVA or Perl. Tools in this group would include Minerva and TSIMMIS.

NLP-based Tools

Natural language processing (NLP) tools examine a document to learn extraction rules for relevant data in natural language documents. These tools apply techniques like filtering and part-of-speech tagging to build relationships between phrases and sentences and derive extraction rules. These rules are based on semantic and syntactic constraints that help to identify the relevant information. Tools in this group include RAPIER, SRV and WHISK.

Wrapper Induction Tools

These tools generate delimiter-based extraction rules derived from a set of training examples. In comparison to NLP-based tools, they do not rely on linguistic constraints, but rather on formatting features that delimit the structure of the data. Representative tools in this group include WIEN, SoftMealy and STALKER.

HTML-aware Tools

This category groups tools together that rely on the structural features of HTML documents. Before extracting the content, these tools parse the document into a tree-based structure that reflects the HTML tag hierarchy. The extraction rules are then created either semi-automatically or automatically and applied to the tree. These tools offer the smoothest learning curve and fastest results but tend to have limited flexibility. We will present two tools, Dapper and OpenKapow in section 39 that fall into this category.

Modeling-based Tools

Depending on a given target structure for objects of interest (e.g. tuples, lists), these tools try to locate portions of data that conform to this structure. Tools such as NoDoSE and DEByE adopt this approach.

Ontology-based Tools

Lastly, we have the ontology-based tools. These tools do not rely on the structure of the presentation to generate extraction rules, but rather rely directly on the data when given a specific domain application. An ontology can be used to locate constants present in the data to construct objects with it. [Laen02] mentions such a tool from the Brigham Young University Data Extraction Group.

4.4.2 Deep Web Navigation

Data extraction has been a study field for some years now. The methodologies each have their advantages and disadvantages; however, in real life situations the extraction of the content is just one part of the game. Password-protected sites, cookies, non-HTML data formats, dynamically added content on AJAX enabled web-sites and session ids are all typical obstacles that makes it difficult to access the content [Baum05].

A few tools on the market, mainly HTML-aware tools like Lixto or Kapow, come with the capability of action-based deep web recording. These tools allow the program to record the user interaction and navigation in a macro-like way without the need for any low-level programming.

Part III

Tools and platforms

5 Mashup Enablers

When creating a mashup, the content is often merged from different sources that can be accessed through a request to a web service provider, a database or Web pages.

However, sometimes you will need to extract data from a Web resource that does not offer an API, or does not want the content to be extracted. The previous chapter introduced Web data extraction in general. Most of the methodologies and tools that build up on these need some programming skill and it therefore will take a developer some time to figure out how to handle these tools the right way.

In the last few years, when using information from the Web became crucial for competition, companies started to specialize in these technologies and offer different solutions for accessing this content. There are tools that are offered as software-as-a-service, software that comes with client and server to deploy inside the enterprise, or standalone server based solutions. These tools may vary a lot. Some software allows a user to visually select elements on web pages [Baum01] and/or define rules; others use regular expression with programming languages like PERL, or the extraction is based on proprietary extraction languages [HsuC02], [WuIC05].

In this section, I introduce two tools, namely Dapper[1] and OpenKapow[2], I refer to as *mashup enablers*. A mashup enabler is a tool that allows users to easily create and adapt robots for content extraction with only novice skill.

Earlier, we introduced the three layers of a mashup fabric composition (page 7). Mashup enablers are set on the ingestion layer, as they provide access to the unstructured content of web pages. The following tools introduce different concepts of how to define extraction robots. I chose these tools because they are available to everyone without a charge. primarily

5.1 Dapper

Dapper [Dapp07] is an online Web extraction service that has a browser based interface. Dapper claims that you can create an API for any site. Using Dapper you can reuse any content that is available on the web, with the limitation that Dapper does not support any log-in mechanism right now. Dapper's user interface works on a fast forward concept and requires no programming skills. The composition of a so-called *Dapp* is done through a wizard-like interface, guiding you through five steps. These are mostly self-explanatory, however they still have some tricky parts.

Dapper is by far the most user-friendly approach on Web data extraction I have seen. You do not have to download any software or have to go through a registration process before starting. It's just as easy as clicking on *create a dapp* and off you go.

Dapper does have some disadvantages however. One of the main disadvantages is that are not be able to extract from the 'Deep Web“, thus you cannot search Ebay[3] and

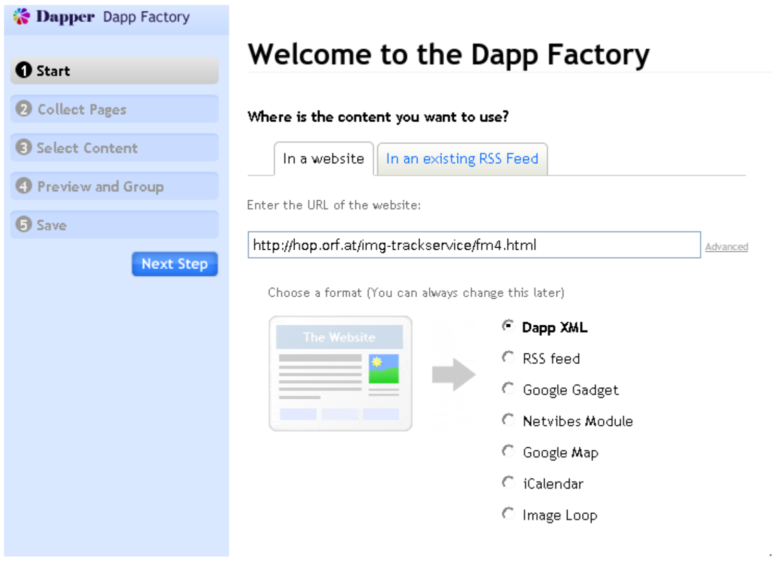
[1]http://www.dapper.net
[2]http://www.OpenKapow.com
[3]http://www.ebay.com

then extract the information from the linked, detailed description pages of the search results.

However, let's see how you can create a Dapp.

5.1.1 Creating a Dapp

When you want to create a Dapp you simply start on Dapper's homepage[4] where you find the link "Create a Dapp". First you select an output format, in this case an XML Feed, then you paste the URL of that page from which you want the content to be extracted (Figure 5.1).



Welcome to the Dapp Factory

Where is the content you want to use?

[✓ In a website] In an existing RSS Feed

Enter the URL of the website:

http://hop.orf.at/img-trackservice/fm4.html

Choose a format (You can always change this later)

[✓ Dapp XML]
RSS feed
Google Gadget
Netvibes Module
Google Map
iCalendar
Image Loop

Figure 5.1: First step creating a dapp

The second step (Figure 5.2) is for collecting pages that are used for analyzing. Dapper works best if there is a paging mechanism and a few pages are collected. In this way it gathers information about the static and the dynamic, repeating content.

After analyzing the pages you can start to select the elements you want to extract and assign a title to it. Dapper preselects all the elements it thinks you want to extract (Figure 5.3); this works on well-structured pages, however I experienced problems and failed on badly programmed pages.

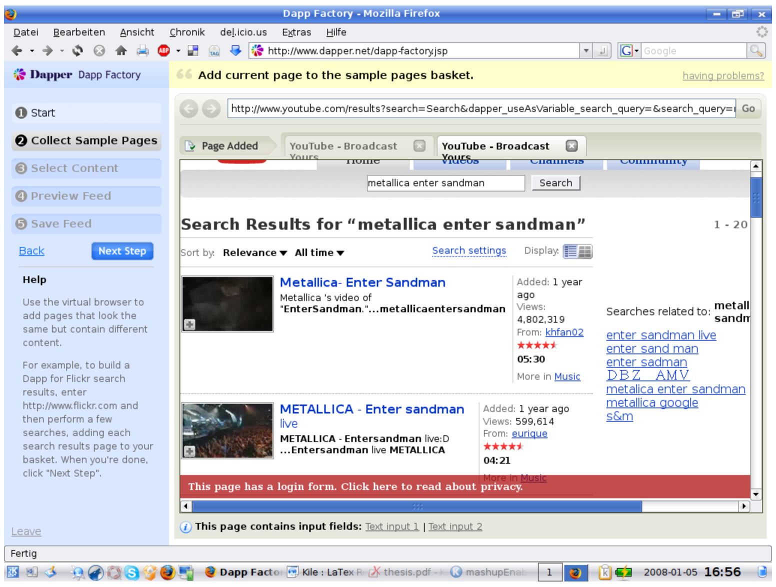
In the fourth step (Figure 5.4), you get a preview of the extracted content. The last step is to assign the elements into individual groups.

Finally you can create an account and save your Dapp and it will be ready for use. Dapper already has a few output formats and widgets that you can choose and preview. A few clicks and you get a Flash widget and its code that you can copy/paste to your blog or wherever you want.

5.1.2 Limitations

Dapper works fine on most machine generated web pages where dynamic content is loaded as tables. However I experienced problems when I tried to extract the content

[4]http://www.dapper.net


Dapp Factory - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik del.icio.us Extras Hilfe
http://www.dapper.net/dapp-factory.jsp
Dapper Dapp Factory
Start
Collect Sample Pages
Select Content
Preview Feed
Save Feed
Back
Next Step
Leave
Page Added
YouTube - Broadcast Yours
YouTube - Broadcast Yours
Search
Fertig

Add current page to the sample pages basket.

having problems?

http://www.youtube.com/results?search=Search&dapper_useAsVariable_search_query=&search_query=

Help

Use the virtual browser to add pages that look the same but contain different content.

For example, to build a Dapp for Flickr search results, enter http://www.flickr.com and then perform a few searches, adding each search results page to your basket. When you're done, click "Next Step".

Search Results for "metallica enter sandman" 1 - 20

Metallica- Enter Sandman
Metallica 's video of "EnterSandman."...metallicaentersandman
Added: 1 year ago Views: 4,802,319 From: khfan02 05:30 More in Music

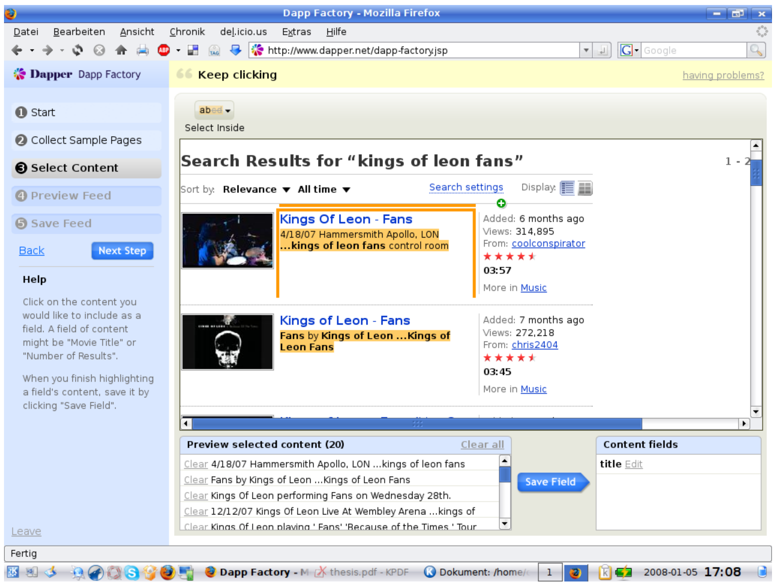
METALLICA - Enter sandman live
METALLICA - Entersandman live:D ...Entersandman live METALLICA
Added: 1 year ago Views: 599,614 From: eurique 04:21 More in Music

Searches related to: metall sandm
enter sandman live
enter sand man
enter sadman
DBZ AMV
metalica enter sandman
metallica google
s&m

This page has a login form. Click here to read about privacy.

This page contains input fields: Text input 1 | Text input 2

Figure 5.2: Collecting sample pages



Keep clicking

having problems?

Help

Click on the content you would like to include as a field. A field of content might be "Movie Title" or "Number of Results".

When you finish highlighting a field's content, save it by clicking "Save Field".

Search Results for "kings of leon fans"

Kings Of Leon - Fans
4/18/07 Hammersmith Apollo, LON ...kings of leon fans control room
Added: 6 months ago Views: 314,895 From: coolconspirator 03:57 More in Music

Kings of Leon - Fans
Fans by Kings of Leon ...Kings of Leon Fans
Added: 7 months ago Views: 272,218 From: chris2404 03:45 More in Music

Preview selected content (20) Clear all

Clear 4/18/07 Hammersmith Apollo, LON ...kings of leon fans
Clear Fans by Kings of Leon ...Kings of Leon Fans
Clear Kings Of Leon performing Fans on Wednesday 28th.
Clear 12/12/07 Kings Of Leon Live At Wembley Arena ...kings of
Clear Kings Of Leon playing ' Fans' 'Because of the Times ' Tour

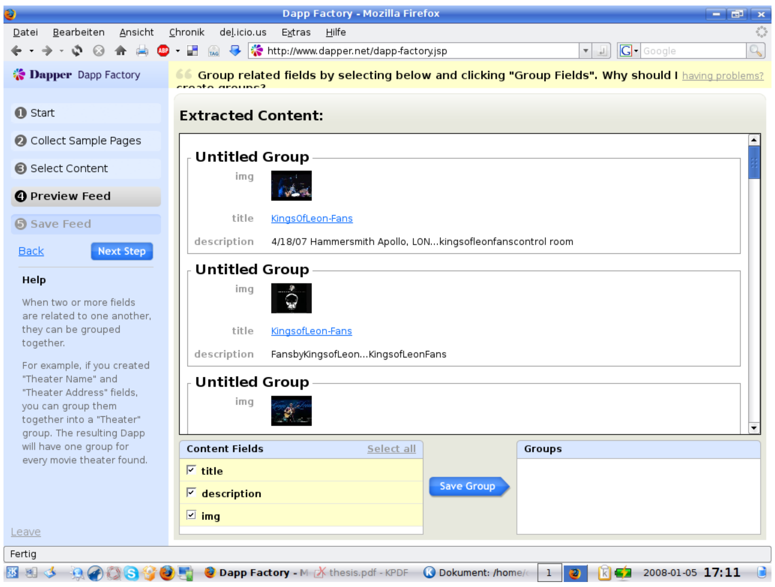
Content fields
title Edit

Figure 5.3: Selecting the content to extract

for our study case. Dapper offers a few fine tuning mechanisms, e.g. you can deselect preselected content or you can remove text before or after the selected content, but if you need more advanced tuning, Dapper can't help you.

5.2 Openkapow

Openkapow [Open07] is an open service platform built on Kapow Technologies' Web extraction framework. To use OpenKapow you first need to download the client software. With the client you can define the extraction mechanism, the so-called robot. If the extraction delivers the required output, one uploads the robot to the OpenKapow



Group related fields by selecting below and clicking "Group Fields". Why should I create groups?

Extracted Content:

Untitled Group
img
title KingsOfLeon-Fans
description 4/18/07 Hammersmith Apollo, LON...kingsofleonfanscontrol room

Untitled Group
img
title KingsofLeon-Fans
description FansbyKingsofLeon...KingsofLeonFans

Untitled Group
img

Content Fields
☑ title
☑ description
☑ img

Groups

Help

When two or more fields are related to one another, they can be grouped together.

For example, if you created "Theater Name" and "Theater Address" fields, you can group them together into a "Theater" group. The resulting Dapp will have one group for every movie theater found.

Figure 5.4: Grouping of the extracted content

server, which is handled through a publish command in the application, and the service is ready to use from within the site.

The IDE (Kapow RoboSuite) allows you to extract information in a visual manner. It is, however, not that easy to handle, and the user interface is not cleaned up enough not to scare away a novice user.

Unlike Dapper, OpenKapow comes with just three output formats, RSS, REST and web clips. To create a robot you have to define one of these. A few tutorials that are provided online give you an idea what is possible.

The user interface (Figure 5.5) is divided into three areas:

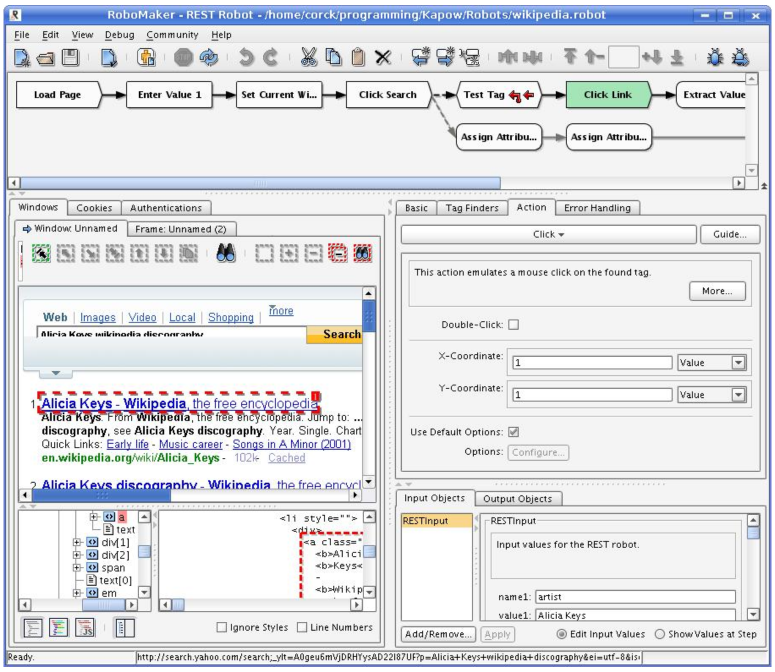
- On top is the step view. There you can see the state the robot is in. Below on the left side is the embedded browser and the source view. The source view presents the exact path in the document, which comes in handy when you need to select rows that you want to loop through.

- The browser area is where the navigation and extraction is done. In here, you simulate user behaviour like clicking on a link or filling out forms. You even can log-in to password protected pages.

- On the right is the properties view. This is for fine adjusting the extraction of each step and where you can find the input and output variables and their state.

Working with the RoboSuite is neat. If you start a new project you are offered the choice of creating RSS/Atom, REST or a webclip. Then you enter the URL where you want to start from. If you need to set any input variables or output variables for a REST service, you can define these too.

In the case study for fmMusic I will need artist data from Wikipedia. Let's have a look as to how this could be done with openkapow. The start page will be Yahoo's search page, where I first enter the value of our input and add the strings for Wikipedia and discography. Next I click the search button to get the result list. Next I add a test



This action emulates a mouse click on the found tag.

Double-Click: ☐

X-Coordinate: 1 Value

Y-Coordinate: 1 Value

Use Default Options: ☑

Options: Configure...

