UNIVERSITY OF ECONOMICS, PRAGUE

Faculty of Informatics and Statistics

Department of Information Technologies (KIT)

Trends in web application development

Bachelor's Thesis

Vojtěch Jasný

Thesis Supervisor: Ing. Tomáš Brabec

January 2007

Abstrakt

Bakalářská práce se věnuje dvěma tématům: prvním z nich jsou trendy ve vývoji webových aplikací z pohledu klienta. Jsou popsány techniky AJAX a COMET, formát pro výměnu dat JSON, nástroje použitelné pro ladění aplikací v Javascriptu a představeny tři Javascriptové knihovny. Důležitou součástí vývoje na klientovi jsou formuláře, které mají v dnešní podobě daleko ideálu. Je proto obsažen popis technologií XForms a Web Forms 2.0 a jejich porovnání. Druhým tématem je vývoj aplikací na straně serveru. Nejprve jsou hodnoceny stávající najčastěji používané technologie. Dále jsou představeny čtyři frameworky, které jsou na závěr porovnávány dle sady kritérií a je diskutována jejich vhodnost pro různé druhy nasazení. Zastoupeny jsou jazyky Python, Ruby a Java.

Abstract

The thesis is divided between two main topics: the first are trends in web application development on the client-side. Described are AJAX and COMET techniques, JSON format for data interchange, tools usable to debug Javascript applications and three Javascript libraries. Important part of client-side development are forms, which are far from ideal in their current state. Hence description of XForms and Web Forms 2.0 and their comparison is included. The second topic is server-side development. First the most used technologies used today are evaluated. Further, several frameworks are described and compared on a set of criteria. Python, Ruby and Java languages are represented.

Acknowledgments

I would like to thank Christian Heilmann from Yahoo! UK Ltd., who kindly provided me with unedited chapter on Javascript libraries from his upcoming book.

I would also like to thank my supervisor for his useful comments and suggestions.

Declaration - Prohlášení

Table of Contents

Chapter 1

Introduction

Advances in web application development are very rapid, however certain milestones however can be identified. Currently as developers, find ourselves on the verge of one such milestone - the onset of technologies and techniques sometimes imprecisely labeled as Web 2.0 is reshaping the ways web applications look and feel. Along with changes in the presentation layer, changes in the way business logic is structured and implemented also take place.

In this bachelor's thesis I will seek to describe some of the trends that have already found their way to the world of web application development and some which are yet to be explored by the web development community too. I divide my attention equally between client and server development. Accordingly, this thesis has two main chapters: first chapter gives overview of techniques used in client-side programming and the second chapter is focused on server-side development frameworks.

Client-side programming or scripting is generally defined as a group of programs run one the side of the client (which in web programming is the browser). They are usually written in *Javascript*, less often in *VBScript* and they utilise the Document Object Model (*DOM*) to make changes to the rendered document. When and how scripts are executed is defined in the W3C HTML standard (events and respective event attributes). Server-side programming deals with generation of markup document on the server, querying databases (or other data sources) in the process and sending the output back through the network using HTTP protocol.

At the side of the client, the most notable trend in web application development is the use of asynchronous HTTP requests for XML or *JSON* message exchange with the web server, without the necessity of reloading the whole page for updates. This technique can be implemented either from scratch or a wrapper library can be used to hide differences between browsers from the programmer and to provide more comfort during development. The first part of the client-side development chapter describes features of three such libraries - prototype, Dojo and Yahoo! User Interface. The second part pertains to markup languages, specifically XForms and Web Forms 2.0. Forms are a critical component in web applications and their present capabilities are lackluster. In the future one of these standards is expected to replace them.

In the server-side chapter I briefly comment on the pros and cons of the three technologies prevalent in server-side development today - *PHP*, ASP.NET and J2EE. I then proceed with introduction of four frameworks which are now gaining in popularity. First is the Google Web Toolkit (*GWT*), a library that introduces an interesting development model based on translating Java code to Javascript. Two dynamic languages, Python and Ruby, are also represented. From the land the snake I introduce two frameworks - Turbogears and Django. In Ruby, the prevalent framework is Ruby on Rails. Although the four are very different products, they have one thing in common - the aspiration to make building complex web publishing and content management systems as simple and fast as possible. At the end of the chapter I attempt to compare the frameworks and discuss.

All of the described products are evaluated from the perspective of medium-size website development (perhaps an e-commerce site or a corporate website with complex web publishing system). Building large sites with millions of unique visitors a month poses completely different challenges both in terms technology choices and performance requirements and is beyond the scope of this thesis.

Chapter 2

Client-side development

2.1. Browser programming

One of the troubles of client-side web programming is long delay before new technologies move from drafts to real world adoption. The reason for this lies in the result of the browser war during the late 90s. At that time Microsoft acquired an overwhelming majority of the web-browser market, peaking at 96%. Since then, with advent of alternative web-browsing programs, Microsoft is slowly losing ground to them, but even most optimistic estimates don't speak of more than 20% alternative browser market share. Once Microsoft won the browser war, it virtually ceased to develop its then superior Internet Explorer. From the economic point of view, this made a lot of sense, but for web application developers it was a major setback.

It was probably during the last three or maybe four years that the table less web design started to become prevalent. It became very soon clear that the so-called alternative browsers often had a much better support for the necessary web standards (CSS2, HTML 4.01, later XHTML 1.0). Technical superiority of Firefox over Internet Explorer was quite evident. Moreover, some innovative new technologies which the web application developers were keen to use (like the canvas tag, for instance) were at first only introduced and implemented in alternative browsers or even not implemented by IE at all.