RESTInput

Input values for the REST robot.

name1: artist

value1: Alicia Keys

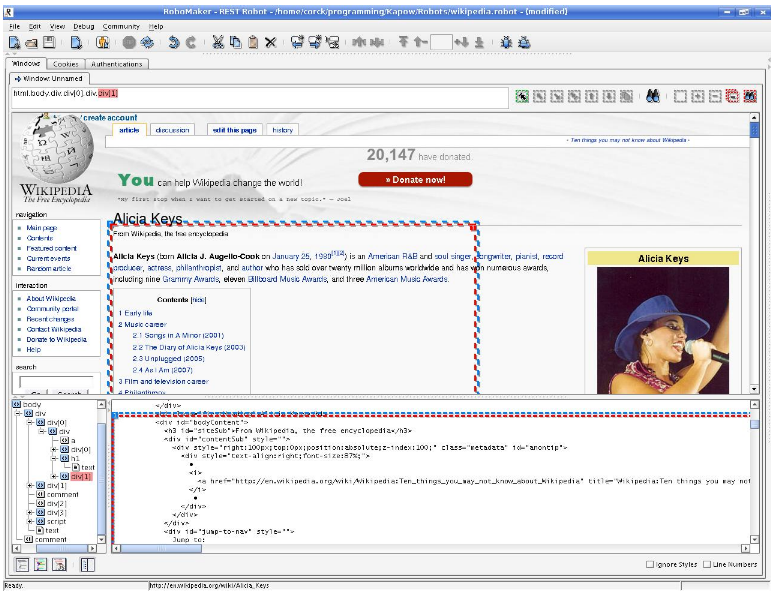
(●) Edit Input Values ( ) Show Values at Step

http://search.yahoo.com/search;_ylt=A0geu6mVjDRHYysAD22l87UF?p=Alicia+Keys+wikipedia+discography&ei=utf-8&is

Figure 5.5: OpenKapow Robosuite IDE

step, where I test if the URL of the first result contains the string en.wikipedia.org. If this is false, I throw an error and the robot navigates to the second branch, there I set the output variable to *No results found*, to have an appropriate error handling mechanism.

If no error occurs, I click the first result, which leads us to the right article and extracts the tag with the required content as HTML (Figure 5.6) and sets this to our output variable.


RoboMaker - REST Robot - /home/corck/programming/Kapow/Robots/wikipedia.robot (modified)
File
Edit
View
Debug
Community
Help
Windows
Cookies
Authentications
Window: Unnamed
html.body.div.div[0].div.div[1]
Alicia Keys
Ignore Styles
Line Numbers
Ready.

http://en.wikipedia.org/wiki/Alicia_Keys

Figure 5.6: Extraction with the RoboSuite

If the robot delivers the right results you are ready to publish it to OpenKapow's webserver and you can start using it.

5.2.1 Limitations

Using OpenKapow, you can build very powerful solutions for web content extraction. If you want to use it in the enterprise and do not want to have security risks, you will need to use a commercial variant. If you generate an RSS feed you are limited to the frequency of its refreshes; if you need a higher refresh rate than a 30 minute refresh, you will need to build an REST robot and query it as often you need . I didn't have tested intensively, but I expect OpenKapow could achieve all kinds of tasks.

5.3 Conclusion

Mashup enablers are highly sophisticated and there are tools available that allow for the creation of extractions in a very short time. Now and in the future, mashup enablers will play a crucial role in using the web as database. Mashups will depend on solutions that are robust and easily adaptable. Web site layouts change very often, nearly every 2-3 years or more. Adapting the extracting solutions to allow a quick response plays a very important role in building robust mashup solutions. When querying huge amounts of data, like various offers from different airlines for a travel comparison site, you need more sophisticated features. Lixto[5] is a commercial vendor of a more advanced mashup enabler. Lixto provides a powerful tool (Visual Developer) for data extraction and a server (Transformation Server) that runs these extraction services.

[5]http://www.lixto.com

6 Mashup Platforms

6.1 Introduction

The next trend in application development will probably be the creation of mashups for the end-user. Last years McKinsey's global survey [Bugh07] suggested that a respectable 21% of companies were investing in mashup technologies, therefore it is not a suprise to see more and more companies developing mashup platforms for enterprise.

In this chapter, I will present a quick look at different mashup platforms available right now. During the writing of this thesis the number of mashup platforms and tools increasedrapidly, so this should not be seen as a complete list. Some tools provide easy access to everyone, others have trial versions or do not offer a free evaluation at all. Since mashup platforms are quite a new field in software development, changes occur often and development is rapid. Therefore keep in mind when reading this that these platforms may have already changed and their feature list expanded.

6.2 What you can expect

So what can you expect using such platforms? We have already pointed out that mashups will enable the development of situational software. The business user will be able to customize his own application in an integrated environment. Mashups will make it possible to have an integrated view, known as a mangement dashboard, of the whole business without having to switch between different applications, wikis or end-user development tools like MS Excel.

We already do this using Web portals, where someone can define his own start pages. Mashups, however, go further. Most mashups already build on widgets that offer flexibility and act as container in which you can load all kind of content. These widgets can also allow the user to connect inputs from other widgets to create new business functionality. Some mashup platforms like QEDWiki from IBM already offer such functionality: e.g. you have a widget that delivers your ten best salesman, and another widget connects these names and displays their telephone number. A third widget then allows you to send a SMS-message to each or all of them. This is where the *situational* aspect comes in. If you want to send an email instead of text-messaging, you just exchange the widget. It's all about what the business user needs here and now, and reduces his dependency on the IT-department to make this possible. Yahoo! Pipes is an amazing example of the usability and simplicity a mashup can offer. If you try to connect services to each other you immediately get visual feedback if the module takes the corresponding input.

The IT department is of course the group in any business that has to provide these services that all these widgets rely on. A lot of commercial software already allows users to generate web services and nearly all commercial database systems allow users to generate web services out of stored procedures, which then can be used as providers to widgets. So it is all just a question of how to generate the web services and offer them as widgets. Until now, this is one of the points seperatong different platforms. Do they offer widgets or do they just provide access to the different systems on a single end or maybe both?

At the moment there is no official standard for creating mashup platforms, or better still, the ingestion layer nor the presentation layer of mashups. We probably won't get one any time soon as there is still too much development going on.

6.3 What you will get

If you start thinking of deploying a mashup in your company you will have to spend some time looking at all the different mashup providers that are available. There will probably not be a single platform that fits all your needs, however one will fit your needs better than the others. It could even possible to stick to different solutions together, especially if you need a good integrator tool and a nice looking, user-friendly frontend.

6.4 Example Mashup platforms

6.4.1 Yahoo! Pipes

Yahoo! Pipes (Pipes) is Yahoo!'s first mashup platform. Yahoo! defines Pipes as: *Pipes is a powerful composition tool to aggregate, manipulate, and mashup content from around the web* [Yaho07]. The name "pipes" comes from the Unix approach of sticking simple commands together to build a powerful ensemble, and this is exactly what you can do with Pipes. Pipes is a browser based Web application that comes with an AJAX enabled GUI interface (Figure 6.1) where individual modules are dropped on a surface and connected to each other.

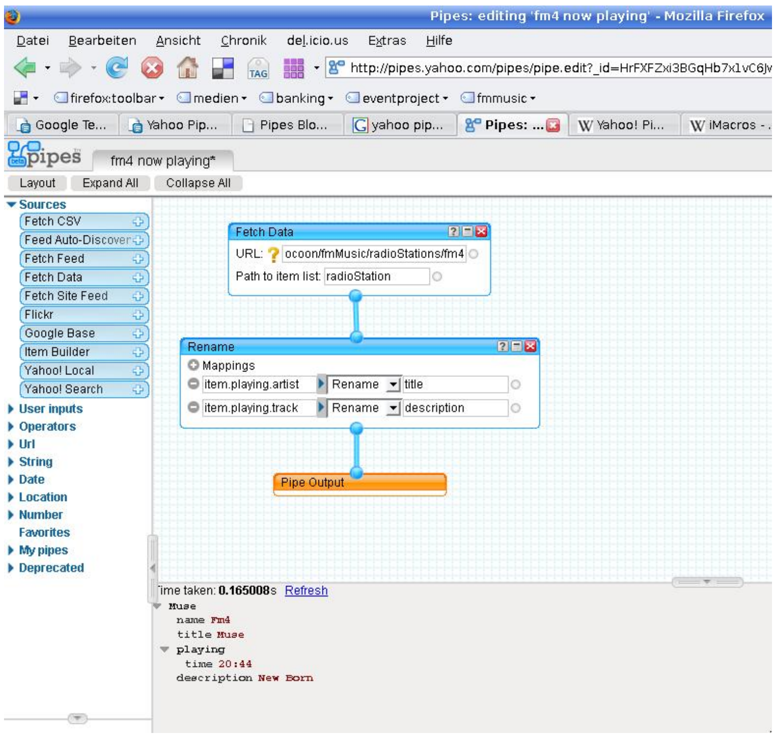
Working with the editor is convenient. A module is simply dragged and dropped onto the surface and is instantly available for use. You can drop as many modules as you want on it. Two modules are connected by drawing a line from one connection point to another one. The visual response of flashing connection points gives immediate feedback as to whether the two modules are compatible.

You can solicit user input and build URL lines to invoke different web services. The drag and drop editor lets you view and construct your pipeline, inspecting the data at each step in the process and of course, you can view and copy any existing pipe.

Concept

Using the Pipes editor you start by using one or more of the sources modules. Pipes come with a different set of source modules, where the main focus is on RSS for fetching news feeds. Other modules can process CSV and XML as well. Special modules exist for processing data from photo sharing site like Flickr or an online database like Google Base; they come with an optional filter attribute *location*. Another two modules exist for searching Yahoo! and Yahoo! Local search.

Most of the time you do not just want to aggregate different content; you will want to search or filter and want this to happen dynamically. Therefore Pipes comes with a user input module that handles multiple values. These inputs can then be connected to the Yahoo! search module to search for it, or the URL builder module could be used, to build an URL that is fed as input to the fetch data module.


Pipes: editing 'fm4 now playing' - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik del.icio.us Extras Hilfe
http://pipes.yahoo.com/pipes/pipe.edit?_id=HrFXFZxi3BGqHb7x1vC6jv
firefoxtoolbar medien banking eventproject fmmusic
Google Te... Yahoo Pip... Pipes Blo... yahoo pip... Pipes: ... Yahoo! Pi... iMacros
pipes
fm4 now playing*
Layout Expand All Collapse All
Sources
Fetch CSV
Feed Auto-Discover
Fetch Feed
Fetch Data
Fetch Site Feed
Flickr
Google Base
Item Builder
Yahoo! Local
Yahoo! Search
User inputs
Operators
Url
String
Date
Location
Number
Favorites
My pipes
Deprecated
Fetch Data
Rename
Mappings
Pipe Output

URL: ocoon/fmMusic/radioStations/fm4
Path to item list: radioStation

item.playing.artist Rename title
item.playing.track Rename description

```
Time taken: 0.165008s Refresh
Muse
  name Fm4
  title Muse
  playing
    time 20:44
  description New Born
```

Figure 6.1: Yahoo! Pipes interface

Having fetched the input data, different operators can be used to extract the data, filter, sort, split, truncate, tail, do calculations or use a simple loop to apply different string operations on each item. Two modules that deserve special attention are the the *location extractor* and the *web service module.*

- **Location extractor** *This module analyzes text in each feed item title and description and attempts to identify addresses, location names or popular map service URLs. If the extractor finds location entities in the feed, it will annotate each item with a y:location sub-element containing that item's latitude and longitude.*

  This module allows a simple mashup of any data with a representation on a map. The amazing thing is the analysis of the content to get all the important stuff out. Here is where the power of having a company like Yahoo! at your back is a real advantage.

- **Web service module** *This module POSTs the items in a pipe in JSON format to an external web service. This allows developers to extend the Pipes functionality to do whatever they need. The original items are replaced by the web service's JSON or RSS response.*

  Even if the modules that come with Pipes allow quite a lot of processing, there are always times when you reach the limits and at this point this module comes in handy. This module allows you to send the pipe data as a POST message to any URL to do additional processing on the server and take back the result. So there are no limits in terms of the programming language or service you want to use. You just have to be aware that the response has to be in JSON or RSS format.

Pipes comes with a really intuitive interface, and it helps the user be productive too, in the way it easily lets you create a pipe and copy it to quickly generate a new pipe to adapt. Having all your pipes always ready to use inside other pipes is efficient too. This helps you to build small services that can be easily modified and stuck together to form a big service. Once again the piping approach is applied.

Pipes offers a nice debugging facility having the debugger at the bottom of the editor. You can have a quick look into the current state of the pipeline by clicking on a module. The debugger will show you the values at this step. (Figure 6.2).

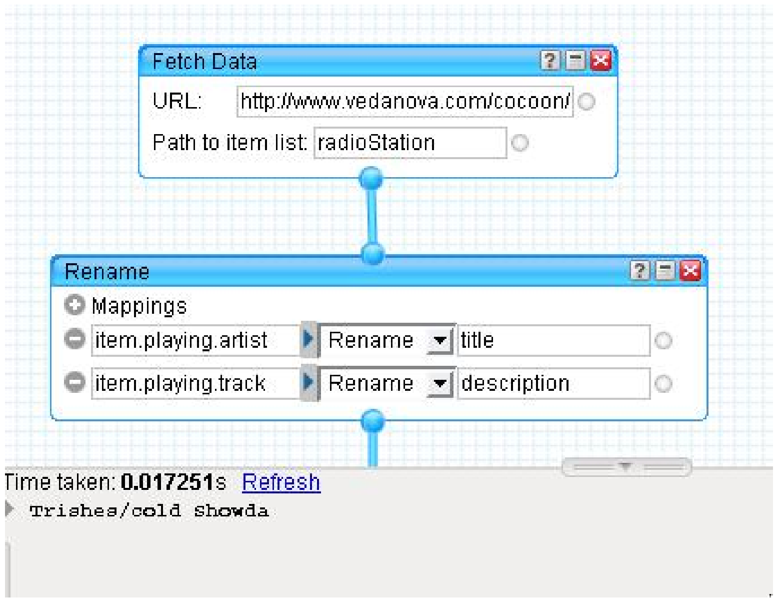


Figure 6.2: Pipes debugger

Conclusion

Pipes is an impressive mashup platform that uses astonishing AJAX technology. It primarily focuses on data mashing and therefore provides just a limited set of frontend output. Pipes is far from a enterprise mashup tool and focuses primary on consumer based mashups. The syndication features and its special modules may be a valuable tool for collecting web data for solving some parts of enterprise integration problems. The simplicity of use and ability to recombine other pipes built other users will help to evolve it quickly.

6.4.2 Microsoft Popfly

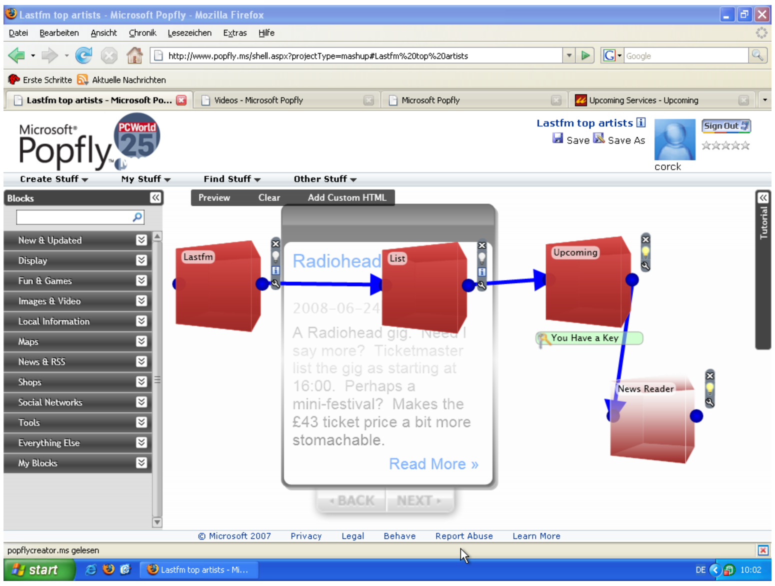
Introduction

Microsoft[1] has released its first foray into the mashup area with Popfly [Micr07]. Popfly can be seen as one of the first Silverlight[2] (formerly WPF/E) applications and demonstrates its enormous potential. Popfly is still in beta so we have to wait to see what the final version will bring.

[1]http://www.microsoft.com
[2]http://silverlight.net/

Concept

That said, Popfly has already received much attention after its first beta release. This is in part because Microsoft is a big player that gets a lot of press whenever it does anything, but also because of its impressive user interface. The principle is similar to Yahoo! Pipes; you get modules that you drop on a surface and then connect them to each other, as seen in Figure 6.3. The output modules of Popfly already offer nice representations to look at (Figure 6.4.


Lastfm top artists - Microsoft Popfly - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://www.popfly.ms/shell.aspx?projectType=mashup#Lastfm%20top%20artists
Microsoft Popfly
Lastfm top artists
Save Save As
Sign Out
corck
Create Stuff My Stuff Find Stuff Other Stuff
Blocks
Preview Clear Add Custom HTML
New & Updated
Display
Fun & Games
Images & Video
Local Information
Maps
News & RSS
Shops
Social Networks
Tools
Everything Else
My Blocks
Lastfm
List
Upcoming
You Have a Key
News Reader
Tutorial
BACK NEXT
© Microsoft 2007 Privacy Legal Behave Report Abuse Learn More
popflycreator.ms gelesen
start

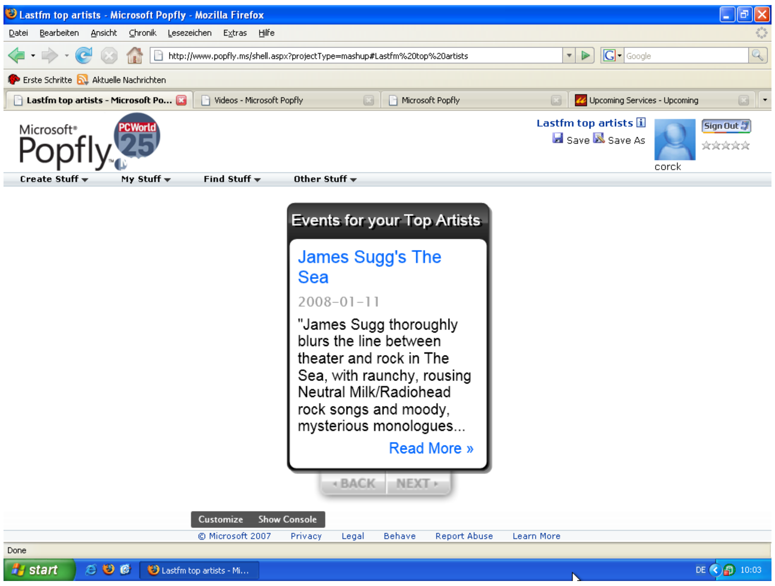
Radiohead

2008-06-24

A Radiohead gig. Need I say more? Ticketmaster list the gig as starting at 16:00. Perhaps a mini-festival? Makes the £43 ticket price a bit more stomachable.

Read More »

Figure 6.3: Popfly interface


Lastfm top artists - Microsoft Popfly - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://www.popfly.ms/shell.aspx?projectType=mashup#Lastfm%20top%20artists
Microsoft Popfly
Lastfm top artists
Save Save As
Sign Out
corck
Create Stuff My Stuff Find Stuff Other Stuff
BACK NEXT
Customize Show Console
© Microsoft 2007 Privacy Legal Behave Report Abuse Learn More
Done
start

Events for your Top Artists

James Sugg's The Sea

2008-01-11

"James Sugg thoroughly blurs the line between theater and rock in The Sea, with raunchy, rousing Neutral Milk/Radiohead rock songs and moody, mysterious monologues...

Read More »

Figure 6.4: Popfly result module

Popfly, even if it is not as long available as Yahoo! Pipes does come with a lot of predefined modules (blocks) for use. If you have a closer look you see that a lot of them were created by hobbyists outside of Microsoft. Popfly allows you to build your own blocks and has an SDK available for download.

Building blocks using a REST service is not too difficult. You can build a block with a simple combination of JavaScript and XML. A good example is D. Waters' Geolocation Block [Wate07]

Conclusion

Popfly demonstrates how easy it is to generate new blocks for integration within a mashup platform and how a visually attractive an editor can look and behave. What it makes it so powerful, Silverlight, is probably the main reason it will not make it to the enterprise very soon. Silverlight is a proprietary browser plug-in that needs at least Internet Explorer SP2.

6.4.3 IBM's Mashup Starter Kit

Introduction

IBM recently entered the mashup market with the release of their Mashup Starter Kit. Even if all their products are still only available on their emerging technologies site alphaWorks[3] IBM is pushing hard into the enterprise sector. With the Mashup Starter Kit IBM follows the three layers model of a mashup composition1.5.1 and provides a product for each layer. The package includes *IBM DAMIA* [IBMA07a] (Ingestion layer),*Mashup Hub* [IBMA07] (Augmentation layer) and *QEDWiki* [IBMA07a] (Presentation layer).

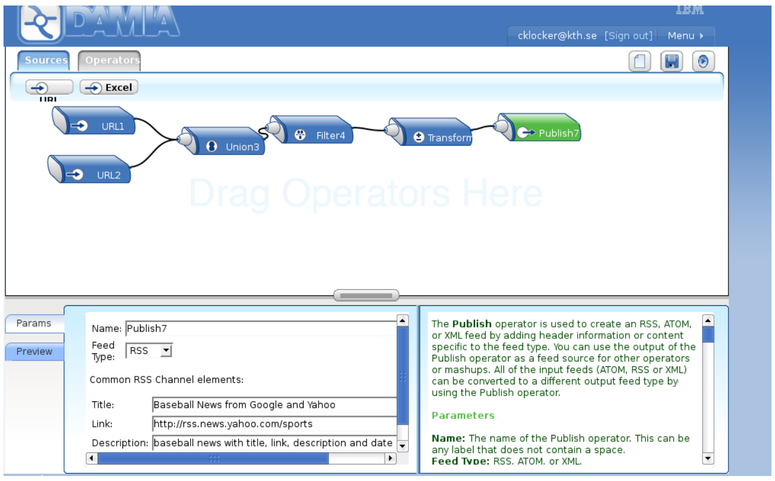
DAMIA

Damia is a tool, once again with a Web-based interface, for data aggregation. It offers similar features to Yahoo! Pipes, whereby you can apply functions like filter, concat, split etc on the data. The development is similar too, as you stick together different modules and enable the data for further consumption. In comparison to Pipes, Damia is already aimed at enterprise usage and offers import from sources like Excel; database support is on the way.

Mashup Hub

Mashup Hub is a kind of registry for all widgets than can be used in the presentation layer. All services that are created with Damia can be registered here and made accessible to the application creator within QEDWiki. Mashup Hub also provides feed generation for enterprise data sources like relational databases, collections of XML documents in DB2, Microsoft Excel files, comma-separated value files, Microsoft Access exported queries, IBM Information Server federated data, and the contents of ordinary XML documents. A user can register existing feeds in the Mashup Hub catalog too. Mashup Hub is the foundation of QEDWiki, which serves as the repository for the available services.

[3]http://www.alphaworks.ibm.com/



Name: Publish7
Feed Type: RSS

Common RSS Channel elements:

Title: Baseball News from Google and Yahoo
Link: http://rss.news.yahoo.com/sports
Description: baseball news with title, link, description and date

The **Publish** operator is used to create an RSS, ATOM, or XML feed by adding header information or content specific to the feed type. You can use the output of the Publish operator as a feed source for other operators or mashups. All of the input feeds (ATOM, RSS or XML) can be converted to a different output feed type by using the Publish operator.

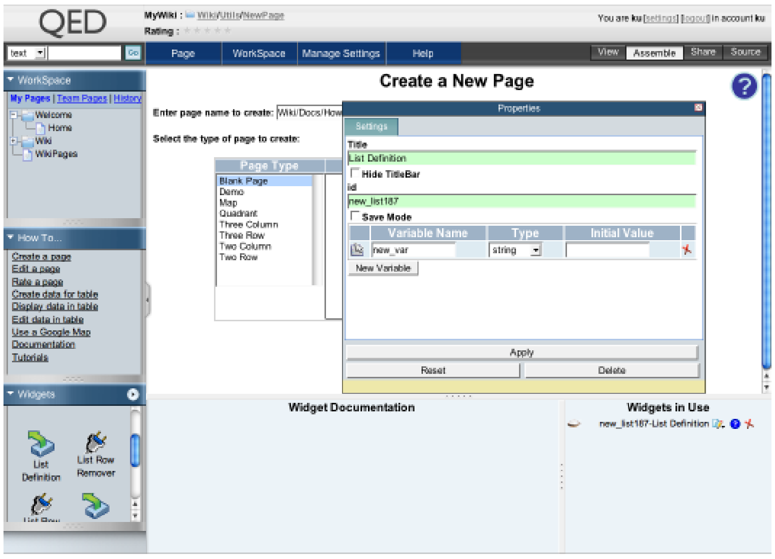
Parameters

Name: The name of the Publish operator. This can be any label that does not contain a space.
Feed Type: RSS, ATOM, or XML.

Figure 6.5: Damia editor

QEDWiki

QEDWiki is IBM's interesting approach to build a Wiki type of mashup platform; it is currently in alpha state. QEDWiki is a browser based development environment consisting of an assembly canvas and a widget collection. Different from other mashup platforms, QEDWiki provides users and developers with a single frontend for application development (Figure 6.6) and final presentation.



Enter page name to create: Wiki/Docs/How

Select the type of page to create:

Title: List Definition
Hide TitleBar
id: new_list187
Save Mode

Variable Name	Type	Initial Value
new_var	string	

new_list187-List Definition

Figure 6.6: QEDWiki user interface

Working with QEDWiki is quite tricky as it is not intuitive at all. There are a few tutorials showing the concepts behind it, and you will need some time to get used to it. The tricky part is the connection between the different widgets, one of its main features. At this stage it is hard to identify how to connect widgets to each other. Providing an interface that is more intuitive to business end users is a major issue, and and the moment I cannot see this being adopted very quickly. Pipes, Popfly, or even Damia shows how intuitive connecting modules can be; QEDWiki makes it unnecessarily difficult.

Lotus Mashups

At the time of writing, IBM announced its new product *Lotus Mashups*[4], which can be seen as follow up to Mashup Starter Kit.

IBM Lotus Mashups is expected to include:

- A graphical, browser-based tool that will help enable easy, on-the-glass assembly of new applications by Web-savvy business users.
- A Mashup catalog which will help facilitate the sharing and discovery of mashup components, with planned, built-in community features like ratings, tagging, commenting.
- A very lightweight Mashup Server, which will be hostable on a variety of platforms for added governance and IT control.
- A rich set of out-of-the-box, business-ready widgets.

Other Mashup Platforms in brief

SnapLogic[5] is a open source, community-based mashup server for data aggregation. It comes as platform independent download and builds on Python. The graphical interface built on Flash is impressive. The server is acting as backend to mashups and provides services to the presentation layer.

Proto[6] is a mashup platform directed to financial services and offers a large set of predefined modules. The focus is on connection of different datasources and the real time manipulation of data for financial simulations.

For further tracking of mashup solutions I recommend Dion Hinchcliffe's blog[7], where he keeps an updated list of mashup platforms.

[4]http://www-306.ibm.com/software/lotus/mashups/
[5]https://www.snaplogic.org/
[6]http://www.protosw.com/
[7]http://blogs.zdnet.com/Hinchcliffe/?p=111

Part IV

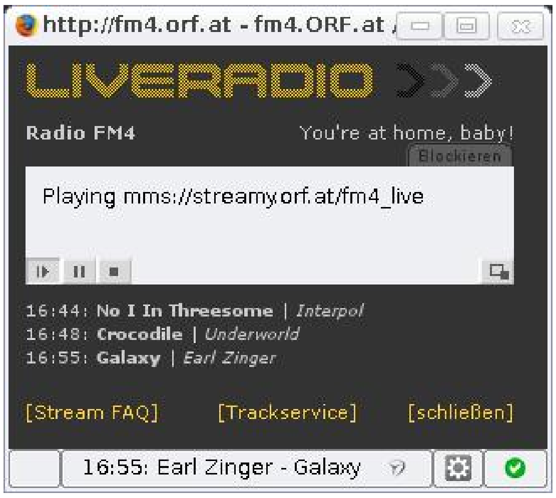
FmMusic - a music mashup

7 Specification

FmMusic is a mashup that extracts the currently playing artist and track information from a radio station's homepage. The information is then used to acquire additional information that could be relevant for the user of a mashup, such as artist information, album on which the song appears, lyrics, et cetera. In this chapter we will define all the information we want to display and what resources we will use.

7.1 Artist/Track

First of all, we need information about what artist and track is playing at the moment. In our example case, the public Austrian radio station, Fm4, is used. The radio station offers a Web stream to its listeners on its Web site. Within the page of the stream, the current and last two songs played are displayed (figure 7.1). The radio station provides neither an RSS Feed nor an API to get this information in a convenient way, so we will have to use a Web data extraction mechanism. During an evaluation of Dapper and openkapow, we tried this extraction with both products. Dapper failed to deliver the right results, as it was not possible to limit the selection to one of the three different tracks. Openkapow showed its strength and the extraction worked fine. However we will not be using openkapow for our extraction, but will rely on the Cocoon framework. Because we gather the information every minute, we don't want to be dependent on a product as prone to downtime as openkapow.



Playing mms://streamy.orf.at/fm4_live

16:44: **No I In Threesome** | *Interpol*
16:48: **Crocodile** | *Underworld*
16:55: **Galaxy** | *Earl Zinger*

16:55: Earl Zinger - Galaxy

Figure 7.1: Fm4 radio stream

7.2 Album, Album cover

One of the main goals is to be able to tell the user on which album a song appears, and display the cover of that album. To get this information the e-commerce API from Amazon (AWS) will be used. Amazon offers SOAP and REST APIs to look up a track in its database and get relevant information from its data pool. In the mashup, the REST API is used, mainly because it is supported out of the box within Cocoon. An interesting point is that Amazon states that 85% of its web services users favor REST as their interface [ORei03].

Finding out on which album the track appears is probably the hardest part of the process. How do I achieve that? Consider the Amazon e-commerce web service. It allows us to query its database for all the products it has available. I will limit my search to the product group “Music” and use the additional response group “Track”to get all the track names that appear on the selected album. For most artists, I will get more than one item back, because most of the artists have more albums brought out. So I need to loop through all the items and compare the track name with the track that is playing right now. If I find a match, I get the album name and image URL. More on that in the next chapter.

7.3 Lyrics

Another feature for the mashup is to display the lyrics for each song. There are quite a lot of lyric platforms available, so I had to decide on which one to choose. After searching for some sample songs on different platforms, I decided to use AzLyrics.com. On the one hand, it provides a large collection of lyrics and on the other hand, lyric extraction appears relatively easy. We have to do another extraction, as there is no API available. Providing lyrics is difficult, especially for the station I chose, because they play a lot of lesser known artists or even play local bands for which you will not find lyrics on any platform.

7.4 Artist Information

A lot of music fans adore their music idols and want to know everything about them. I am not such an enthusiast but I often do want to know more about an artist, so it seems likely that many other users will too. There are a few portals that offer good collections of artist information, but here too, it is hard to find a complete source. There are always plenty of new bands who are played by the radio stations, but there is nothing written about them except on their personal webpage. In the last years *Last.fm*[1] has started to become the most valuable source for artist information. Unfortunately, their content is not under a shared license, so I will not use it. Wikipedia, which contains information about nearly everything, also provides information on artists, thus we will use it at content source. Wikipedia unfortunately still has not released any API, so again a screen scraping mechanism is in order.

[1]http://www.last.fm

7.5 Music Videos

Typically radio stations play songs that are released as singles and with each single, a complementary video is released. The most popular video resource on the web right now is YouTube, which even offers a convenient API for querying their content through a REST interface and delivers complementary information. YouTube is also great for finding videos that have not been released as single. A lot of times you can eve get a few live recordings of the song.

7.6 Pictures

The last part of the specification will be pictures of the artist. For this, we will use Flickr, which has had huge success right from the beginning of the Web2.0 era, and is now one of the biggest photo sharing sites on the net. It does not only provide a lot of meta-information, it also gives access to its collection through a REST API.

8 Apache Cocoon

8.1 Introduction

In the next chapter, we will build the fmMusic mashup on top of Cocoon. Therefore, this necessitates a quick introduction to Cocoon. We have read in previous chapters that the technologies behind mashups are common standards that can be implemented in various programming languages or application frameworks. Cocoon is an XML publishing framework that comes with many so-called blocks. The main components behind Cocoon are generators that acquire or generate data, transformers that modify the data and serializers that generate the output formats. Cocoon is very modular and can easily be used as a very scalable SOA application framework. Within Cocoon it is easy to define small services, so-called pipelines that can be connected in a simple way to build complex applications. Aggregation functionality allows easy combining and manipulating of data. Another benefit is the sophisticated built-in caching functionality, where each pipeline can be assigned an expiration value that defines how often the cache must be refreshed. This is useful when using Web scraping to avoid e.g. scraping the information for an artist for each request.

We can see that Cocoon offers functionality in processing and delivering data. In building one's own mashup, there will always be a choice of technology with which to implement it. Most often, one would use a familiar technology, or build the mashup on a framework already implemented. Next, I will explain why I chose Cocoon as my framework:

- Cocoon is an XML based framework: I will use Web service providers that offer REST interfaces. All such interfaces deliver an XML response. This means that the structure for processing within Cocoon already exists.

- Caching functionality: Nearly all the information the mashup provides will not change very often. Good caching functionality will limit the computing power needed and the traffic routed through the server.

- Scalability: It is very easy to extend Cocoon's functionality. During implementation, most of the functionality will be carried out as small services, each of which are REST interfaces.

- Different output formats: With its clear separation of data generation, transformation and presentation, Cocoon offers the ability to produce data in different output formats such as HTML, XML or JSON. Even if fmMusic is just a Firefox extension, it could easily be implemented as a browser mashup.

- Basic Web scraping possibilities with XSLT using a built-in tidy[1] module (HTML-Generator).

[1] tidy.sourceforge.net/

Next, I will examine the concepts used by Cocoon and will then explain how to build the backup for fmMusic.

8.2 The Cocoon framework

Developing Web applications began with coding static HTML documents delivered to the user in a static way. This soon changed when developers realized the need to generate content dynamically. A common source for such dynamic content was a database capable of querying, with results being added to the static document content. Displaying dynamic content often required some kind of programming logic, as well. It was at this point that dynamic scripting languages such as Hypertext Processor (PHP), Active Server Pages (ASP) and JavaServer Pages (JSP) evolved. These scripting languages offered developers a great deal of functionality, but also had drawbacks. For example, these scripting languages have no separation of content and application logic. Cocoon was developed with this in mind. The Apache Cocoon Project homepage [Apac05] introduces the concept of separating content and application logic as follows.

8.3 Separation of Concerns (SoC)

> Apache Cocoon is a web development framework built around the concepts of separation of concerns (making sure people can interact and collaborate on a project, without stepping on each others' toes) and component-based web development. Cocoon implements these concepts around the notion of "component pipelines", each component on the pipeline specializing on a particular operation. This makes it possible to use a "building block" approach for web solutions, hooking together components into pipelines without any required programming. Cocoon is "web glue for your web application development needs". It is the glue that keeps concerns separate and allows parallel evolution of the two sides, improving development pace and reducing the chance of conflicts.

Page designers commonly do not focus on the application logic that lies behind a Web application. Application developers do not want to be bothered with presenting the data in a user-friendly manner. Therefore, the best way to manage a Web site is to separate the logic and presentation into different components that can be acted on individually. For the sake of coordination, there must of course be some management capabilities available.

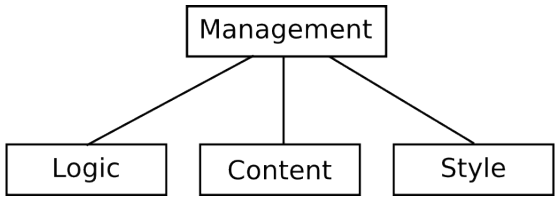


Figure 8.1: Three concerns of Web page development and management

Using such separation makes it easy to change a representation and just involves the person responsible for rendering the output. Similarly, a change in the content source would only affect the application logic provider. If the content itself changes, only the person responsible needs to make adaptations.

8.3.1 Model View Controller

The architectural pattern to strictly separate data (model) from presentation (view) became known as the model-view-controller (MVC). The separation of these components allows a simple replacement without one aspect affecting the other. Cocoon was built with this concept in mind. Other well known frameworks using this concept include Struts and JSF in the world of Java and Ruby on Rails. This latter gained much attention in recent years.

The MVC-model consists of three parts [Wess06]

- **Model** Data (Model), the domain-specific class where the attributes are stored, mainly a database
- **View** Presentation of the data in the user interface
- **Controller** Control or the processing of the user interaction or modification.

The isolation advances the exchangeability of the presentation (view) and the reusability of the business logic (model). Therefore the same business logic can be used by different output channels, e.g. a Web interface and a mobile client. This figure 8.2[2] is an illustration of how the three components interact with each other.