Eventually, the slow decline of IE use led Microsoft to update their browser. It was first announced on February 15, 2005 by Bill Gates, but a final version was not available until October 18, 2006. Large part of Internet Explorer's rendering engine has been overhauled and a number of improvements to CSS, DOM, and HTML support been made. IE7 is distributed as a priority update through the Windows Update service, therefore it is replaces older versions of IE fast.

Thanks to the above events it becomes feasible to employ innovative new techniques in web applications. This is most noticeable in the way applications look and feel to the users. Some of the best known web sites, which take advantage of the emerging technologies, include Google Maps or Google's web mail service called GMail. Good portion of the new methods is under the hood, hidden away from the eyes of an average user. That however does not make them any less important or interesting from the developer's point of view.

This chapter of my thesis focuses on client-side application development trends. I discuss AJAX and COMET, which are both technologies used for asynchronous communication with the web server, and Javascript libraries that make developer's life easier. In the second part I discuss XForms and Web Forms 2.0 specifications, their future and compare them.

2.1.1. Browsers in the market

After the development of Netscape Navigator was stopped its codebase was open-sourced and after some years of developer work Mozilla Suite was the result. With exception of corporate environments it was later replaced by a less resource-hungry colleague called Firefox. At the same time Opera, a small web browser and mail suite was being developed by a rather small company based in Oslo, Norway. It is a proprietary closed-source program and for long time it was only available for a fee or as an ad ware. This has changed in September 2005 when Opera company announced availability of its browser for free. Opera has also announced its partnership with the Nintendo company which will quite possibly help it improve its market position a great deal in the following years, because the market share of Nintendo's handhelds and its recently launched Wii console is anything but small. Opera is also very successful with its mobile version Opera mini for Java enabled cell phones.

Another important browser in the market is Konqueror. It was initially developed as a browser for KDE (K Desktop Environment) on Linux and in the beginning its rendering qualities were no match for major browsers. This has radically changed in 2002 when the code base of Konqueror's KHTML rendering engine was adopted by Apple as a basis for the new Safari browser. After a year of development Apple has released the code for revamped KHTML engine under the name of WebKit as it was required to do by the *LGPL*. It turned out relatively difficult to port all of the changes made by Apple back to KHTML - in part because it took a year of development at Apple before the modified code was first released to the public and in part because the changes were not too well documented. Apple later set up a CVS repository where it publishes all the changes and some of their contributions were successfully back-ported since. One of the notable patches made KHTML Acid2 compliant. KHTML rendering engine is also generally recognized as being faster then the Gecko engine used in Firefox. To date, no production quality browser exists for the Windows operating system.

2.1.2. AJAX & COMET

The need to reload a page on every user's action has long been felt by the web developers as a pain of the web application paradigm and they strove to overcome this issue. In the beginnings however, their effort was not much supported by the web browser vendors. Therefore, certain hacks had to be invented to make it possible. Either a script generated blank image was used and its URL was modified using Javascript to execute server-side code or an IFRAME tag hidden from the user by *CSS* was used. Other methods included using Microsoft ActiveX objects, Flash movies or Java Applets. Obviously, kludgy methods like that were not very easy for the programmers to use or debug. Hence some felt that a support for this kind of behaviour in the browser was called for. Figure shows principle of traditional approach to web applications with page refresh per every action of the user.

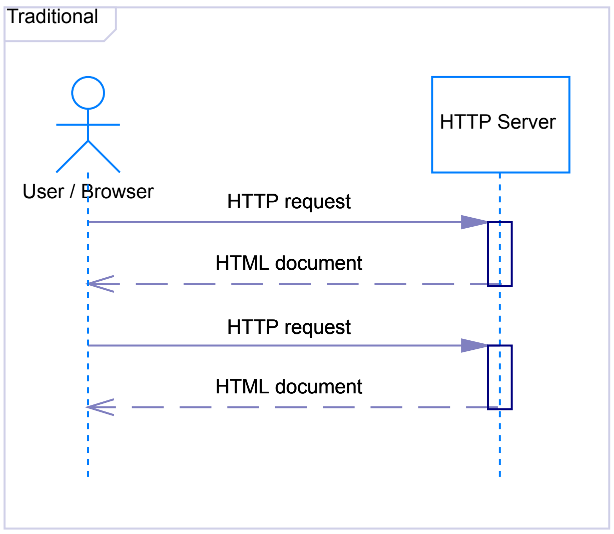

Figure 2.1. Sequence diagram - traditional approach to web application

As an answer to these wishes the XMLHttpRequest (XHR) was developed by Microsoft as part of Outlook 2000 web interface. The first Internet Explorer version that shipped with it was version 5.0 and it was implemented as an ActiveX object, requiring special syntax to initialize and use in Javascript. In 2000 Mozilla 1.0 followed and released a compatible native implementation. Soon it was also included in Opera 8.0, Konqueror and Safari. Based upon existing implementations, W3C released a working draft specification for the XMLHttpRequest object's API on 5 April 2006 [2]. In February 2005 Jesse James Garrett first used the term AJAX, an abbreviation for Asynchronous XML and Javascript. The technologies he called AJAX have long existed before, but he needed a shorthand term for a proposal to his client, it should be noted that AJAX itself is not a technology but a name for a group of technologies.