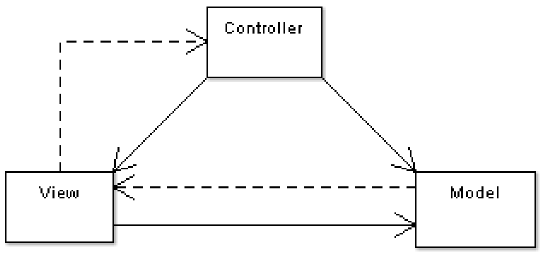


Figure 8.2: Model View Controller

8.3.2 Cocoon, MVC and Flowscripts

Within Cocoon, the *model* is represented by the contents of the pipeline. The controller is a composition of the pipeline components and how they are arranged within the application. Additionally, Cocoon has implemented its own controller concept called Flowscript. Flowscript is a JavaScript application programming interface that allows redirection of the application depending on various states. Transformers and serializers are responsible for the presentation of the content, the *view*.

8.4 The pipeline model

The application model of the World Wide Web is based on a request/response architecture. A Web client initiates a request using HTTP. The receiving server most commonly

[2]http://de.wikipedia.org/wiki/Model_View_Controller

either reads a static local file from disk and/or requests data from a database or then generates a dynamic response, usually in HTML. Other formats are also used. The server then sends this as a response to the client. This process can be visualized as individual components in a flowchart.

Cocoon uses a similar model, called the pipeline model. In this model, an XML file traverses the various components in a pipeline. These components generate, transform or serialize the traversing XML in different stages to produce the final presentation for the requesting client (figure 8.3). These stages can be seen as several chains in a pipeline. In Cocoon, one can define different pipelines and connect these to new pipelines.

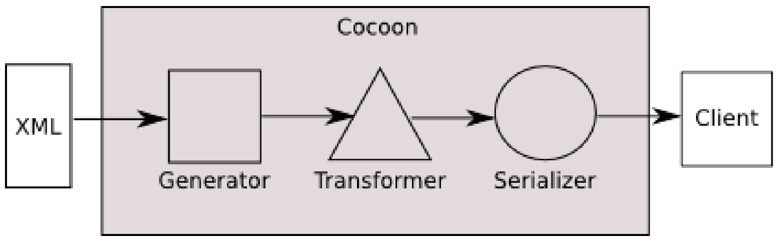


Figure 8.3: Main components within Cocoon

- **Generator** The pipeline's starting point: It generates XML content as SAX events and starts the processing through the pipeline. The default generator reads in static files.
- **Transformer** The XSLT transformer is usually used for transforming the content into the desired format for internal processing or final rendering. The transformer is the central point of a pipeline where most of the processing takes place. Several transformers can be linked together.
- **Serializer** The serializer is the end point of a pipeline. The serializer transforms the SAX events into binary or character streams for the consuming client.

Each component of the pipeline is individual and can be replaced without affecting the other components. This is extremely useful when developing an application. One can first define an XML document that is the start of the pipeline and then work on transforming the pipeline. The generator can easily be exchanged with another pipeline that queries a database and delivers the same structure.

A Cocoon pipeline can have two or more components. The most basic processing is to route a static XML file through the pipeline where a generator and serializer need to be defined. Usually, the transformer is needed as well.

We now have an idea what the framework is all about. Next, I will more closely examine each of these components and introduce a few others. This will provide the fundamental knowledge to understand the technology involved in building the mashup.

8.4.1 Generator

Each pipeline starts with the generator component. This component generates the XML content that is parsed and sent as SAX events down the pipeline. The default generator is the *File Generator* to read local XML files. It not only allows parsing of local files, but the generator can be used to invoke REST services, as will be seen later on. Several other generators exist to produce content from various other sources. Please see the Cocoon documentation for an explanation of all the generators. A second generator that I will use for Web data extraction is called the HTML Generator. Cocoon needs well-formed XML data to be processed in a pipeline; most Web sites, however, use HTML which is not well-formed. The HTML Generator is built up on Tidy[3]. *HTML Tidy is an open source program and library for checking and generating clean XHTML/HTML.*. Using this generator, one can read in any HTML source. It will be parsed with Tidy and transformed into well-formed XHTML. I will use this for Web scraping issues later on.

8.4.2 Transformer

A transformer takes the input from the generator and transforms it to a new document representation. For example, the XSLT Transformer transforms the input using a defined XSLT stylesheet and puts it into a new structure. Another useful operation of a transformer is to query a database. The XInclude transformer is helpful in acquiring different XML sources at once. We will use the XInclude Transformer in our application to acquire content based on a search result delivered by the HTML Transformer. Cocoon comes with several transformers for different purposes.

8.4.3 Serializer

A serializer is the end point in the pipeline. The document in its final presentation structure is serialized for end user representation. A default serializer is the HTML Serializer that returns an HTML document to the requesting client. Another commonly used serializer is the PDF Serializer. It takes XSL-FO input that is transformed to PDF using the underlying FOP[4] library.

8.4.4 Matchers

After learning about the basic and most important concepts involved in building a pipeline, the next step is how to access these pipelines. One of the main concepts of the Cocoon framework is to provide different output formats. An example application could provide two output formats, one a pipeline for HTML, and another one generating a PDF document. Each of these pipelines can be invoked using its own URL pattern. For example all requests using a URL ending with *.html* will invoke the HTML generating pipeline; every URL ending with *.pdf would activate the pipeline for generating the PDF. Matchers are the connection between the request and the pipeline that will process the request. Each request sent to Cocoon is evaluated to determine which pipeline to use for processing. The individual pipelines are defined in an XML based configuration, the so-called site map. When a match is found, processing of the request starts. The following list 8.1 defines two pipelines, one for HTML content and one for PDF generation.

Listing 8.1: Sitemap defining two different pipelines

```
<map:match pattern="*.html">
```

[3]http://tidy.sourceforge.net
[4]xmlgraphics.apache.org/fop

```
  <map:generate src="{1}.xml"/>
  <map:transform src="stylesheets/document2html.xsl"/>
  <map:serialize/>
</map:match>

<map:match pattern="*.pdf">
  <map:generate src="{1}.xml"/>
  <map:transform src="stylesheets/document2pdf.xsl"/>
  <map:serialize type="pdf"/>
</map:match>
```

Invoking the URL *sitemapdirectory/doc.html* would transform the file *doc.xml* into an HTML representation, whereas the URL *sitemapdirectory/doc.pdf* would generate the PDF *doc.pdf*. Using the matchers, one can easily achieve a clear separation of content and representation.

9 fmMusic backend

Previously, I talked about various kinds of mashups and introduced the main technologies used. Tools, or so called mashup enablers were discussed, which help extract information. I also introduced you to a few mashup platforms already available. Now it is time to start building our own mashup. First I will start to build the backend on top of Cocoon. It is used to extract data, invoke Web services and aggregate and transform the data. The output is then provided as separate services. The sources have already been defined in the specification chapter. In this chapter, I will discuss how to use Cocoon to deal with each of the sources.

The following figure 9.1 illustrates what content is used and from where it is acquired.

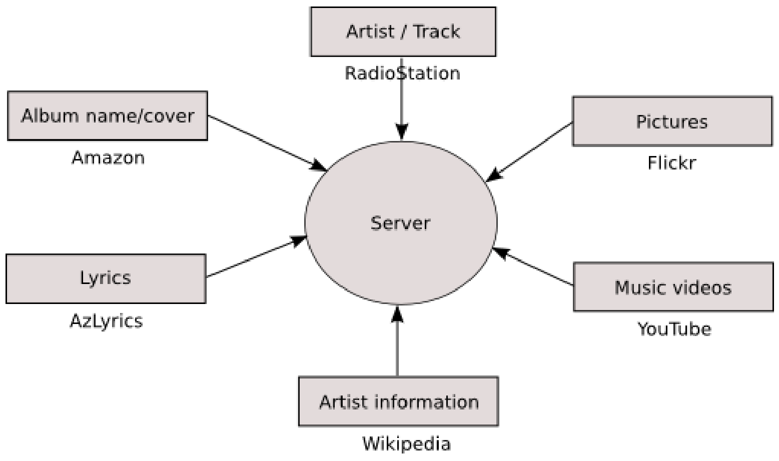


Figure 9.1: Sources that will be aggregated

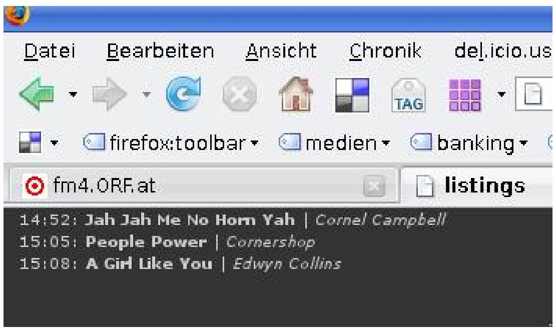
9.1 Filestructure

To get a basic Apache Cocoon up and running is not difficult, so I will not explain it further, but instead will refer to the project homepage[1]. Cocoon can be run using the provided Jetty, or it might be necessary to run a servlet container. I use Apache Tomcat[2], and deploy the .war file to it that you will build during the installation. Having done this, you can easily check if is Cocoon running browsing to *http://localhost:8080/cocoon/*. For our mashup I create the folder */servletContainer/-cocoon/fmMusic* that will be the root of the mashup, so all files and directories will be created inside it.

[1]http://cocoon.apache.org/2.1/installing/index.html
[2]http://tomcat.apache.org/

9.2 Scraping Fm4

The radio station *Fm4* as our station of choice resulting as the provider what artist and track is played at the moment. Figure 7.1 shows the window with an embedded media player, the track list and a few links and images. A closer look at the source code reveals that this is a frameset consisting of three frames, whereas the track list is a frame on its own. The option (available in Firefox) - *current frame* - *show only this frame* is really handy to display only the required frame. The result is that you get only the required frame with the corresponding URL that you need for extraction (figure 9.2).



14:52: Jah Jah Me No Horn Yah | Cornel Campbell
15:05: People Power | Cornershop
15:08: A Girl Like You | Edwyn Collins

Figure 9.2: Frame with tracklisting

Now only the last three played tracks are displayed on the page. Notice that the page is hosted at URL *http://hop.orf.at/img-trackservice/fm4.html*. In Cocoon, Web data extraction is accomplished using an XSLT Transformation, applied to an XHTML document. Therefore, the exact URL where the content is hosted is required. The two mashup enablers OpenKapow and Dapper have already been introduced. These tools allow direct action on the Document Object Model. Using such tools, one needn't worry if there are one or more frames. One can simply act on the final representation. This is the main advantage when acting on content that is loaded asynchronously and Cocoon probably will fail. I recommend using Firebug[3]. This is a Firefox add-on that lets one investigate where the asynchronous requests are sent to in order to identify the dynamic content.

Another benefit to knowing the exact source of a URL is the ability to reduce the data downloaded for each extraction. If there are many requests, this can save time in terms of traffic. Reducing the data download can also possibly limit processing resources for each extraction.

Before starting the extraction, I must think about what information I need and how I want to handle it in our mashup platform. These days, most tracks played on the radio are about 3-5 minutes in length. Between tracks, the radio host talks, so the time difference between each song can vary quite a lot. If I extract the information every three minutes, I could miss a song change a few seconds, so I will display the current song another three minutes, what is bad for the user. I could even miss a song that is less than 3 minutes. I will start with a one minute periodical check. If an update is missed, then after one minute I will display the correct song. Reducing it to a 30 second update should cause no problems.

[3]http://www.getfirebug.com/

The result of the radio station extraction will be the basis of our application. Each service will use either artist, track or both; therefore, the structure should be well defined within Cocoon. The XML in listing 9.1 is the one we transform the extracted content into.

Listing 9.1: XML structure for artist and track played

```
<radio provider="Fm4">
  <radioStation name="Fm4">
   <playing>
     <time>time</time>
     <artist>artist</artist>
     <track>trackname</track>
   </playing>
  </radioStation>
</radio>
```

Following elements are defined

- Root element *radio* with the attribute *provider*. The attribute provider is optional, but I defined it for future extensions when I may have other sources that deliver the information. Instead of extraction from HTML, this could be a fingerprint recording of the current song that would be submitted to Gracenote's MusicID[4] service. This service matches the fingerprint against its database and responds with the current track played. A few mobile phones already support such functionality.
- *radioStation*: This element comes with the attribute *name* that defines from which radio station the current content comes.
- *playing*: This is a wrapper around the main information.
- *time*: The time the current track started to play.
- *artist*: The name of the artist.
- *track*: The name of the track.

To start the actual extraction, look at the source code and note it is written in HTML. Therefore, it must be transformed into XHTML using the HTML Generator. This generator manipulates the DOM. The outcome must be inspected to ensure a correct stylesheet was created. Hence I start by creating the directory *radioStations* where I will place all the extractions. Inside I define a site map file, with the pipeline shown in the listing 9.2.

Listing 9.2: Pipeline showing HTML Generator output

```
<!-- Pipline for acquiring data from fm4 -->
<map:match pattern="fm4">
 <map:generate type="html" src="http://hop.orf.at/img-
    trackservice/fm4.html" label="source" />
 <map:serialize type="xml" />
</map:match>
```

[4]http://www.gracenote.com/business_solutions/music_id/

The pipeline matches the pattern *fm4* and thus can be invoked by browsing to *radioStation/fm4*. The generator type is *html*, which means that the *HTML Generator* is used for generation. The *HTML Generator* will download the HTML page, use Tidy to transform it to well-formed XHTML, passes it to the *XML Serializer* which delivers XML output. Pointing a browser to *http://localhost:8080/cocoon/fmMusic/radioStations/fm4* will show the well-formed version (listing 9.3) of the initial page.

Listing 9.3: Well-formed output from HTML Generator

```
 <?xml version="1.0" encoding="ISO-8859-1"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta content="HTML Tidy, see www.w3.org" name="generator"/>
    .....
    <style type="text/css">      &lt;!--
body {
   background-color: #333333;
   .....
}
.tracktitle {font-weight:bold;}
.artist {font-style:italic;}
      --&gt;
   </style>
</head>
<body>
   <div>10:49: <span class="tracktitle">Herculean</span> | <span
      class="artist">The Good The Bad &amp; The Q</span></div>
   <div>10:56: <span class="tracktitle">You Talk</span> | <span
      class="artist">Baby Shambles</span></div>
   <div>11:02: <span class="tracktitle">Foundations</span> | <
      span class="artist">Kate Nash</span></div>
</body>
</html>
```

The body tag of the track listing, contains three *div* elements, one for each track played. The timestamp reveals that the last updated entry is the third one. The stylesheet shown in listing B.1 will perform the extraction. When writing the stylesheet, keep in mind that the *HTML Generator* transforms all elements into the namespace *http://www.w3.org/1999/xhtml*.

Some additional processing occurs during extraction of the track name. This is because this radio station often plays remixes and live tracks and I want to eliminate this additional information. With all components together, I can complete the Web scraping, adding the stylesheet to the pipeline (listing 9.4). I add an additional transformer to point to the newly created stylesheet and when I reload the page I get the resulting XML shown in listing 9.5.

Listing 9.4: Final pipeline for scraping radio station data

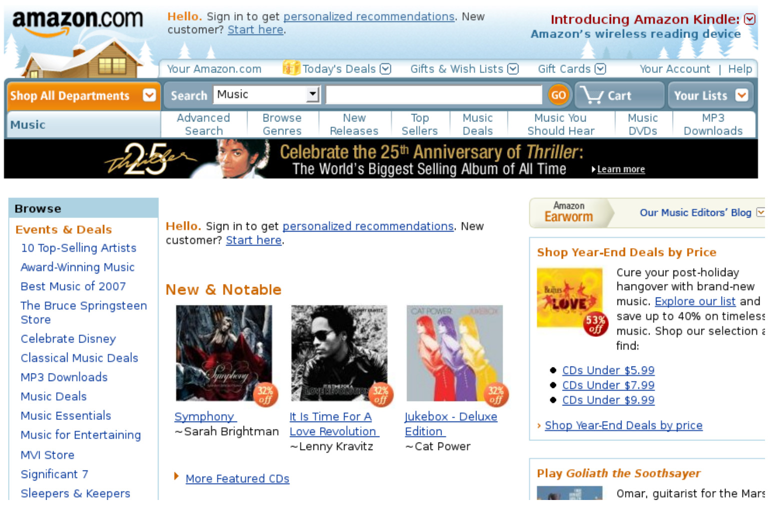
```
 <map:match pattern="fm4">
   <map:generate type="html" src="http://hop.orf.at/img-
      trackservice/fm4.html" label="source" />
   <map:transform type="xslt-saxon" src="xslt/fm4.xslt" />
   <map:serialize type="xml" />
</map:match>
```

Listing 9.5: XML structure for artist and track played

```
<radio provider="Fm4">
  <radioStation name="Fm4">
   <playing>
     <time>11:02</time>
     <artist>Kate Nash</artist>
     <track>Foundations</track>
   </playing>
  </radioStation>
</radio>
```

9.3 Amazon web services

Amazon.com[5] is probably the biggest online reseller and offers a broad collection of products. One of its product ranges is music and innumerable albums and singles are available (figure 9.3). Amazon provides several Web services, and the Amazon *Associates Web Service (A2S)*[6], formerly *E-Commerce Service*, provides a simple way to gain access to its product range. The Web service is free, as long as one brings traffic back to Amazon. It is common to have ads that link to Amazon on different pages. In the mashup, clicking on the cover will lead to the product page of the album.



Hello. Sign in to get personalized recommendations. New customer? Start here.

Cure your post-holiday hangover with brand-new music. Explore our list and save up to 40% on timeless music. Shop our selection a find:

Omar, guitarist for the Mars

Figure 9.3: Amazon's Music portal page

The A2S offers two main ways to retrieve information from its catalog. One can either search for products, using the *ItemSearch* operation, or the *ItemLookup* operation to look up detailed information for a product. To do a *ItemLookup*, the Amazon ID for the product is required. I will use both operations. First I will search for the Amazon product ID (ASIN) and save it in a database (not discussed here) so I do not have to look up a song each time it is played. Then I will use the ID for displaying information on-the-fly using the *ItemLookup* operation.

[5]http://www.amazon.com

[6]http://www.amazon.com/E-Commerce-Service-AWS-home-page/b/ref=sc_fe_c_0_15763381_1?ie=-UTF8&node=12738641&no=15763381&me=A36L942TSJ2AJA

9.3.1 Using the ItemSearch operation

The ItemSearch function offers different sets of parameters for each category (Music, Books, etc.) and the results can therefore be limited.

- Artist- All or part of artist's name
- Author - All or part of author's name
- Actor - All or part of actor's name
- Director - All or part of director's name
- Composer - All or part of composer's name
- Conductor - All or part of conductor's name
- Orchestra - All or part of orchestra name
- MusicLabel - All or part of record label name
- Publisher - All or part of publisher's name

However, we are limited as to the information extracted from the radio station. Artist is one of the parameters that can be searched, but title of the track cannot be used. To find the album on which the track appears, I will search using the artist's name. This will result in a listing of all albums and tracks associated with this artist. Then I will loop over the result set and compare the track names of each album with the current track to find a match. Once a match is found, I will use the title, ASIN and the URL to the cover image.

The following listing 9.6 shows the URL with all parameters for invoking the REST service.

Listing 9.6: Request URL for querying Amazon

```
http://webservices.amazon.de/onca/xml?Service=AWSECommerceService
    &SubscriptionId=YourId&Operation=ItemSearch&SearchIndex=Music&
    ResponseGroup=Medium,Tracks&Artist=Massive+Attack
```

Parameters provided with the request

- SubscriptionId: To use the A2S, one must register for the service and receive an ID that must be quoted when setting a query.
- Operation: Here I use the ItemSearch operation to search for data in the catalog.
- SearchIndex: The search should be limited to the category *Music*.
- ResponseGroup: Amazon offers three different response groups that differ in the amount of data that is turned back. We use the medium group which includes the URL to the images. Additionally, the tracks parameter is used, as these are not included in the medium result set.
- Artist: Finally I provide the name of the artist for whom I am looking.

A2S offers many different parameters and search options. A complete reference can be found on its programming guide[7].

The request results in ten different item sets, each representing one of the artist's albums. As can be seen in the response snippet 9.7, all the necessary information is provided.

Listing 9.7: Item snippet of search response

```
<Item>
  <ASIN>B000E5L8D4</ASIN>
  <MediumImage>
    <URL>http://ecx.images-amazon.com/images/I/21KFYDF6GVL.jpg</
       URL>
    <Height Units="pixels">160</Height>
    <Width Units="pixels">160</Width>
  </MediumImage>
 <Tracks>
  <Disc Number="1">
    <Track Number="1">Safe from harm</Track>
    <Track Number="2">Karmacoma</Track>
      ....
    <Track Number="14">Live with me</Track>
   </Disc>
 </Tracks>
</Item>
```

9.3.2 Matching and extracting response information

Before programming the extraction stylesheet I must first summarize what I need to do. As seen, I received about 14 albums that I have to loop through. Each album item contains a set of tracks that I will have to match. Once accomplished, I will then check if the item containing the matching track is of the binding type CD, as I do not want to select vinyl recordings. It is possible to find more than one match, so I also sort the sales ranking, and expect that the item with the best sales is the one I want. The stylesheet in Listing B.2 uses XSLT 2.0 techniques to extract the content. The transformation result is shown in listing B.3.

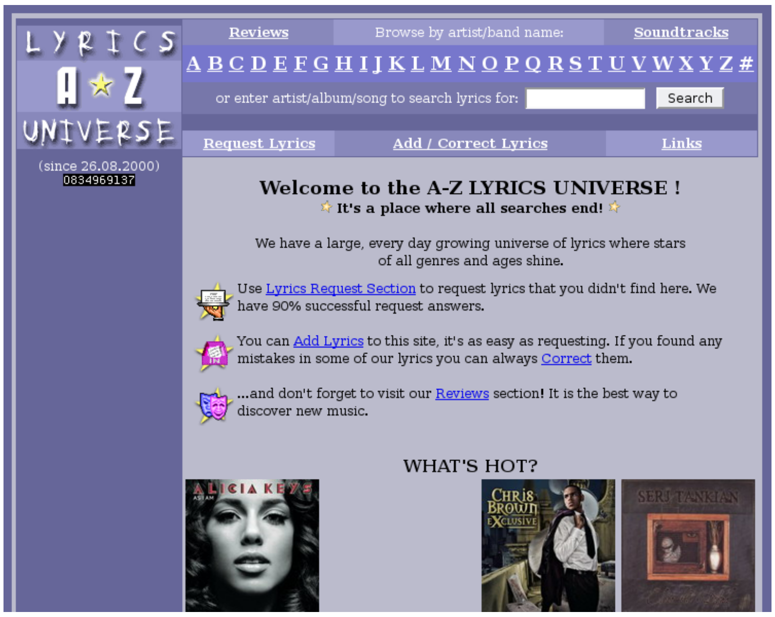
Using this information, the song being played can be displayed, and one can see the title and cover for the track and link directly to Amazon's product page. Most important, however, is the Amazon product ID (ASIN), which can be used to retrieve all other relevant information provided by Amazon at any time. This means I can easily implement client side mashup functions, loading the ASIN on the client and initiating asynchronous requests directly to Amazon, thereby avoiding traffic going through the server. We heard about the security restrictions using the XHTTPRequest inside a browser that forbids to access services outside your domain. Using JSON, this restriction can be bypassed. This is when Amazon's option to provide an external stylesheet comes in useful. A stylesheet can be set up on a server and can transform the request results into the JSON format. Amazon then uses the stylesheet on-the-fly and delivers the JSON response instead of XML.

[7]http://docs.amazonwebservices.com/AWSEcommerceService/2005-03-23/PgSearchingCatalog.html

9.4 Lyrics extraction

Searching for lyrics is one of the harder tasks. There are many different privately or community based lyrics platforms on the Web. However, none of these provides a complete set of lyrics for all songs. With no control of the content, there is no guarantee that all data is available at all times. It might be possible to have a Service Level Agreement (SLA) with the content provider, but many times information is extracted without the content owner's knowledge.

For the mashup, I will use *AZLyris.com*[8] as the lyrics provider (figure 9.4). I decided to use this one because it offers a large set of lyrics, an easy search option providing a parameter to the URL and information is easy to extract.



Welcome to the A-Z LYRICS UNIVERSE !
It's a place where all searches end!

We have a large, every day growing universe of lyrics where stars of all genres and ages shine.

Use Lyrics Request Section to request lyrics that you didn't find here. We have 90% successful request answers.

You can Add Lyrics to this site, it's as easy as requesting. If you found any mistakes in some of our lyrics you can always Correct them.

...and don't forget to visit our Reviews section! It is the best way to discover new music.

WHAT'S HOT?

Figure 9.4: AZLyrics.com - Content source for lyrics

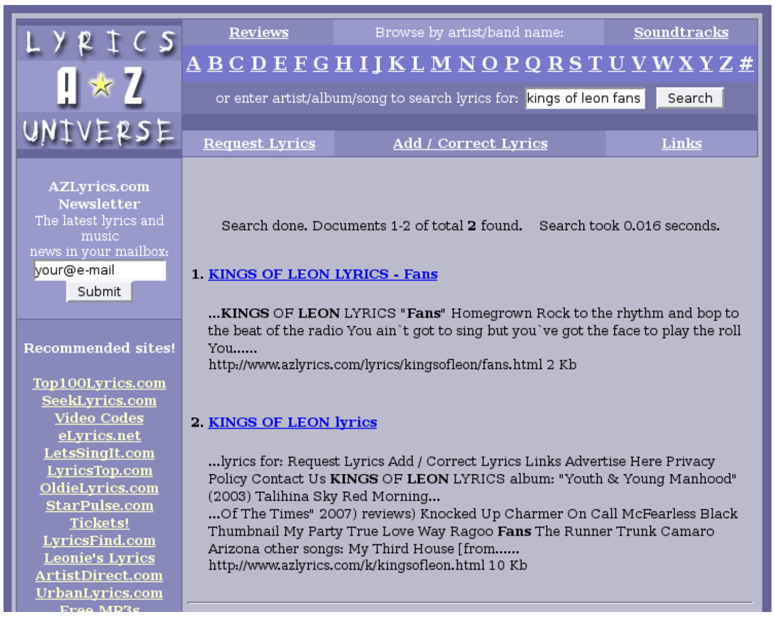
9.4.1 Sub-page navigation

AZLyrics is a platform where users employ a search function to look for an artist or track title. This search results in a listing, shown in figure 9.5, and hopefully the request is matched. The link leads directly to the lyrics page.

There is no option to access the content directly thus I need to find a way to request the search and simulate the navigation to the first result. Using a mashup enabler would have been a convenient option. However for demonstration purposes, I will present a way to use Cocoon for deep Web navigation.

So how can this be achieved using the Cocoon framework? I will start by setting up two different pipelines. The first one (listing 9.8) will be used to initiate the search for the lyrics and to extract the search results for further navigation. AZLyrics is submitting the parameters appended to the URL (using GET method), therefore I will URI-encoded artist and title and add them to the URL.

[8] http://AZLyris.com



Search done. Documents 1-2 of total **2** found. Search took 0.016 seconds.

1. KINGS OF LEON LYRICS - Fans

...**KINGS** OF **LEON** LYRICS "**Fans**" Homegrown Rock to the rhythm and bop to the beat of the radio You ain`t got to sing but you`ve got the face to play the roll You......
http://www.azlyrics.com/lyrics/kingsofleon/fans.html 2 Kb

2. KINGS OF LEON lyrics

...lyrics for: Request Lyrics Add / Correct Lyrics Links Advertise Here Privacy Policy Contact Us **KINGS** OF **LEON** LYRICS album: "Youth & Young Manhood" (2003) Talihina Sky Red Morning...
...Of The Times" 2007) reviews) Knocked Up Charmer On Call McFearless Black Thumbnail My Party True Love Way Ragoo **Fans** The Runner Trunk Camaro Arizona other songs: My Third House [from......
http://www.azlyrics.com/k/kingsofleon.html 10 Kb

Figure 9.5: Search results when searching after an artist and title

Listing 9.8: Pipeline for searching lyrics and extracting results

```
<map:match pattern="searchAz/*">
  <map:generate type="html"
     src="http://search.azlyrics.com/cgi-bin/azseek.cgi?q={1}"
         label="source" />
  <map:transform type="xslt-saxon" src="xslt/az_url.xslt" />
  <map:serialize type="xml" />
</map:match>
```

The second pipeline (listing 9.9) will use the extracted URL. Once again, the first result is expected to be the best.

Listing 9.9: Pipeline for lyrics extraction

```
<map:match pattern="getAzLyrics">
 <map:generate type="html" src="{flow-attribute:lyricUrl}"/>
 <map:transform type="xslt-saxon" src="xslt/azlyrics.xslt" />
 <map:serialize type="xml" />
</map:match>
```

9.4.2 Linking pipelines using flow logic

We have now defined two pipelines needed to get the lyrics. In listing 9.9, the pipeline used for the extraction, the generator src attribute does not link to a source. Instead, the source URL is provided dynamically from the flow function.

In order to use flow logic in our site map, I need to tell the sitemap where the JavaScript file is located. This is done using the *map:flow element* (listing 9.10). Now any function can be called from within a pipeline. In listing 9.11 the pipeline *lyricsLookup* is called, with the artist and track included in the URL. The pipeline invokes the function *getLyricURL* (listing 9.12) providing the artist and track parameters.

Listing 9.10: Including flow logic to Cocoon

```
<map:flow language="javascript">
  <map:script src="js/lyrics.js"/>
</map:flow>
```

Listing 9.11: Pipeline calling a flow function

```
<map:match pattern="lyricsLookup/*/*">
  <map:call function="getLyricUrl">
    <map:parameter name="artist" value="{1}" />
    <map:parameter name="track" value="{2}" />
  </map:call>
</map:match>
```

Flow logic

Following is a closer examination of the *getLyricURL* function listed in listing 9.12. First, another pipeline is invoked which defines global functions used to access XML elements. Inside the function, the parameters from the pipeline are parsed and assigned to variables. Using these variables, the URL for the search request is concatenated, the pipeline is requested and the result (listing 9.13) is loaded into the variable *xml*.

In the extraction stylesheet, I limited the results to the first one. Otherwise, I could implement additional logic to analyze the resulting URLs, or even parse all resulting pages. However I keep the first result and expect it to be the correct one.

Next, I assign the URL of the search result to a new variable *lyricUrl*. This URL should point to the lyrics of the track requested. The flow function *sendPage* can now be used to redirect Cocoon to the extraction pipeline. The *sendPage* method gives the option to include additional parameters, which then can be used within the pipeline. Therefore I define an object *parameters* and assign the URL to it. Finally, I redirect Cocoon to the pipeline (listing 9.9), providing the pipeline name and parameters object. It can be seen that the corresponding pipeline uses another dynamic attribute for the source attribute. Using *flow:attribute:parameterName* I can access the parameter I provided.

Listing 9.12: Flow function for lyrics extraction

```
cocoon.load("cocoon://fmMusic/resources/global.js"); // [1]
function getLyricUrl() {
var artist = cocoon.parameters.artist; // [2]
var track = cocoon.parameters.track;
var url = "cocoon:/searchAz/" + artist + "+" + track; //[3]
var xml = loadDocument(url); [4]
var lyricUrl = getElementContent("lyricUrl", xml); // [5]
if (lyricUrl != false) {
 // lyric found, return to chosen sitemap
 var parameters = null;
 parameters = { //[6]
 "lyricUrl": lyricUrl
 };
 cocoon.sendPage("getAzLyrics", parameters); //[7]
 }
}
```

Listing 9.13: Result of pipeline 9.8

```
<lyricUrl>resultURL</lyricUrl>
```

To finish, I look at the stylesheets used for extraction. The first one (listing B.4) is used for extracting the results. Here I apply the template to the first node *html:a* that contains the string *www.azlyrics.com/lyrics* inside its *href* attribute.

The second one B.5 is applied to the lyrics content page. Here I step down to the element *html:font* with attribute *size* containing value *5*. Inside this element I find the content I want to extract. Simply copying the elements as they are and adding an additional *div* element for the title results in a simpler styling of the title when displayed.

9.5 Pictures from Flickr

Another feature of the mashup will be the ability to view pictures of the artist currently being played. Flickr, a popular Web site that allow users to tag their photos, provides the pictures I want to display to the user. In this mashup, I will use the API it provides. As API is provided as REST service we will see how easily REST request can be implemented in Cocoon.

To access the API, a key is required. Since Yahoo! has acquired Flickr, a Yahoo account must first be created. This will be used to register the key. The API from Flickr is very powerful. Flickr has provided for nearly all manipulation and querying a service you can access. However in our mashup we just have to do a simple search after tags applied to the pictures.

Following a short explanation of search API parameters

- **method**: As I want to search for tags, I have to use *flickr.photos.search* to gain access to the search functionality.
- **api_key**: Personal key received upon registration.
- **tags**: We will search for *artist* and include additional tag **band** to get better results. Searching for artists' names such as *Doves* will definitely result in the wrong pictures, so including *band* should help.
- **sort**: A few sort orders are available, and I will use relevance as a criterion.
- **per_page**: This option let us define in how many results the query should end.
- **text**: One can not only search within tags, but also within the description.
- **tag_mode**: Defines if the search is for all tags provided (AND) or just one (OR).

In listing 9.14 how the REST service is used. A static URL with the required parameters simply points to the Web service, as with any other XML file. The corresponding response is the result of photo elements (Listing 9.15) containing attributes that again can be used to build a URL pointing to these.

Listing 9.14: Stylesheet for lyrics extraction

```
<map:pipeline type="caching">
  <map:parameter name="expires" value="access plus 10 hours"/>
  <map:match pattern="artist/*/*">
    <map:generate type="file" src="http://api.flickr.com/services
        /rest/?method=flickr.photos.search&amp;api_key=867
        a3a85d2a6a12bdac847622ba9912c&amp;tags={2},band&amp;sort=
        relevance&amp;tag_mode=all&amp;per_page={1}&amp;text={2}"
        label="source" />
    <map:transform type="xslt-saxon" src="xslt/flickr.xslt" />
    <map:serialize type="html" />
   </map:match>
</map:pipeline>
```

Listing 9.15: Element describing an image on Flickr

```
<photo id="11111111" owner="111111111" secret="1111111" server
   ="2141" farm="3" title="Algunos disquitos" ispublic="1"
   isfriend="0" isfamily="0"/>
...
```

The transformer then applies the stylesheet of Listing B.6 to generate the URL that points to the image. In the stylesheet I first assign the attributes to variables which are then used to concatenate the URL. The output of the stylesheet I see in Listing9.16 is a list *div* containing the *img* elements pointing to the pictures.