2.1.2.1. Asynchronous model

As opposed to traditional web pages which are reloaded and redisplayed as a whole every time the user clicks a link or requests data refresh, AJAX-enabled pages use asynchronous approach. Figure illustrates how asynchronous approach is used in a web application:

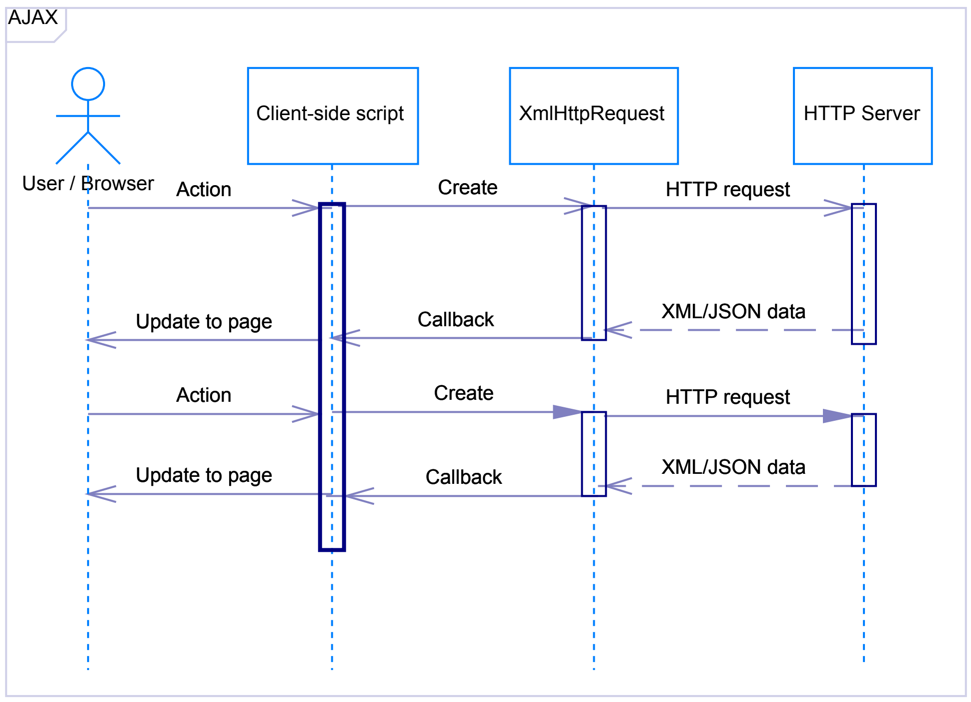

Figure 2.2. Sequence diagram - asynchronous approach to web application

On the action of the user, an `XmlHttpRequest` - object responsible for making HTTP requests to the web server - is instantiated. `XmlHttpRequest` has several attributes that can be set before HTTP request is sent: the URL (usually limited to the same server as the currently viewed page for security reasons), HTTP request method (usually GET or POST methods are used), user name and password to use to access the URL and an arbitrary number of custom HTTP headers. Some of the often set headers include Content-Type and Accept-Charset. Last but not least, an event handler (callback function) needs to be set so that the response from the server can be processed. When all parameters of the object instance have been set, it can be sent using the `send()` method. An optional argument to the method is the request body. This is useful for transmitting larger data using the POST method.

The request progresses through several phases as indicated by its *readyState* property: uninitialized, loading, loaded, interactive and finally complete. In our client-side script we can utilize *readyState* to implement graphical activity indicators (sometimes called throbbers) informing the user that there is browser activity going on - for each of the phase changes the callback function is called. When *readyState* is complete, it still does not mean that our request went through successfully - a well written check the *status* property which is the HTTP status code of the response and only proceed to process response data when it is 200 (HTTP code for OK) or other meaningful value [6].

Response data of the successful request are available either as an XML DOM tree (given correct text/xml content-type is sent) or as plain text (*responseText* and *responseXml* members respectively). After necessary parsing or decoding they can then be used to inform the user about success or failure of his action or to update the page with new information. Optionally, a whole portion of the page can be replaced by *responseText* using *innerHTML* property which is supported by most modern browsers. This is often preferable to parsing complicated XML/JSON messages.

2.1.2.2. COMET

COMET is a technique based on the same principle as AJAX using the same technology but the other way around. Using traditional web techniques it is not possible for the server (or rather the application running on that server) to contact the browser once the document has been sent - i.e. current web is a pull technology not push. With COMET it is possible to overcome this hurdle. Sequence diagram stays the same as figure 2.2, but mechanism by which the open connections are used is different. Typically after the page is loaded one or more XHRs are made. The server these connections and keeps them open until an important event takes place or it has something to send to the client.

This way a very fast application reaction time can be achieved, because there would be significant (time, data, etc.) overhead associated with opening many HTTP connections (as would be needed if the client asked for changes periodically). Unfortunately, this means also some additional strain on the server because keeping many connections open consumes memory and processing resources. Software is being developed to deal with this issue such as Comet and Ajax Request Router[1] and others.

Use of COMET is advantageous especially in applications which require a lot of interactivity, such as games, online chats, collaborative drawing applications and other.

2.1.2.3. Development and debugging

Process of developing and debugging Javascript source code is a lot more complicated compared to debugging server-side. This is in part because Javascript is a dynamic language. At this point in time, Javascript IDEs are no match for Java or Python IDEs. Use of asynchronous connections only adds to the problem. Common IDEs can usually offer programmers functions such as syntax highlighting, brace matching, basic code outline and syntax checking, but not much more. Furthermore, natural running environment for Javascript code is the web browser, therefore an ideal tool has to have a level of connection with it. This is best fulfilled by an add-on or plug-in to browser, which is how current tools are designed. Among the best is the fireBug extension[2] for Firefox. In its 1.0 version it features the following:

- Javascript command line, which can be used to evaluate complex Javascript expression in the context of current document
- console window which you can log messages to
- Javascript debugger complete with profiler
- DOM inspector
- network activity monitor showing incoming and outgoing `XmlHttpRequest`s along with their contents
- many other Javascript unrelated, but immensely useful tools, mostly for CSS debugging

The extension is actively developed and soon after initial release became almost indispensable for anyone developing applications in Javascript. FireBug has a lite version which enables some of the basic functions in browsers other than Firefox.

Another useful tool for Javascript development in Firefox is the Venkman JavaScript Debugger, which dates back to April 2001. It has all functionality expected from a debugger: watches, breakpoints and a profiler. It is also possible to further extend it by Javascript code. Venkman is probably more mature and stable than the Javascript de-

[1]<http://rphd.sourceforge.net/>
[2]<http://www.getfirebug.com/>

bugger included in fireBug, but it has less functionality and not much following in the web developer community.

Another interesting approach that I discuss in the Server-side chapter of my thesis is that of Google Web Toolkit, where all code is written in Java and then cross-compiled to Javascript. That way it is possible to utilize modern feature-rich IDEs for Java and still get Javascript code compatible with all modern browsers.

2.1.2.4. Pros & cons

Obvious advantage of the asynchronous approach to page update is that user is not forced to wait for a whole potentially complex page to reload. Resulting sites tend to be easier to use because their interface resembles traditional desktop applications.

The downside is that use of AJAX is not yet so common and less computer savvy users may have trouble understanding how the application works. What doesn't help is that using XHR calls breaks the function of back/forward buttons. When an asynchronous call is sent and the page is updated the browser history doesn't change because the URL stays the same. Current kludgy solutions are forced to exploit fragment part of the URL to overcome this. Another downside from the developers point are differences between browsers. Using a Javascript library is a de facto necessity.

2.1.3. JSON

Although XML is a fine format for structured data it is not always suitable for uses where significant number of requests is made. The overhead of XML parsing and processing can have highly negative impact on overall performance. In this case JSON may be the format to use in an AJAX application instead of XML. Another situation where it can come in handy is when data messages are relatively simple in terms of size and structure.

JSON is a subset of Javascript literal object notation syntax that can be used as means of data exchange between object-oriented programming languages. It supports 6 data types native to Javascript: number (integer or floating point), string, boolean, array (sequence of items in square brackets delimited by comma), object (also serving as hash table data structure in Javascript) and the null value. The following example shows possible JSON message about a student, carrying one parameter for each of allowed data types.

Example 2.1. An example JSON message

```
{
    "name": "Vojtech Jasny",                        // string
    "age": 22,                                      // integer
    "member": false,                                // boolean
    "studiedYears": 3.5,                            // float
    "emails": ["xjasv03@vse.cz, "voy@voy.cz"]   // array
    "studyPlace":                               // object (hash table)
    {
        "faculty": "Faculty of Informatics and Statictics",
        "departement": "Departement of Information Technology"
    }
}
```

2.1.4. Javascript libraries

Due to differences between browser implementations and general verbosity of Javascript and DOM constructs, writing client-side Javascript applications in a portable and manageable fashion is a pain. For this purpose, a number of Javascript helper libraries was developed. They are usually sets of functions or extensions to the Javascript built-in objects. Extending Javascript built-ins is trivial, because it is a prototype-based language. It is also a very good way to provide backward compatibility for older browsers.

We can classify Javascript libraries as primary and secondary. Primary libraries provide basic functionality, while secondary, more specialized libraries, build on top of them. An example of a primary library is prototype, an example of secondary library is script.aculo.us, which utilised prototype to create a set of rich widgets and animations. Some libraries, such as Yahoo! User Interface have properties of both types. Having such classification in mind, because it allows for modular approach when choosing library for use in a web application. In some cases relatively simple primary library is a good choice, instead of a behemoth with 90% functions we will never use in our particular application. Unfortunately, libraries more often than they don't incorporate elements of both primary and secondary libraries.

To summarize, main functions of Javascript libraries are usually following:

- they make differences between DOM implementations and their various quirks transparent to the programmer - ideally this leads to more readable code which is also easier to maintain at the same time

- provide wrappers around `XmlHttpRequest` machinery and reduce number of lines of code needed common tasks along with associated error checks
- provide a set of common controls (components, widgets) such as auto completing fields, editable table listings, sliders, date pickers, etc.
- offer API for frequently used effects like animations, fading and morphing of DOM objects

As of December 2006, there are probably more than 150 Javascript libraries and frameworks in existence[3], but only a handful implement the needed functionality and also are well designed. For my thesis I have chosen to acquaint you with three of very popular ones: Prototype in conjunction with script.aculo.us, which is one of the most widely spread libraries, Dojo and Yahoo! User Interface library.

2.1.4.1. prototype & script.aculo.us

One of the most frequently used Javascript libraries is prototype. As has been said before, it is an example of a primary library. Prototype has not been named by chance - Javascript is a prototype-based language. In prototype-based languages, classes are not present - instead, inheritance is achieved by cloning existing objects, which serve as prototypes. This has a very beneficial side effect for web developers - implementations of document object model tend to be vary across different browsers and by exploiting the prototype model cleverly you can compensate this to a degree. By assigning new methods to prototypes of standard objects it is possible to add new functions to them or modify behaviour of default ones. Prototype adds many new methods to standard objects and also brings some new objects of its own such as the `AJAX` object, which is a wrapper for asynchronous HTTP requests. This approach has also some drawbacks in that prototype may have trouble co-existing with another framework in one application. It is fair to say however that this problem is not specific to prototype. Results are usually unpredictable when you start using more than one library at the same time.