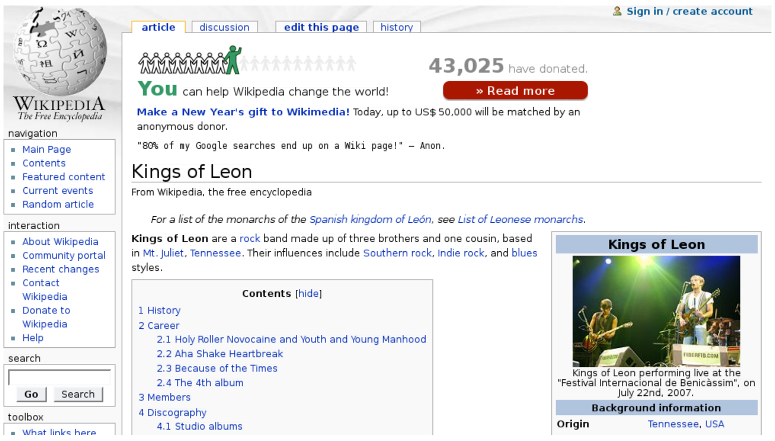
Listing 9.16: HTML pointing to Flickr images

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
   http://www.w3.org/TR/html4/loose.dtd">
<div id="flickrPix" provider="flickr">
<div id="photo">
  <img src="http://farm1.static.flickr.com/203/468558279
      _ed66572343_m.jpg"></div>
<div id="photo">
  <img src="http://farm1.static.flickr.com/185/468558267
      _98cd1f3204_m.jpg"></div>
  ....
</div>
```

9.6 Wikipedia

Wikipedia is the source of the information about the artist (figure 9.6). Wikipedia uses no frames, but mainly *div*-tags for their layout. So first I have to identify the tag that contains the main content with all the information about the artist. Within the description, Wikipedia has many cross references to other Wikipedia articles. However, this will not be allowed inside the fmMusic, so I will remove these links. Apart from that, everything will remain unchanged and the content will be displayed as casual HTML. In that way I can style it the way I want.

Following is a short explanation of the listing B.7, the extraction stylesheet. I first step down to the *div*-element with the ID *bodyContent*, which holds all information I want to display. Inside, I apply the identity templates to all nodes except the *span*-element, in order to remove the edit link. Then, an additional template that matches all *a*-elements is used to remove all links. In figure 9.7 I see the final result within a Web browser. Applying an additional stylesheet allows control over the layout of the result.



43,025 have donated.

You can help Wikipedia change the world!

Make a New Year's gift to Wikimedia! Today, up to US$ 50,000 will be matched by an anonymous donor.

"80% of my Google searches end up on a Wiki page!" – Anon.

Kings of Leon

From Wikipedia, the free encyclopedia

For a list of the monarchs of the Spanish kingdom of León, see List of Leonese monarchs.

Kings of Leon are a rock band made up of three brothers and one cousin, based in Mt. Juliet, Tennessee. Their influences include Southern rock, Indie rock, and blues styles.

Contents [hide]

1 History
2 Career
2.1 Holy Roller Novocaine and Youth and Young Manhood
2.2 Aha Shake Heartbreak
2.3 Because of the Times
2.4 The 4th album
3 Members
4 Discography
4.1 Studio albums

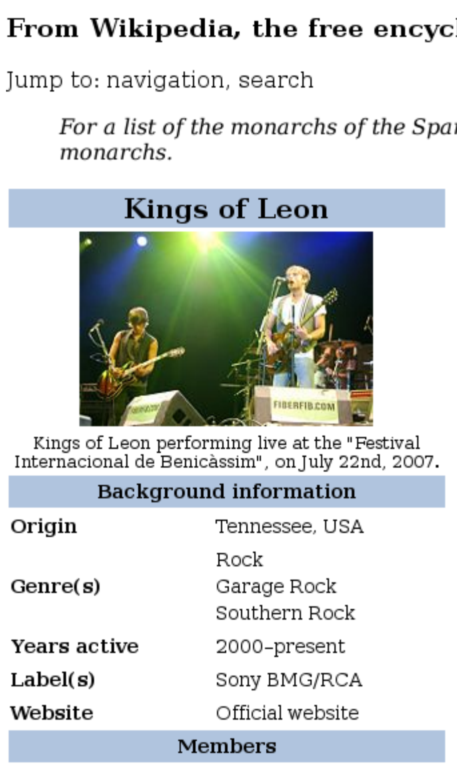
Kings of Leon

Kings of Leon performing live at the "Festival Internacional de Benicàssim", on July 22nd, 2007.

Background information

Origin Tennessee, USA

Figure 9.6: Artist information on Wikipedia



From Wikipedia, the free encyclopedia

Jump to: navigation, search

For a list of the monarchs of the Spanish kingdom of León, monarchs.

Kings of Leon

Kings of Leon performing live at the "Festival Internacional de Benicàssim", on July 22nd, 2007.

Background information

Origin	Tennessee, USA
Genre(s)	Rock Garage Rock Southern Rock
Years active	2000–present
Label(s)	Sony BMG/RCA
Website	Official website

Members

Figure 9.7: Extracted artist information from Wikipedia

9.7 Musicvideos from YouTube

When listening to a song on a radio station I want to provide users the possibility of watching the music video for the track, or if there is no music video for the track, there may be a live version of that song available. YouTube provides nearly all popular music videos and many live versions.

To display the video, an embedded object must be defined. I will implement a simple REST query to search YouTube and transform the result into the video object.

Here are the parameters used for searching the videos:

- **vq**: Used for keywords to search with. I will use the artist and title.
- **max-result**: Results will be limited to 5 videos.
- **order-by**: Order is based on relevance

The principle of our extraction mechanism will be the same as when defining the Flickr pipeline. The pipeline (listing 9.17 uses the file generator to start a request on YouTube's API. The result, shown in listing B.8 will then be transformed into HTML content that contains embedded flash objects that load and display the videos from YouTube (listing 9.8.

Listing 9.17: Pipeline using YouTube's API

```
<map:pipeline type="caching">
 <map:match pattern="searchAPI/*">
  <map:generate type="file"  src="http://gdata.youtube.com/feeds/
     api/videos?vq={1}&amp;max-results=5&amp;orderby=relevance"
     label="source" />
  <map:transform type="xslt-saxon" src="xslt/searchApi.xslt" />
  <map:serialize type="html" />
 </map:match>
</map:pipeline>
```

Listing 9.18: HTML representation using embeded videos

```
<?xml version="1.0" encoding="UTF-8"?>
<div id="videos" provider="youTube">
    <div id="vEntry">
        <div id="vTitle">Kings Of Leon - Fans</div>
        <div id="vVideo">
            <object width="280">
                <param name="movie" value="http://www.youtube.com
                   /v/-0sH3e_qr7c"/>
                <param name="wmode" value="transparent"/>
                <embed src="http://www.youtube.com/v/-0sH3e_qr7c"
                    type="application/x-shockwave-flash" wmode="
                   transparent" width="280"/>
            </object>
        </div>
        <div id="vDesc"/>
    </div>
    ..............
</div>
```

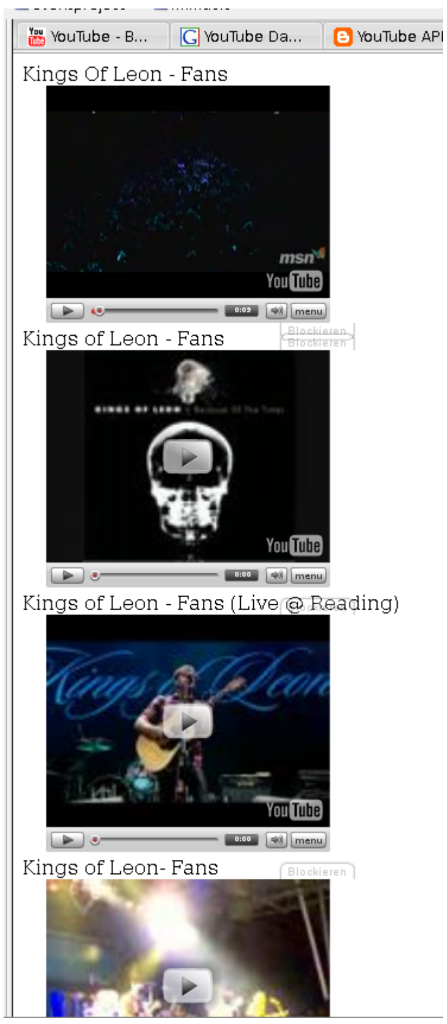


Figure 9.8: Result showing embedded videos

10 fmMusic frontend - a Firefox sidebar

In the previous chapter I described using Cocoon as our mashup server to collects all the sources I want to present to the user. An important part of a mashup is to build a comfortable user interface, so that the surplus of the application can be utilized. There are always several solutions for presenting the content, such as a simple HTML page with all data on one page, a tabbed interface, several widgets, et cetera.

However, I decided to build a Firefox extension, or more precisely, a sidebar, as the frontend for fmMusic, because I think it is a quick and convenient way to access the content. In figure 10.1 you see the open sidebar where the mashup sits. The sidebar consists of five different tabs, each containing a different content. Figures 10.2 and 10.3 display the different tabs. At this point, the mashup content is not really appealing, because it is part of an ongoing project and is still undergoing additional changes. Because the sidebar uses XUL elements styled with CSS, the styling can be implemented at a later stage.

Sidebar elements

- The first tab contains the title, cover, artist of the album the material appears on.
- The second tab displays the artist information from Wikipedia
- The third tab displays embedded videos with their description inside.
- The fourth tab provides the lyrics of the current song.
- On the last tab, some pictures of the artist are displayed.

Having a tabbed interface makes it easy to extend: for example I could easily add a new tab with information about events where the artist will be playing.

I will not provide a complete discussion of extension development in this chapter, but will explain some concepts and have a look how to mash it all together. Mozilla[1] provides a good online resources for extension development [Mozi08]; a separate tutorial for a sidebar extension can be found here [MozD06]. You will see that you can use the same concept to build a similar mashup using a widget-based front-end.

10.1 Extending Firefox

Firefox's user interface is written in XUL and JavaScript. When I build an extension for the Firefox browser, I simply modify a current part of the browser and add new widgets and functionality. Adding such widgets is done by adding new XUL DOM elements to the browser window and adding functionality with new scripts.

[1]http://developer.mozilla.org

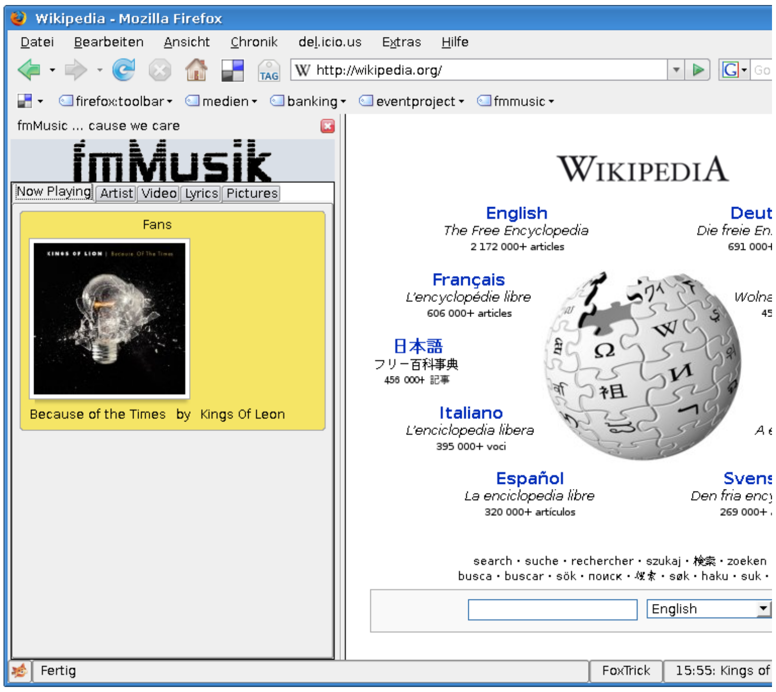


Figure 10.1: Music Extension sidebar inside the Firefox Browser

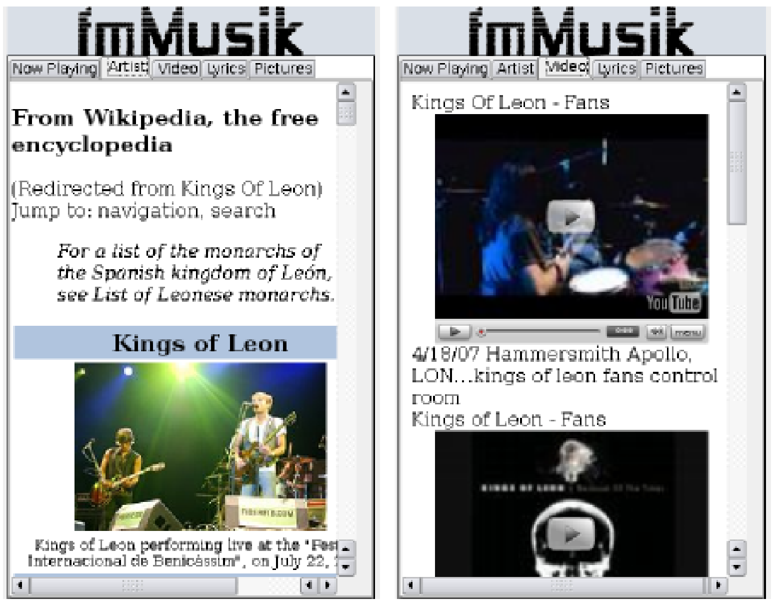
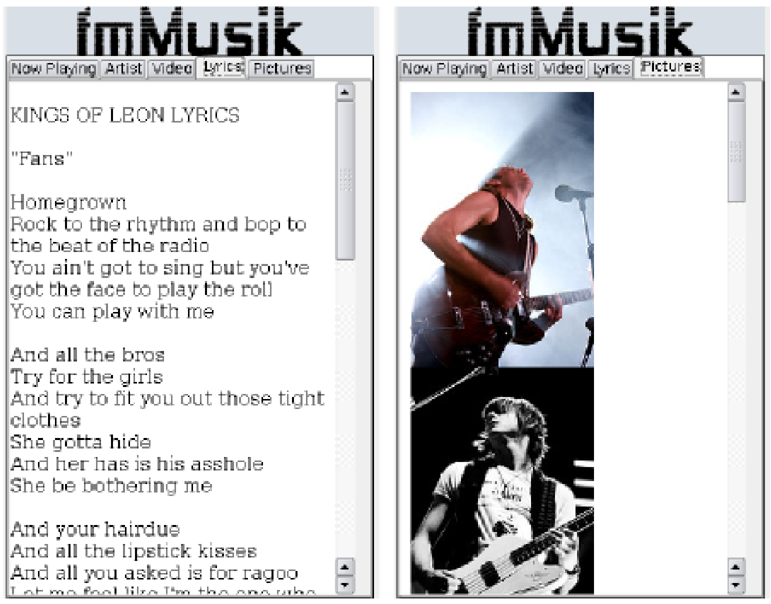

fmMusik
Now Playing Artist Video Lyrics Pictures
From Wikipedia, the free encyclopedia
(Redirected from Kings Of Leon)
Jump to: navigation, search
For a list of the monarchs of the Spanish kingdom of León, see List of Leonese monarchs.
Kings of Leon
Kings of Leon performing live at the "Fest Internacional de Benicàssim", on July 22,
fmMusik
Now Playing Artist Video Lyrics Pictures
Kings Of Leon - Fans
4/18/07 Hammersmith Apollo, LON...kings of leon fans control room
Kings of Leon - Fans

Figure 10.2: Artist and video tab

10.2 XUL Overlays

XUL Overlays are used to attach UI widgets to an XUL document at run time. An overlay file contains the attachment point where the new fragments are to be inserted, modified or removed. On closer inspection of figure 10.1, in the status bar on the bottom right you see a snippet of the current artist and track played, better seen in figure 10.4. Listing 10.1 provides the overlay code used to achieve this.



KINGS OF LEON LYRICS

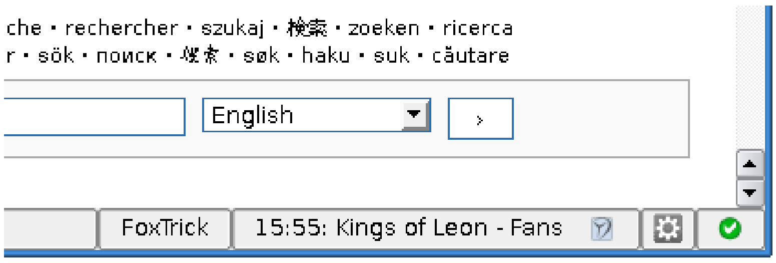
"Fans"

Homegrown
Rock to the rhythm and bop to the beat of the radio
You ain't got to sing but you've got the face to play the roll
You can play with me

And all the bros
Try for the girls
And try to fit you out those tight clothes
She gotta hide
And her has is his asshole
She be bothering me

And your hairdue
And all the lipstick kisses
And all you asked is for ragoo

Figure 10.3: Lyrics and pictures tab


che · rechercher · szukaj · 検索 · zoeken · ricerca
r · sök · поиск · 搜索 · søk · haku · suk · căutare

15:55: Kings of Leon - Fans

Figure 10.4: Statusbar displaying current song played

Listing 10.1: Overlay to display information in the browser's statusbar

```
<?xml version="1.0" encoding="UTF-8"?>
 <!DOCTYPE overlay >
 <overlay id="fmMusicStatusbar" xmlns="..." >
 <script xmlns="..." type="application/x-javascript" src="
    javascript/global.js" />
 <script xmlns="..." type="application/x-javascript" src="
    javascript/request.js" />
 <statusbar id="status-bar">
   <statusbarpanel id="fm4tracks" label="Loading..." context="
      popup">
     <label id="nowPlaying" value="Loading..."/>
     <toolbarbuttonid="sidebOpen" image="chrome://fmmusic/skin/
        icons/icon16x16.png" command="viewfmMusic"/>
   </statusbarpanel>
 </statusbar>
 </overlay>
```

10.3 Extended functionality with JavaScript

Using overlays, you can extend and manipulate the user interface of Firefox. With JavaScript, you can manipulate the XUL elements or DOM structure dynamically at runtime. You may have done this with HTML already. Working with the XUL DOM is similar to working with the HTML DOM.

In listing 10.2 I define three functions that give me the ability to display the track and the artist current playing. I will only describe this important concept for one tabs, since the basic principles for each tab's connection to the server are the same. I already introduced AJAX (2.8) and its XMLHttpRequest object (2.8.2), one of the foundations of this example.

At the beginning of the listing I attached a new *EventListener* to the window that invokes the function *getFm4Tracks* on loading of the window. *getFm4Tracks* calls the function *jsonRequest* using the URL that points to the pipeline that extracts the information of the radiostation. The function is then repeatedly called every minute.

The function *jsonRequest* encapsulates the XMLHTTPObject, and where the request to the backend is made. On change of the *XMLHttpRequest* object, the last function, *setStatusBar*, is called. If the request was successful, the response is transformed into a JSON object and the values are concatenated and assigned to the statusbar element identified by *nowPlaying*.

Listing 10.2: Javascript to dynamically assign values to the statusbar overlay

```
window.addEventListener("load", getFm4Tracks, false);
function getFm4Tracks() {
    jsonRequest('http://localhost/cocoon/fmMusic/radioStations/
        fm4.json1');
 self.setTimeout('getFm4Tracks()', 60000);}
function jsonRequest(url) {
   http_request = false;
   http_request = new XMLHttpRequest();
    if (!http_request) {
      println('Cannot create XMLHTTP instance');
      return false; }
   http_request.onreadystatechange = setStatusBar;
   http_request.open('GET', url, true);
   http_request.setRequestHeader("Content-Type","text/plain", "
      charset=utf-8");
   http_request.send(null);  }
function setStatusBar() {
 if (http_request.readyState == 4) {
      if (http_request.status == 200) {
   var myJSONObject = eval('(' + http_request.responseText + ')')
      ;
   var artist = myJSONObject.artist;
   gConfig.artist = artist;
   globalVar = artist;
   var track = myJSONObject.track;
   var station = myJSONObject.station;
   var time = myJSONObject.time;
   var artistTrack = time + ": " + artist + " - " + track + " ";
   document.getElementById('nowPlaying').value = artistTrack;
      } else {
```