One of the most useful features is the `Enumerable` module (or technically a *mixin*) which has been inspired by Ruby language. It is a general means of traversing collections without regard for number of their elements. Collections can be not only traversed, but also filtered, tested on conditions etc. Using `Enumerable` in your Javascript code can reduce total amount of boiler-plate and increase readability of code. In the following

[3]Estimated by list of frameworks at <http://ajaxpatterns.org/Ajax_Frameworks>

example we try to find the number of hyper links which point to Czech domain names (cz) first without `Enumerable`:

```
 1 var links = document.getElementsByTagName('A');
 2 var local = []
 3
 4 for(var i = 0; i < links.length; i++) {
 5     if(links[i].href.match(/\.cz/i)) {
 6         local.push(links[i]);
 7     }
 8 }
 9
10 alert(local.length);
```

First, we assign all of the hyper links in the current document in the *links* array (line 1) and create an empty array (*local* - line 2) which we will use to store links to Czech top-level domain. Then we traverse *links* in a for-loop using a temporary integer variable and size of the *links* array. We match the individual elements of the array against a regular expression and append them to the *local* array. At the end of the script we display the number of links found in a pop up window. Let's now compare to similar code in prototype:

```
var local = $$('A').findAll(function(link) {
    return link.href.match(/\.cz/i)
});

alert(local.length);
```

On the first sight, the prototype version may look a little intimidating and cryptic, but that really isn't the case. Instead of calling the `getElementsByTagName` method of the current document, we have simply used the *$$*[4] convenience function of prototype, which selects elements from the current document by CSS selector. The result of this function can be enumerated using prototype `Enumerable` module. Now, if we wanted just to run some operations on the array of links, we would call the `each` method - in this case however, it is much easier to use the `findAll` method. All of the enumerators in prototype take a single argument, which is a function that get's called for every element and that element is passed to it parameter (mostly anonymous functions are used for this). The `findAll` method returns those elements, for which the enumerator function

[4] `$` is actually a legal character for function names in Javascript.

has returned true. It is then a simple exercise to return the boolean result of a regular expression match and we have our list of links to the Czech domains. Clearly, the code in prototype is not only shorter by a couple lines but also a lot more expressive (after some getting used to). In terms of speed prototype code should perform roughly on par with standard Javascript loop in the first example.

Because prototype provides the firm ground of basic functionality, it was used by many authors to build more advanced functionality upon, authors of the script.aculo.us library among them. Currently in version 1.7, it makes it easy to build sites with simple AJAX controls (auto-complete, drop down lists etc.), animation effects and drag and drop interface. Script.aculo.us also contains a set of classes to facilitate test-driven development.

2.1.4.2. Yahoo! User Interface library

Note

This chapter uses information from the upcoming book by Christian Heilmann from Yahoo! UK Ltd. on Javascript libraries which has not been published yet. Therefore, it is not mentioned in the bibliography.

Yahoo! UI library a.k.a. YUI is a set of utilities and widgets which is developed at Yahoo! and available under BSD license. It was originally started because different teams struggled with similar problems and issues (browser incompatibilities and such) and a lot of resources was wasted. All developers at Yahoo! now use YUI and report problems to the YUI team. YUI is relatively sizable, but modularized well, so that developers can choose to use only the parts of YUI they need:

- Animation Utility
- Browser History Manager - currently experimental, helps retain the functionality of back/forward buttons on scripted pages
- Connection Manager - AJAX capabilities
- DataSource Utility - provides a common configurable interface for other components to interact with different types of data, from simple JavaScript arrays to online servers over XHR
- DOM Collection Element Utility - element retrieval, altering, adding and deleting
- DragDrop Utility - functions which allow creation of drag and drop interfaces

- Event Utility
- YAHOO global object - wraps everything into a common namespace and provides some common functions

On top of utilities Yahoo! engineers have built a set of widgets (called Yahoo! library controls), which can be easily included in HTML documents:

- AutoComplete
- Button Control
- Calendar
- Container (including Module, Overlay, Panel, Tooltip, Dialog, SimpleDialog)
- DataTable
- Logger
- Menu
- Slider
- TabView
- TreeView

To demonstrate use of a controls, I will show how YUI can be used to create a calendar (the example used is taken from [20]). First, we need to create an element to which the calendar will be rendered:

```
<div id="cal1Container"></div>
```

Consequently, we need to initialize the component:

```
<script type="text/javascript">
  var cal1;
  function init() {
    cal1 = new YAHOO.widget.Calendar("cal1", "cal1Container");
    cal1.render();
  }
```

```
7   YAHOO.util.Event.addListener(window, "load", init);
8 </script>
```

This is done by instantiating the `Calendar` object (line 4) and calling the `render` method on it (line 5). Initialization is called after page load, which also showcases the Event utility (line 7). The result is a fully functional calendar widget in place of the *cal1Container* div element. In real application we might like to initialise it to a default value (if for instance editing a row in table) which would be trivial using the `select` method. Many parameters of the Calendar can be set by the third parameter to the constructor call (config object), which I omit here for simplicity. Among them are the minimum and maximum dates allowed for selection, localized day/month names and other. Another layer of customizability are members of the Calendar object. You can define events handlers to be called when date is selected, before calendar is closed etc. or set CSS classes for different parts of the calendar markup.

Very interesting is the DataTable component (in beta as of writing), which represents a table of records obtained using XHR from text, XML or JSON data. It supports client side pagination, sorting and much more. As opposed to calendar, which a developer proficient in Javascript could probably implement in a couple days, this widget is much more sophisticated and it would take a lot more developer time to re implement it (additional time would be required to make it cross-browser). Components are probably the greatest benefit of YUI and using them can save significant amounts of time and resources. Coming from corporate background, YUI is thoroughly documented.

2.1.4.3. Dojo

I am concluding the description of Javascript libraries by Dojo, because it is a little different. It is designed to run not only in web browsers but also in any other environment which supports Javascript sanely - in other words should be interpreter independent and could be used for instance by ActionScript users (implementation of Javascript in Adobe Flash).