```
    document.getElementById('fm4tracks').label = "Sorry, There was a
         problem contacting the server.";
       }
}
```

10.4 Creating a tabbed interface

The XML User Interface Language (XUL) is well designed for creating GUIs easily. The language itself has been available for quite a long time. However, after receiving a lot of attention in the beginning, XUL hasn't been used for a lot of applications outside of the Mozilla world. Maybe after the final release of Joost[2], an internet, p2p video player built on XUL or Mozilla's Prism[3] initiative, more developers will be attracted to build applications on XUL.

In the previous figure 10.1 you can see inside the header image (fmMusic) in the sidebar image, and below that, the tabcontainer, containing five tabs. XUL comes with a box model[4] for laying out the content. Vertical boxes align their elements vertically and horizontal box horizontally.

The sidebar's heading (listing C.1) is defined in the vertical box with Id *vboxHeader* [1], where an image element [2] points to the local image file. Below you find the tab-interface. Tabs are defined inside a *tabbox* [3] container. First you define all the *tab* [4] elements for each tab inside the *tabs* element, similar to the defining table columns in HTML. The *tabpanels* [5] contain one *tabpanel* [6] element for each tab. Following is an excerpt of the *Now Playing* tab, which shows a groupbox [7] with labels and an imagelink. These contain default values that are dynamically replaced when the content is delivered from the backend. I have done the same in the statusbar example. The request is initiated when the sidebar opens, and on receipt of content, the replacement is done.

Using XUL allows great flexibility for creating a front-end. With a large set of elements user interfaces, the GUI can be developed like any other desktop application. Providing the possibility to embed HTML code, offers you a simple way to create content that can be embedded in casual HTML pages as well.

[2]http://www.joost.com/
[3]http://labs.mozilla.com/2007/10/prism/
[4]http://www.xulplanet.com/tutorials/xultu/boxes.html

Part V

Future Issues

11 Legal issues

Creating mashups often involves using content from sources outside your own domain. Sometimes you may use mashup enablers or Web data extraction techniques to acquire this content. Dealing with external data, even with public APIs, has to be done carefully, as you can come in conflict with the content owner.

11.1 Copyright

Copyright covers the exclusive rights of the creator of an original work limited period of time, with respect to reproduction/copying, communication, adaption and performance. Copyright law is enacted by national governments and may vary between different countries.

Mashups walk a thin line when the whole or parts of the original content is being reproduced. [Brie06] puts it this way:

> Therefore, mashups and remix will inevitably encounter legal problems when the whole or a substantial part of the original material has been reproduced, copied, communicated, adapted or performed - unless a permission has been given in advance through a voluntary open content licence like a Creative Commons licence, there is fair dealing involved (the scope of which is extraordinarily narrow), a statutory licence exists, or permission has been sought and obtained from the copyright owner.

11.2 Copyright in the digital age

The problem of sharing is nothing new; people have always shared. Two decades ago, people exchanged music cassettes, CD's or videos and may have copied them for personal use. The last decade saw the rise of peer-to-peer technologies that allowed you to share not only with your immediate social group, but with the whole world, and that has made it a big issue.

Mashups often use screen scraping mechanisms to access content; doing this can definitely be seen as taking someone else's content and using it without permission. Mashups, however, often focus on giving existing content new value or are a creative act and do not compete with the primary market of the copyright owner. [Suz06] suggests that the copyright law should be reformed in a manner as to allow certain reuses of copyright material if the derivative do not impact the primary market of the copyright owner.

Lawrence Lessig, a Stanford Law Professor, believes [Farb06]

> For the first time in history human expression by default is subject to regulation because of two architectural features.
>
> First, the cultural objects or products created on a computer can be easily copied, and secondly the default copyright law requires the permission of

the owner. The result is that by law you need permission to engage in acts of remixing.

So is it the case that you become a criminal when being creative? Is it a bad thing to create something useful? I think it is time to rethink the copyright law in favor of mashups.

11.3 Data ownership

In a WIRED article [McHu07], the case of Ryan Sit, a software developer from San Diego, is discussed. Sit founded a site *Listpic* that used extraction robots to extract listings from craigslist for-sale listings and presented them in a easier-to-navigate, more attractive format. The service was a huge success and quickly became popular within craigslist users. The craigslist CEO was not happy with the service, as it took quite a lot from its revenue and asked Sit to stop the service and banned its robots.

This definitely is a case study of how mashups are dependent on the content owners willingness to share content with a the mashup creator. Listpic offered a better user experience than craigslist and therefore quickly became popular. The lockout by craigslist was mainly a downside for the end user, who lost the better service. Meanwhile Listpic pulls data from a different set of listing and continues to provide its service.

Some other companies are happy when mashups list their content to gain higher visibility. The pricecomparison site *Geizhals* started with just a few small resellers; now many big companies are on it. Such a listing is crucial to gain sellers' attention; not having a listing means you will be overlooked.

11.4 Conclusion

For the next few years, while copyright law stays the same, the views on mashups will always be divided. On one side, creators will claim some kind of copyright infringement when someones uses their work in a new way or context. On the other side, creating a mashup may be perceived as a creative act or innovation, where the added or new value is in the interest of us all.

The main question in the future will be how to handle these copyright infringements. Should someone be sentenced and punished for a creative act, or should there be at least some kind of indulgence? The last years have demonstrated that trying enforce copyright law does not work. Think about Digital Rights Management (DRM), where a user was allowed to play his music only on a limited number of devices. Music labels have started to offer DRM free music again. I think it will be the same with content within mashups; as long as the mashup creator does not resell the content, it will not be an infringement of copyright.

12 Future Trends

Every now and then some technologies are outed as being on the rise, and claim to provide new solutions that allow users to to build *instant software*. Whereas those promises were never fulfilled, mashups and service-oriented-architecture have already proven the concept in terms of rapid reuse and large-scale deployment.

Gartner [Gart08] identifies mashups on its *Top 10 Strategic Technologies for 2008* in place six and sees mashups as the dominant composite enterprise applications by the year 2010.

The economics of software development is starting to change. Applications will be developed in days or weeks instead of months or years required in traditional software development. Until now, local problems that may only be an issue for a short time could not be addressed because with the costs of development it was not feasible to fix them. With a drastic reduction of these costs it is possible to address more problems, or deliver a wider set of solutions that extend the value of existing business data. This awareness results in a lower barrier to committing a bug fix or adjusting the requirements of an adaptation of an existing solution. Within these trends the individual end-user will become a more critical contributer to the application design, testing and development process [Prot07].

Mashup applications will provide modularized applications and services that can be connected. What common 5GL development tools already offer will be available in the new mashup development environments. This simplification of application development will enable domain experts and business users to create solutions without the involvement of the IT department. More technically advanced users could extend prebuilt components to fit their needs. QEDWiki is heading in this direction but is still far from being user friendly.

IT departments that use extensive service oriented architecture are already mashup-ready and can provide functional components to these mashup platforms with little or no development efforts. Creating these mashable pieces will remain the task of software developers.

The main problem to overcome is the lack of standards within the evolving mashup platforms to allow interoperability.

A lot of mashups are created by private people with no commercial interest, nevertheless more and more companies have started creating mashups with commercial interest. [Gott07] points out common problems mashup creators have to deal with:

Provider dependency APIs can change without notice and the mashup probably breaks

Discontinued services Terms of services under which APIs are provided typically gives the provider the right to discontinue the service without notice

Competition from API provider The provider may decide to provide the service himself, when the mashup service is successful

Data dependency Mashups initially do not have any content and therefore are vulnerable to changes on the provider side

12.1 Role of Mashups in the Enterprise

Enterprise IT has undergone constant change in recent years, with the shift to a service-oriented architecture as one of the most important changes. The rise of SOA has already laid important groundwork for the upcoming mashup technologies. However the focus was mainly on service creation, governance and infrastructure issues, not on ease of consumption. This is where mashup platforms come into play providing a way to combine these existing services into small applications without any greater software development processes. With the decoupling of services from interface issues a much higher service reusability can be reached.

Until now spreadsheets were widely used to enable users to build quick customized solutions to various business needs. Mashup platforms infiltrate this area by enabling a comparable application lifecycle but with the advantage of sitting on top of already defined services and to be more flexible in customization. Consequently mashups will allow business users and individuals to design advanced customized solutions in a short time which support their decision making. An important part will be assigned to the IT department to create services and modules that are not only available on these platforms but furthermore allow these modules to be connectable.

12.1.1 Business Intelligence

In enterprises one of the main tasks of a mashup platform lies in quick and simple applications for business intelligence (BI) enabling more business units to apply quicker decisions to situational problems. These mashup platforms will not compete with current powerful BI tools, however they will find a place for quick adaption and make BI available to companies where these specialized tools aren't a fit or companies that simply cannot afford these tools. These small and medium-sized businesses (SMB) have already started to gather an enormous amount of data and need tools that allow them to interpret this data. This is an enormous market segment for mashup tools.

12.2 Mashup tools and platforms

In this thesis we already talked about *mashups*, *mashup enablers* and *mashup platforms*, someone else could talk about *mashup builders*. So what tools provide mashup development and what are their aims?

When defining mashup tools it is hard to draw a clean separation and categorize these tools. Generally we think that A. Jhingran [Jhin06] tree layers for a mashup composition (section 1.5.1), *Ingestion*, *Augmentation* and *Presentation* is a good approach for a categorisation, even if a lot of tools traverse two or all three layers. Dapper (section 5.1), is a tool which can be used for web data extraction or accessing RSS feeds (Ingestion), furthermore you can combine different Dapps (Augmentation) and render it in various output formats, e.g. a Flash widget (Presentation).

A lot of tools now and in future will not compete but supplement each other. Some tools will mainly be used for visual presentation within a mashup development environment, where modules are combined and arranged together; some other tools will provide the services and modules for these. Even if most of the presentation mashups will try to allow non-technical users to build situational applications, it is not a must. More advanced connectivity will probably be hard to implement in a user-friendly way, such that that non programmers can take advantage of it.

12.3 Conclusion

Mashups will, without question, provide a new way to analyze data or solve problems, whether in the enterprise or for private use on the web. The World Wide Web made it possible for nearly anyone to access all kind of information. Web 2.0, has helped to enhance all kinds of information on the web, providing meta data, tags or other social context, for a better use and higher information quality.

Mashups will provide a new way to organize, structure and present information, allowing users to easily combine nearly any sources available. People now have to browse to several resources to get to all the information they need, in the future they will simply create their own mashup to get all information into one place.

To conclude, I will present a statistic from ProgrammableWeb shown in figure 12.1, where you can see a timeline of available mashups in the last six months. A spectacular rise from about 2100 to 2700 is shown, what is a nearly a 30% gain.

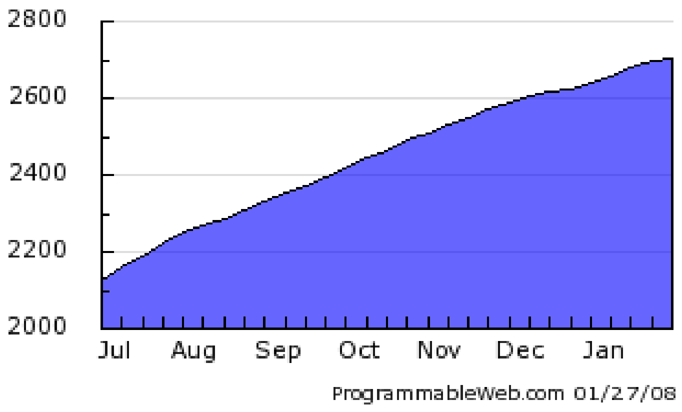


Figure 12.1: Mashup Timeline - New mashups on ProgrammableWeb, last 6 months

Part VI

Appendix

A YouTube API response

Listing A.1: YouTube response

```
<entry>
  <id>http://gdata.youtube.com/feeds/api/videos/ZTUVgYoeN_o</id>

  <published>2007-02-16T20:22:57.000Z</published>
  <updated>2007-02-16T20:22:57.000Z</updated>

  <category scheme="http://schemas.google.com/g/2005#kind"
    term="http://gdata.youtube.com/schemas/2007#video"/>

  <category scheme="http://gdata.youtube.com/schemas/2007/
     keywords.cat"
    term="Steventon"/>
  <category scheme="http://gdata.youtube.com/schemas/2007/
     keywords.cat"
    term="walk"/>
  <category scheme="http://gdata.youtube.com/schemas/2007/
     keywords.cat"
    term="Darcy"/>

  <category scheme="http://gdata.youtube.com/schemas/2007/
     categories.cat"
    term="Entertainment" label="Entertainment"/>

  <title type="text">My walk with Mr. Darcy</title>

  <content type="html"><div ... html content trimmed ... ></
     content>

  <link rel="self" type="application/atom+xml"
    href="http://gdata.youtube.com/feeds/api/videos/ZTUVgYoeN_o
       "/>
  <link rel="alternate" type="text/html"
    href="http://www.youtube.com/watch?v=ZTUVgYoeN_o"/>

  <link rel="http://gdata.youtube.com/schemas/2007#video.
     responses"
    type="application/atom+xml"
    href="http://gdata.youtube.com/feeds/api/videos/ZTUVgYoeN_o/
       responses"/>
  <link rel="http://gdata.youtube.com/schemas/2007#video.ratings"
    type="application/atom+xml"
    href="http://gdata.youtube.com/feeds/api/videos/ZTUVgYoeN_o/
       ratings"/>
  <link rel="http://gdata.youtube.com/schemas/2007#video.
     complaints"
```

```
      type="application/atom+xml"
      href="http://gdata.youtube.com/feeds/api/videos/ZTUVgYoeN_o/
          complaints"/>
    <link rel="http://gdata.youtube.com/schemas/2007#video.related"
      type="application/atom+xml"
      href="http://gdata.youtube.com/feeds/api/videos/ZTUVgYoeN_o/
          related"/>

    <author>
      <name>Elizabeth Bennet</name>
      <uri>http://gdata.youtube.com/feeds/api/users/liz</uri>
    </author>

    <media:group>
      <media:title type="plain">My walk with Mr. Darcy</media:title
          >
      <media:description
        type="plain">Walk in the beautiful gardens of Steventon</
            media:description>
      <media:keywords>Steventon, walk, Darcy</media:keywords>
      <yt:duration seconds="79"/>
      <media:category label="Entertainment"
        scheme="http://gdata.youtube.com/schemas/2007/categories.
            cat">Entertainment</media:category>
      <media:content
        url="rtsp://rtsp.youtube.com/youtube/videos/ZTUVgYoeN_o/
            video.3gp"
        type="video/3gpp" medium="video" isDefault="true"
            expression="full"
        duration="215" yt:format="1"/>
      <media:player url="http://www.youtube.com/watch?v=ZTUVgYoeN_o
          "/>
      <media:thumbnail url="http://img.youtube.com/vi/ZTUVgYoeN_o
          /2.jpg"
        height="97" width="130" time="00:00:03.500"/>
      <media:thumbnail url="http://img.youtube.com/vi/ZTUVgYoeN_o
          /1.jpg"
        height="97" width="130" time="00:00:01.750"/>
      <media:thumbnail url="http://img.youtube.com/vi/ZTUVgYoeN_o
          /3.jpg"
        height="97" width="130" time="00:00:05.250"/>
      <media:thumbnail url="http://img.youtube.com/vi/ZTUVgYoeN_o
          /0.jpg"
        height="240" width="320" time="00:00:03.500"/>
    </media:group>

    <yt:statistics viewCount="93"/>

    <gd:feedLink rel="comments"
      href="http://gdata.youtube.com/feeds/api/videos/ZTUVgYoeN_o/
          comments"/>
  </entry>
```

B Stylesheets and transformation results

Listing B.1: XSL Transformation for extraction of artist and track played

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/
   XSL/Transform"
  xmlns:html="http://www.w3.org/1999/xhtml" exclude-result-
     prefixes="html">
 <xsl:output indent="no" />
 <xsl:template match="/">
  <radio provider="Fm4">
   <radioStation name="Fm4">
    <xsl:apply-templates select="//html:body" />
   </radioStation>
  </radio>
 </xsl:template>
 <xsl:template match="html:body">
  <xsl:for-each select="html:div[last()][html:span[2]!='']">
   <xsl:sort select="replace(text()[1],':','')" order="descending
      "/>
   <playing>
    <time>
     <xsl:value-of select="replace(text()[1],':\s','')" />
    </time>
    <artist><xsl:value-of select="html:span[2]" /></artist>
    <xsl:variable name="track" select="html:span[1]"/>
    <track>
     <!-- remove all (live) and (Rmx) Strings -->
     <xsl:value-of select="replace($track,'\s+[\(]+live[\)]+|\s
        +[\(][a-zA-Z\s]+[Rr]mx[\)]','')" /></track>
   </playing>
  </xsl:for-each>
 </xsl:template>
</xsl:stylesheet>
```

Listing B.2: Stylesheet for matching and extracting album information

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:album="http://webservices.amazon.com/AWSECommerceService
    /2005-10-05"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 exclude-result-prefixes="album xs">
 <xsl:param name="trackname"/>
 <xsl:param name="artist"/>
 <xsl:output indent="no" />

 <xsl:template match="/">
```

```
  <album provider="amazon">
  <trackname><xsl:value-of select="$trackname"/></trackname>
  <xsl:apply-templates select="for $i in //album:Item/album:
     Tracks/album:Disc/album:Track
        return $i[.=$trackname]/ancestor::album:Item[album:
            ItemAttributes/album:Binding[contains(.,'CD')]]">
   <xsl:sort select="number(album:SalesRank)" order="ascending"
      />
  </xsl:apply-templates>
  </album>
 </xsl:template>

 <xsl:template match="album:Item">
   <album>
     <amazonId><xsl:value-of select="album:ASIN"/></amazonId>
     <amazonDetailUrl><xsl:value-of select="normalize-space(album
        :DetailPageURL)"/></amazonDetailUrl>
    <xsl:apply-templates select="album:ItemAttributes"/>
    <imgUrl><xsl:value-of select="normalize-space(album:
       MediumImage/album:URL)"/></imgUrl>
   </album>
 </xsl:template>

 <xsl:template match="album:ItemAttributes">
  <title><xsl:value-of select="normalize-space(album:Title)"/></
     title>
  <releaseDate><xsl:value-of select="normalize-space(album:
     ReleaseDate)"/></releaseDate>
  <artist><xsl:value-of select="normalize-space(album:Artist)
     "/></artist>
  <binding><xsl:value-of select="normalize-space(album:Binding)
     "/></binding>
 </xsl:template>
</xsl:stylesheet>
```

Listing B.3: Extracted album information for internal representation

```
<album>
  <amazonId>B000006045</amazonId>
  <amazonDetailUrl>http://www.amazon.de/gp/redirect.html%3FASIN=
     B000006045%26tag=ws%26lcode=xm2%26cID=2025%26ccmID=165953%26
     location=/o/ASIN/B000006045%253FSubscriptionId=0
     DNP7XNME3TMSH5DSA02</amazonDetailUrl>
  <title>Mezzanine</title>
  <releaseDate>1998-04-17</releaseDate>
  <binding>Audio CD</binding>
  <imgUrl>http://ecx.images-amazon.com/images/I/31QY7BFWVHL.jpg</
     imgUrl>
</album>
```

Listing B.4: Stylesheet extracting search results 9.8

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/
   XSL/Transform" xmlns:html="http://www.w3.org/1999/xhtml"
   exclude-result-prefixes="html">
```

```
<xsl:output indent="no" />
 <xsl:template match="/">
 <lyrics provider="az">
   <xsl:apply-templates select="//html:a[contains(@href,'www.
      azlyrics.com/lyrics')][1]" />
  </lyrics>
 </xsl:template>

 <xsl:template match="html:a">
   <lyricUrl><xsl:value-of select="@href" /></lyricUrl>
 </xsl:template>
</xsl:stylesheet>
```