Dojo's functionality is divided into name spaces which cover particular areas and can have dependencies defined between them. Range of functionality Dojo offers is wide due to its orientation on multiple execution environments. Although a good portion of it is targeted on the HTML platform, some functionality (command line unit testing support for instance) may find better use in other environments. I provide only a basic list of Dojo's functionality:

- form validation

- form widgets
- layout facilities (content panes, split container, etc.)
- AJAX features
- object-orientation utilities (provides a layer on top of OOP in Javascript which resembles more traditional object-oriented languages like Java)
- debugging facilities
- unit testing package
- offline data storage (cookies, but also other means depending on the target platform)

Dojo has a relatively extensive UI widget library which aside from relative simplicities such as a number spinner and an auto completing text field features more unusual ones like as a color picker or a rich text editor. You can organize various widgets using different layouts, which can help you abstract the user interface from the target platform. For the web the widgets are packaged complete with HTML code and style-sheets, but they don't stand compared to the ones packed with YUI in terms of look and feel.

2.1.4.4. Comparison

For most simple applications prototype is the best choice. Along with script.aculo.us it reasonably simplifies daily Javascript tasks, offers a library of visual effects and even some simple widgets. An advantage of prototype is that it is sort of a standard and is a lot of open source code available on the web, which builds on top of it. However, because of the way it extends standard objects prototype can have trouble co-existing with code not taylor-made to it.

For large applications where complex user interface is needed the library of choice is definitely YUI. It has a large collection of widgets which covers most of what you may ever need. It is backed by a strong company which can devote resources to making its widgets cross-browser compatible and overall the library is very well designed and documented.

If you happen to be developing code intended for sharing between different Javascript environments, the way to go for you may be Dojo. For use on public sites its widgets are somewhat unpolished, although with some effort they could be customized.

2.2. Markup languages

2.2.1. XForms

Forms of various types and their validation are ubiquitous in almost all web applications. Support for them in HTML and XHTML 1.0 specifications however, is far from ideal. Various libraries and techniques are needed to work around their technical limitations: taglibs in Java EE or controls in ASP.NET and their validators, to name a few. Common purpose of these technologies is to offer simple interface to help use complex form components, while generating rather primitive HTML code at the output. This is a satisfactory solution for the real world, but not preferable in terms of elegance and portability. Some languages (for example PHP) even don't provide any infrastructure for complex forms design and validation at all, or not without the use of additional frameworks.

XForms are the next-generation forms and should be the solution for this problem. It is an XML based format which is meant to be embedded in other documents, such as XHTML 1.0 or XHTML 2.0 (of which it will be part of). It is primarily intended as a replacement for traditional HTML forms, but it can be easily added directly into other XML documents - SVG, WML or DocBook for instance.

One of the main characteristics of XForms is separation of data from presentation. This means that individual form items and constraints associated with them are defined separately from the structure of the form. Furthermore, XML, which is a more natural environment for repeating data types like lists or arrays than plain query string, is used to transfer data from the client to the server.

Of course, XForms are not just reformulation of old HTML forms, although they can be almost as simple as they are, if needed. They bring to the table many features that could have been traditionally achieved only with use of many lines of JavaScript (or AJAX) code. Some of them can be done with much less effort in XForms declaratively. In the following part of the thesis you will be briefly introduced to some of the new interesting concepts of XForms.

2.2.1.1. Basic form in XForms

To teach reader XForms is by no means an aim of my thesis. There are much better materials for detailed study available online such as [9]- I will limit actual code examples to bare minimum needed for illustration. I expect basic knowledge of XPath and XML Schema, although with some common sense you can figure out the examples without it. The following first example shows simplified form consisting of two text fields and one password field. I am abstracting from the document (possibly HTML) which would probably enclose this code in real application:

```
<xforms xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <model>
    <instance>
      <student>
        <name     xsi:type="xsd:string" />
        <born     xsi:type="xsd:date" />
        <password xsi:type="xsd:string" />
      </student>
    </instance>
    <bind nodeset="/student/name"    id="name" />
    <bind nodeset="/person/born"     id="born" />
    <bind nodeset="/person/password" id="password" />
    <submission id="form1" action="submit.html" method="get" />
  </model>
  <input  bind="name"><label>Student Name</label></input>
  <input  bind="born"><label>Date of Birth</label></input>
  <secret bind="password"><label>Password</label></input>
  <submit submission="form1"><label>Submit</label></submit>
</xforms>
```

You can see that the XForms document is divided in two parts. The first part enclosed in `<instance>` tag contains the definition of the data model along with data types of its individual members. This is called an instance and it is possible to store it in separate files and reference to them using src attribute of the instance tag. In our fictitious form, the instance represents a student, for whom we track name, date of birth and a password.

Name and password are both strings and date of birth is not surprisingly of the date type. Data types used come from the XML Schema specification and for this purpose we have defined xsd and xsi XML name spaces for XML Schema and XML Schema instance respectively. An intelligent user agent would inspect the data types of the fields and could offer aids for the user to enter data, such date pickers where appropriate, etc. The next part of the model are three occurrences of the bind element. The bind element can be used to map data members of our student to the actual form fields below using id's and XPath expressions. This comes in handy when we want to use one instance definition for multiple form layouts.

Below the model definition, there is the second part of the form, which defines the form layout. In a real web application, we would style this using CSS. Our sample form has four user interface controls. Text fields for name and date of birth and a password field for password. We also have a submit button, submission attribute of which references the id of the submission element defined back in the model.

These three types of controls are just a subset of what XForms offer - all other controls known from HTML forms are also available, some with a twist. For example, for the file upload control, XForms define a number of possible input sources [9]:

- file upload - pick a file using a system dialog, standard behaviour of the file upload control in current HTML
- scribble - applicable to images, user can draw an image using a hardware device
- acquire image - also applicable to images, scanner or a digital camera can be used to acquire the image data
- record audio or video

It will be interesting to see what new web applications making use of those features will crop up - I can easily imagine applications working with various card readers, social networking applications that would allow users to easily record their video or audio profile or draw pictures collaboratively. From the current web's point of view, the possibilities are endless.

Another control that wasn't present in the HTML forms at all is the range control, which provides an easy way to enter a numerical value from an interval.