Listing B.5: Stylesheet for lyrics extraction

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/
   XSL/Transform" xmlns:html="http://www.w3.org/1999/xhtml"
   exclude-result-prefixes="html xhtml">
 <xsl:output indent="no" />
 <xsl:template match="/">
  <div id="lyricsMain">
   <xsl:apply-templates select="//html:font[@size='5']" />
  </div>
 </xsl:template>

 <xsl:template match="node()|@*">
    <xsl:copy copy-namespaces="no">
    <xsl:apply-templates select="@*"/>
    <xsl:apply-templates/>
    </xsl:copy>
  </xsl:template>

 <xsl:template match="html:font">
   <xsl:apply-templates/>
 </xsl:template>

 <xsl:template match="html:font/html:a | html:i[contains(.,'
    Thanks')]
       | text()[normalize-space(.)='[']
       | text()[normalize-space(.)=']']"/>
 <xsl:template match="html:font/html:b">
   <div id="lyricTitle">
    <xsl:value-of select="."/>
   </div>
 </xsl:template>
</xsl:stylesheet>
```

Listing B.6: Stylesheet to build pointers to the images on Flickr

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:html="http://www.w3.org/1999/xhtml"
 exclude-result-prefixes="html">
 <xsl:output indent="no" />
 <xsl:template match="/">
  <div id="flickrPix" provider="flickr">
   <xsl:apply-templates select="//photos/photo"/>
  </div>
 </xsl:template>
 <xsl:template match="photo">
  <div id="photo">
    <xsl:variable name="f-id" select="@farm"/>
    <xsl:variable name="s-id" select="@server"/>
    <xsl:variable name="id" select="@id"/>
    <xsl:variable name="secret" select="@secret"/>
    <xsl:variable name="url" select="concat('http://farm',@farm
       ,'.static.flickr.com/',@server,'/',@id,'_',@secret,'_m.jpg
       ')"/>
   <img src="{$url}"/>
  </div>
 </xsl:template>
</xsl:stylesheet>
```

Listing B.7: Stylesheet to extract artist information from Wikipedia

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/
   XSL/Transform"  xmlns:html="http://www.w3.org/1999/xhtml"
   exclude-result-prefixes="html">
<xsl:output indent="no" />
 <xsl:template match="/">
  <artist provider="wikipedia">
   <xsl:apply-templates select="//html:div[@id='bodyContent']" />
  </artist>
 </xsl:template>

 <xsl:template match="html:div[@id='bodyContent']">
  <description>
   <xsl:apply-templates select="node()[not(html:span[class='
      editsection'])]" />
  </description>
 </xsl:template>

 <!-- remove all Links -->
  <xsl:template match="html:a">
   <xsl:apply-templates/>
  </xsl:template>

 <xsl:template match="node()|@*">
    <xsl:copy>
     <xsl:apply-templates select="@*"/>
     <xsl:apply-templates/>
    </xsl:copy>
 </xsl:template>
</xsl:stylesheet>
```

Listing B.8: YouTube query response (shortened)

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<feed xmlns='http://www.w3.org/2005/Atom' xmlns:openSearch='http
   ://a9.com/-/spec/opensearchrss/1.0/' xmlns:media='http://
   search.yahoo.com/mrss/' xmlns:yt='http://gdata.youtube.com/
   schemas/2007' xmlns:gd='http://schemas.google.com/g/2005'>
 <title type='text'>YouTube Videos matching query: kings of leon
    fans</title>
 <logo>http://www.youtube.com/img/pic_youtubelogo_123x63.gif</
    logo>
 <entry>
  <id>http://gdata.youtube.com/feeds/api/videos/-0sH3e_qr7c</id>
  <published>2007-06-17T23:28:26.000-07:00</published>
  <updated>2008-01-02T00:43:08.000-08:00</updated>
  <category scheme='http://gdata.youtube.com/schemas/2007/
     keywords.cat' term='room' />
  <title type='text'>Kings Of Leon - Fans</title>
  <content type='text'>4/18/07 Hammersmith Apollo, LON</content>
  <link rel='self' type='application/atom+xml'
   href='http://gdata.youtube.com/feeds/api/videos/-0sH3e_qr7c'
      />
  <author>
   <name>coolconspirator</name>
   <uri>
    http://gdata.youtube.com/feeds/api/users/coolconspirator
   </uri>
  </author>
  <media:group>
   <media:title type='plain'>Kings Of Leon - Fans</media:title>
   <media:description type='plain'>4/18/07 Hammersmith Apollo,
      LON</media:description>
   <media:keywords>control, of, fans, leon, kings, room</media:
      keywords>
   <yt:duration seconds='237' />
   <media:category label='Music'
    scheme='http://gdata.youtube.com/schemas/2007/categories.cat
       '>Music</media:category>
   <media:content url='http://www.youtube.com/v/-0sH3e_qr7c'
    type='application/x-shockwave-flash' medium='video' isDefault
       ='true'
    expression='full' duration='237' yt:format='5' />
   <media:player
    url='http://www.youtube.com/watch?v=-0sH3e_qr7c' />
  </media:group>
  <yt:statistics viewCount='309735' />
  <gd:rating min='1' max='5' numRaters='657' average='4.91' />
  <gd:comments>
   <gd:feedLink
    href='http://gdata.youtube.com/feeds/api/videos/-0sH3e_qr7c/
       comments'
    countHint='367' />
  </gd:comments>
 </entry>
 ............
</feed>
```

C Extension source code

Listing C.1: XUL Code for creating a tabbed interface

```
<vbox id="sidebarBackground" flex="1">
 <vbox width="200px" heigth="42px" border="1px" id="vboxHeader"
       class="vboxHeader"> [1]
   <hbox class="sidebarHeading" id="header">
     <image id="logo" src="chrome://fmmusic/skin/images/fmmusic.
         gif" width="200px" height="42px" /> [2]
   </hbox>
 </vbox>
 <tabbox flex="1"> [3]
   <tabs>
     <tab label="Now Playing" /> [4]
     <tab label="Artist" />
     <tab label="Video" />
     <tab label="Lyrics" />
     <tab label="Pictures" />
   </tabs>
   <tabpanels  flex="1"> [5]
     <tabpanel class="tab"> [6]
       <vbox flex="1" id="tabBox">
         <groupbox id="nPlayingGroup">
           <grid id="grid"> [7]
             <columns>
               <column id="nPlaying" />
             </columns>
             <rows>
               <row>
                 <label value="Acquiring Data" id="npTrack" />
               </row>
               <row>
                 <hbox height="160px">
                   <hbox class="img-shadow">
                     <imagesrc="chrome://fmmusic/skin/images/
                         acqCover.png" width="160px" height="160
                         px" id="npAlbumCover" />
                   </hbox>
                 </hbox>
               </row>
               <row>
                 <hbox>
                   <description  value="Acquiring Data" id="
                      npAlbum" />
                   <description value="by" />
                   <description value="Acquiring Data" id="
                      npArtist" />
                 </hbox>
```

```
                </row>
              </rows>
            </grid>
          </groupbox>
        </vbox>
      </tabpanel>
      <tabpanel class="tab">
          .......

    </tabpanels>
  </tabbox>
</vbox>
```

Bibliography

[Apac05] The Apache Software Foundation, *The Apache Cocoon Project*, project homepage, http://cocoon.apache.org/2.1/, 2005 - visited 2007-10-03

[Atom08] *The Atom Syndication Format and Publishing Protocol* http://www.intertwingly.net/wiki/pie/FrontPage, visited 2008-01-27.

[Baum01] Robert Baumgartner, Sergio Flesca, Georg Gottlob *Visual Web Information Extraction with Lixto*, Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

[Baum05] Robert Baumgartner, Michal Ceresna1, Gerald Ledermüller, *Deep Web Navigation in Web Data Extraction* International Conference on Intelligent Agents, Web Technologies and Internet Commerce, 2005

[Baum06] Robert Baumgartner, *Methoden und Werkzeuge zur Webdatenextraktion*, DBAI, Institute for Informationssystems, Technical University Vienna, 2006

[Brie06] O'Brien, Damien and Fitzgerald Brian. *Mashups, remixes and copyright law.* Internet Law Bulletion Volume 9, Issue 2, 2006

[Bugh07] Jacques Bughin, James Manyika, *How businesses are using Web 2.0: A McKinsey Global Survey*, http://www.mckinseyquarterly.com, 2007. visited 2008-01-07

[Chau2002] Akmal B. Chaudhri, Rainer Unland, Chabane Djerba, Wolfgang Linder, *XML-based Data Management and Multimedia Engineering – EDBT 2002*, Springer Berlin, 2003

[Chid97] B Chidlovskii, U Borghoff, PY Chevalier, *Towards sophisticated wrapping of web-based information repositories*, Proc. 5th RIAO Conference, 1997

[Cmsp00] C. M. Sperberg-McQueen, Henry Thompson, XML Schema, http://www.w3.org/XML/Schema, 2000, visited 2007-11-21

[Cran05] David Crane, Eric Pascarello, Darren James, *Ajax in Action*, Manning, 2005.

[Croc06] Douglas Crockford, Network Working Group of The Internet Society, http://www.ietf.org/rfc/rfc4627.txt?number=4627, 2006, visited 2007-11-21

[Curb02] Francisco Curbera, Matthew Duftler, Rania Khalaf,William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana, *Unraveling the Web Services Web, An Introduction to SOAP, WSDL, and UDDI*, IEEE Internet Computing, 2002

[Dapp07] Dapper, http://www.dapper.net, visited 2007-09-12

[Dorn05] Andy Dornan, *Mashup Basics: Three for the Money*, http://www.networkcomputing.com/showitem.jhtml?articleID=201804223, 2005, visited 2008-01-10

[Eikv99] Line Eikvil, *Information Extraction from World Wide Web - A Survey*, Norwegian Computing Center, 1999

[ErlT04] *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Prentice Hall, 2004

[Farb06] Dan Farber, *Mashups and the law*, http://blogs.zdnet.com/BTL/?p=2614, visited 2008-01-25

[Fiel00] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*,Doctoral Dissertation, University of California, Irvine, CA, 2000

[Frit03] , Michael Fitzgerald, *Learning XSLT*, O'Reilly, Inc. 1005 Gravenstein Highway North, Sebastopol, CA 95472, November 2003

[Garr05] Jesse James Garrett, *Ajax: A New Approach to Web Applications*, Adaptive Path, 2005

[Gart08] Garnter Inc., *Gartner Identifies the Top 10 Strategic Technologies for 2008*, http://www.gartner.com/it/page.jsp?id=530109, visited 2008-01-25

[Gott07] Gottfried Vossen, Stephan Hagemann, *Unleashing Web 2.0: From Concepts to Creativity*, Academic Press, 2007

[Hamm03] Ben Hammersley, *Content Syndication with RSS*, O'Reilly Media Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2003

[Hamm05] Ben Hammersley, *Developing Feeds with RSS and Atom*, O'Reilly Media Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2005

[Haro01] Elliotte Rusty Harold, W. Scott Means, *XML in a Nutshell - A Desktop Quick Reference*, First Edition, O'Reilly Associates, Inc, 101 Morris Street, Sebastopol, CA 95472, 2001

[Howe07] Jon Howell, Collin Jackson, Helen J. Wang, Xiaofeng Fan, *MashupOS: Operating System Abstractions for Client Mashups*, Proceedings of the 16th international conference on Hot Topics in Operating Systems, 2007

[HsuC02] Chun-Nan Hsu, Hung-Hsuan Huang, Siek Harianto, Elan Hung, Jiann-Jyh Lu *Design and Implementation of WNDL - Web Navigation Description Language*, Institute of Information Science, Academia Sinica, Taiwan, 2002.

[IBMA07a] IBM Alphaworks, *DAMIA*, project homepage, http://services.alphaworks.ibm.com/damia/, 2007, visited 2007-11-23

[IBMA07] IBM Alphaworks, *Mashup Hub*, project homepage, http://services.alphaworks.ibm.com/mashuphub/, 2007, visited 2007-11-23

[IBMA07a] IBM Alphaworks, *QEDWiki*, project homepage, http://services.alphaworks.ibm.com/qedwiki/, 2007, visited 2007-11-01

[IBM08] IBM, *IBM Advances Web 2.0 Platform for Business* http://www-03.ibm.com/press/us/en/pressrelease/23378.wss, Press Release, 2008-01-23, visited 2008-01-25

[InCo04] Integration Consortium, Enterprise Information Integration: A New Definition, http://www.dmreview.com/news/10096691.html, 2004, visited 2007-12-21

[Jack07] Collin Jackson, Helen J. Wang, *Subspace: Secure Cross-Domain Communication for Web Mashups*, Proceedings of the 16th international conference on World Wide Web, 2007

[Jhin06] Anant Jhingran,*Enterprise Information Mashups: Integration Information, Simply*, Very Large Data Bases Conference Keynote, 2006

[Laen02] , Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, Juliana S. Teixeira, *A Brief Survey of Web Data Extraction Tools*, SIGMOD Record, 2002

[Lafo07] Yves Lafon, Web Services Activity Lead, http://www.w3.org/2002/ws/Activity, W3C, 2007. visited 2007-11-10

[LiuX07] Xuanzhe Liu, Yi Hui, Wei Sun, Haiqi Liang, *Towards Service Composition Based on Mashup*, IEEE Congress on Services, 2007

[Kest07] Anne van Kesteren, *The XMLHttpRequest Object.* World Wide Web Consortium, Working Draft WD-XMLHttpRequest-20070618, June 2007.

[Mahe06] Michael Mahemoff, *Ajax Design Patterns*, O'Reilly, Inc. 1005 Gravenstein Highway North, Sebastopol, CA 95472, June 2006.

[McHu07] Josh McHugh, *Should Web Giants Let Startups Use the Information They Have About You?*, http://www.wired.com/print/techbiz/media/magazine/16-01/ff_scraping, visited 2008-01-25

[Mesb06] Ali Mesbah and Arie van Deursen, *An Architectural Style for Ajax*, Delft University of Technology, Software Engineering Research Group, Technical Report Series, Report TUD-SERG-2006-016 2nd revision

[Micr07] Microsoft, *Popfly*, project homepage, http://www.popfly.ms, 2007, visited 2007-12-14

[MozD06] Mozilla Developer Center, *Creating a Firefox sidebar*, http://developer.mozilla.org/en/docs/Creating_a_Firefox_sidebar, 2006

[MozD07] Mozilla Developer Center, *The Joy of XUL*, http://developer.mozilla.org/en/docs/The_Joy_of_XUL, 2007

[Mozi08] Mozilla Developer Center, *Building an Extension*, http://developer.mozilla.org/en/docs/Building_an_Extension, 2008

[Mueh04] Michael zur Muehlen1a, Jeffrey V. Nickersona, Keith D. Swensonba Wesley J. Howe, *Developing Web Services Choreography Standards - The Case of REST vs. SOAP*, Decision Support Systems, Volume 40, Issue 1, 2005

[ORei03] Tim O'Reilly, REST vs. SOAP at Amazon, http://www.oreillynet.com/pub/wlg/3005, 2003, visited 2007-10-25

[ORei05] Tim O'Reilly, *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*, http://www.oreillynet.com/pub/a/general/print_code.html, 2005, visited 2007-01-10

[Open07] openkapow, http://openkapow.com, visited 2007-09-30

[Poor06] Poornachandra Sarang, Ph.D., *Pro Apache XML*, Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013, 2006

[Prot07] Jonathan Protzenko, XUL, *Entwicklung von Rich Clients mit der Mozilla XML User Interface Language*, German edition, Open Soure Press, Munich, 2007

[Proo07] Proto *MASHUPS Understanding Mashup Building Platforms for Business Applications*, http://www.protosw.com/, whitepaper by Proto Software, 2007, visited 2008-01-10

[Rich07] Leonard Richardson, Sam Ruby, *RESTFul Web Services*, O'Reilly Media Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2007

[Shan07] Francis Shanahan, *Amazon.com Mashups*, Wiley Publishing, Inc., 10475 Crosspoint Boulevar, Indianapolis, IN 46256, 2007

[Suz06] Nicolas Zusor, *Transformative Use of Copyright Material*, LLM Thesis, Queensland University of Technology School of Law, 2006

[Wate07] Dan Waters, *Creating An IP Geolocation Popfly Block*, http://blogs.msdn.com/dawate/archive/2007/05/24/creatinganipgeolocation-popflyblock.aspx, MSDN Blogs, 2007, visited 2008-01-11

[Webe08] Volker Weber, *Lotusphere: Lotus detailliert Pläne für Notes und Domino* http://www.heise.de/newsticker/meldung/102336, Heise Newsticker, visited 2008-01-25

[Wess06] Matthias Wessendorf, *Struts*, 2nd edition, W3l Herdecke, 2006

[Wild07] Erik Wilde, *Declarative Web 2.0*, School of Information, UC Berkeley, IEEE Conference on Information Reuse and Integration, 2007

[WuIC05] I-Chen Wu; Jui-Yuan Su; Loon-Been Chen, *A Web data extraction description language and its implementation*, Computer Software and Applications Conference, 2005

[W3CC99] W3C Communications Team, *XML in 10 points*, http://www.w3.org/XML/1999/XML-in-10-points.html.en, revised version from 13 Nov. 2001, visited 2007-11-14

[W3Co06] W3C, *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, http://www.w3.org/TR/REC-xml/, 2006, visited 2007-11-25

[W3Co08] W3C, *Extensible Markup Language (XML)* http://www.w3.org/XML/, 2008, visited 2008-01-07

[W3Co99a] W3C, *XML Path Language (XPath) Version 1.0* - http://www.w3.org/TR/xpath, 1999, visited 2007-12-12

[W3C02] W3C, *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)* http://www.w3.org/TR/xhtml1/, 2002, visited 2007-11-25

[Yaho07] , Yahoo!, *Pipes*, project homepage, http://pipes.yahoo.com, visited 03.12.2007

[YouT07] YouTube Data API documentation, http://code.google.com/apis/youtube/developers_guide_protocol.html, visited 2007-10-10

[Zhan04] Lili Zhang, *RESTful Web Services*, Department of Computer Science, University of Helsinki, 2004