This form is very simple, but we can already see multiple improvements over the HTML forms. Data model definition is separated from the form layout and fields have data type definitions associated to them. If, for example, the user enters string in a date of birth field, he will be warned when he attempts to submit the form. Data types are derived from XML Schema and include date, time, number of various sizes, boolean or whitespace separated lists.

Some could object that in comparison to HTML forms, XForms are much more verbose - a lot more code is needed to display what is basically just three fields and a button. Designers of XForms have thought of this and for simple use-cases you can leave out most of the model. The only thing you cannot drop is the submission element.

When the form is submitted, data are received by the server in the same format as we have defined in the model. Example follows:

```
<instance>
  <student>
    <name>Vojtěch Jasný</name>
    <born>12.03.1984</born>
    <password>mvDHTQxJ</password>
  </student>
</instance>
```

Advantages of XML over query string are quite obvious - for instance, it can be transformed using XSLT or queried with XPath. Additionally, to prefill a form with some values, all we would need to do would be to add our desired values to the model just in the example above.

2.2.1.2. XForms properties

So far I have shown how you can validate data in XForms against a set of basic data types. This is an improvement, but XForms go further with XForms properties. Properties are used to define restrictions that can affect the behaviour of XForms controls. To connect properties with data members, the bind element is used. We have already used this element in previous section to bind individual nodes of the model to the form fields.

There are multiple properties that can be used as attributes to the bind element, but the following will probably be used most often:

- calculate - used to define calculation to be performed on the item
- constraint - defines a constraint (validation) for the item
- readonly - sets the item to be read-only
- required - sets the item as required
- type - this is just another place type can be associated to the item aside from the model, as we have used it in previous examples

The only question that remains is how can we express the constraints and calculations. In traditional forms, validation on the client-side was usually written in Javascript. In XForms, this is not the case. Instead, - constraints are expressed declaratively using XPath expressions. Following simplistic version of the student model set's the student's name as required:

```
<model>
 <instance>
   <student>
       <name xsi:type="xsd:string">
   </student>
 </instance>
```

```
7   <bind nodeset="/student/name" required="true()"/>
8 </model>
```

Note the line 7 where the student name node is selected using XPath and the required attribute is assigned a value of XPath `true()` function. Of course, this is only a most basic example of what can be achieved using properties, for more information you can refer to [9] or many other XForms articles and tutorials on the web.

2.2.1.3. Current state of XForms

As great as all the features of XForms sound, their native implementation in current browsers is unfortunately far from complete. That doesn't mean they cannot be used at all, there are four possible solutions:

- Plug-in or an extension for your browser - plug-ins are available for both IE and Firefox. This solution is ideal for learning or for environments where you want to make use of XForms and have complete control over the workstations that use the application.
- Javascript implementation of the XForms standard. This way you can send XHTML and XForms to the browser and the library will take care of translation from XForms to HTML elements.
- Server-side compiler that can translate the form into HTML and Javascript code.
- Pure client-side application which doesn't live in the browser. Can be used for form design and debugging, but won't help you much with deployment of your application.

In essence, you will either have to have access to your client's computers or sacrifice some elegance (possibly also speed) and use the translation to good old HTML.

Currently under way are also works on XForms 1.1 working draft, which refines the XML processing platform introduced by XForms 1.0 by adding several new submission capabilities, action handlers, utility functions, user interface improvements, and helpful data types as well as a more powerful action processing facility, including conditional, iterated and background execution, the ability to manipulate data arbitrarily and to access event context information [17].

2.2.2. Web Forms 2.0

After a W3C workshop in 2004 Apple, Mozilla and Opera were becoming increasingly concerned about the direction web technology development was heading. W3C was mostly focusing on developing complex over engineered standards such as XHTML 2.0 and was mostly ignoring real world needs of web authors. They formed a new working group called Web Hypertext Application Technology Working Group (WHATWG) which intends to maintain and develop future versions of HTML with HTML 4.01 as its starting point. One of the specifications that WHATWG works on are Web Forms 2.0, which are currently in their working draft.

Unlike XForms, Web Forms have no intention to revolutionize web in one fell swoop. Instead, WHATWG chooses the path of smaller steps which will be easier to implement and adopt by the community. This doesn't mean that work on XForms will go down the sink. Web Forms aim to be a bridge between business logic and data described by XForms. They will simplify the task of transforming XForms into documents that can be rendered by browser without support for XForms [18]. This concept is illustrated by figure:

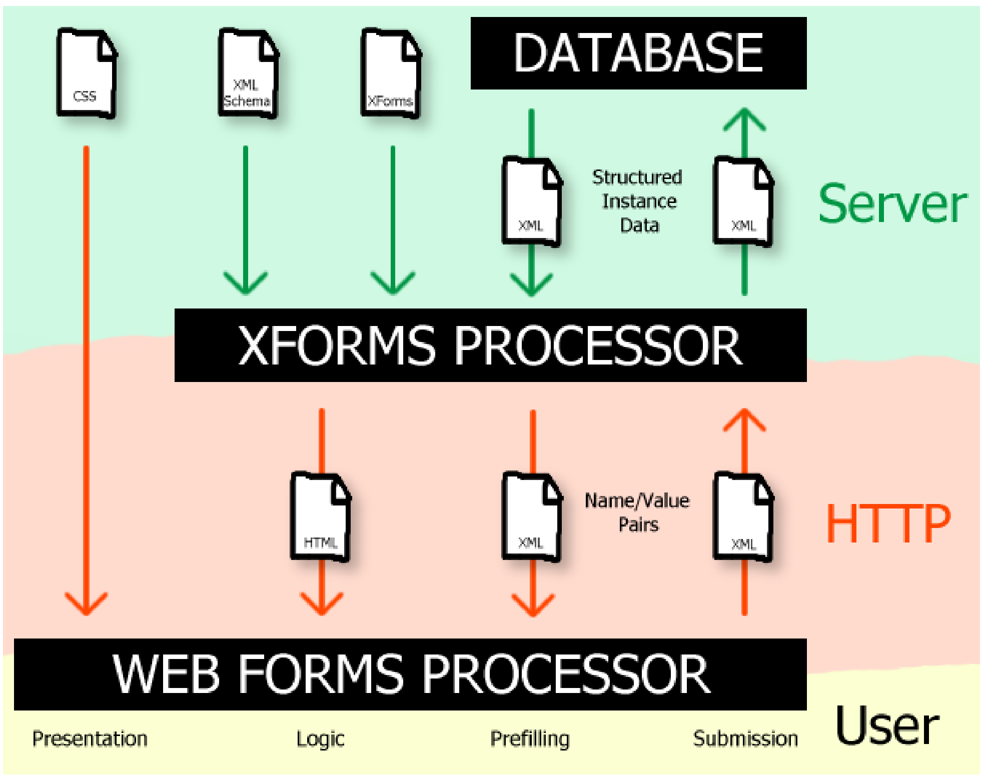

Figure 2.3. Relation between the Web Forms and XForms standards [18]

2.2.2.1. Form controls

One of the most frequent complaints against HTML forms in their current state is that they don't offer enough input controls for most situations. Web Forms solve this by proposing several new input types: datetime, datetime-local, date, month, week, time, number, range, email and url. User agents are recommended to show specialized widgets, instead of requiring that the user enter the data into a text field [18]. Where appropriate, Web Forms introduce the pattern attribute, which checks value of the input against a regular expression. Another new element datalist can be used to provide a list of values to select from in an input. Using a data attribute, values can be fetched from and external file.

File upload fields can specify a minimum and a maximum number of files that must be attached for the control to be valid.

Forms elements can belong to multiple forms and they can be set as required by a required attribute. After the form is submitted and successfully validated, it can be sent as plain text or XML. Developers can use a hidden control with the name _charset_ to specify encoding of the transmitted data - this has previously not been handled in HTML.

2.2.2.2. Repetition model

Some forms have parts which can be repeated multiple times. Items in a shopping cart with total for the order is an example. With Web Forms it will be possible to include repeating parts in a form without resorting to client scripting (note that XForms offer similar capability, which I haven't covered in my introduction to them). It will be possible to add or delete repeating blocks and change their order by moving them up and down. Repeating blocks can be nested.

Performing dynamic calculations over the form is made easy by the output element and its onforminput event handler. Example follows (taken from [18]):

```
<form>
 <p>
   <input name="a" type="number" step="any" value="0"> *
   <input name="b" type="number" step="any" value="0"> =
   <output name="result" onforminput="value = a.value * b.value">
     0
   </output>
```

```
  </p>
</form>
```

When the two number are filled, their total will appear in place of output tag, which behaves similar to span tag. This example also demonstrates the new number input with a starting value of `0` and no step defined.

2.2.2.3. Current state of Web Forms

As of writing (May 2007) Web Forms 2.0 “are in the very final stages and will very shortly become a call for implementations” [18]. Some of the features have in fact already been implemented in the Opera browser. It seems very likely that other browser vendors will soon come up with implementations of their own and developers will be able to start using the spec. Backwards compatibility however will be a problem for a long time and will limit usage of Web Forms 2.0 to non-public applications.

2.2.3. Comparison

XForms and Web Forms as so much competing technologies as it could seem. While Web Forms are merely (although much needed!) update to the HTML spec, XForms are a whole new powerful XML based language, which can be used even outside the web development area. When used correctly, these two specs will complement each other very well. For simple applications, developers will probably use HTML forms, because they are much easier to learn and use. For sophisticated applications I would recommend XForms, either directly included in XHTML source and interpreted by a browser plug-in, or translated into HTML. New input controls defined by Web Forms 2.0 will make the translation even easier and possibly faster, because a lot of boiler plate Javascript won't be needed any more.

Chapter 3

Server-side development

3.1. Presence

Before I write about recent trends in server-side web application development, I think of it as necessary to summarize current technologies most frequently used first and say a few words about their positives and negatives. The three most popular environments for application development are probably Java Enterprise Edition, ASP.NET and PHP. In terms of design patterns the concept in web application development is the Model-View-Controller (MVC) pattern. Therefore, this chapter starts with a gentle introduction to it and in the later text I focus on how different frameworks use of it.

3.1.1. Model-View-Controller pattern

Model-View-Controller (MVC) pattern dates back to the 1979 when it was first described by Trygve Reenskaug, who was then working on Smalltalk at Xerox research labs [11]. The problem domain addressed by MVC is separation of the application into layers:

At the core there's an observation that applications typically consist of three more or less independent parts. First is the view, which deals with presentation of data to the user. It can be a form with GUI elements, an HTML source of a page or even an image, depending on the application. View usually needs some information based on which it can present itself meaningfully to the user. This information is encapsulated in the model. In practice, model can be represented by various data objects (e.g. a JavaBean). Finally, there is the controller which ties model and view together. It handles interaction with the user (HTTP requests, mouse clicks, etc.), then accesses the model and updates it according to the action of the user.

The benefit of breaking the application in three different parts is the possibility to replace one of the three components, with a different implementation. This can be found useful for instance when there are significant changes in the data backend, when changing the user interface or adding new output formats.

MVC model can by extended by the observer pattern. In that setup, the model knows when the underlaying data have changed and sends a message to the view to update itself. The Observer pattern however is difficult to apply to web application, because the view is sent to the browser and it cannot be notified by the model later. The next figure shows how different components interact in the MVC schema:

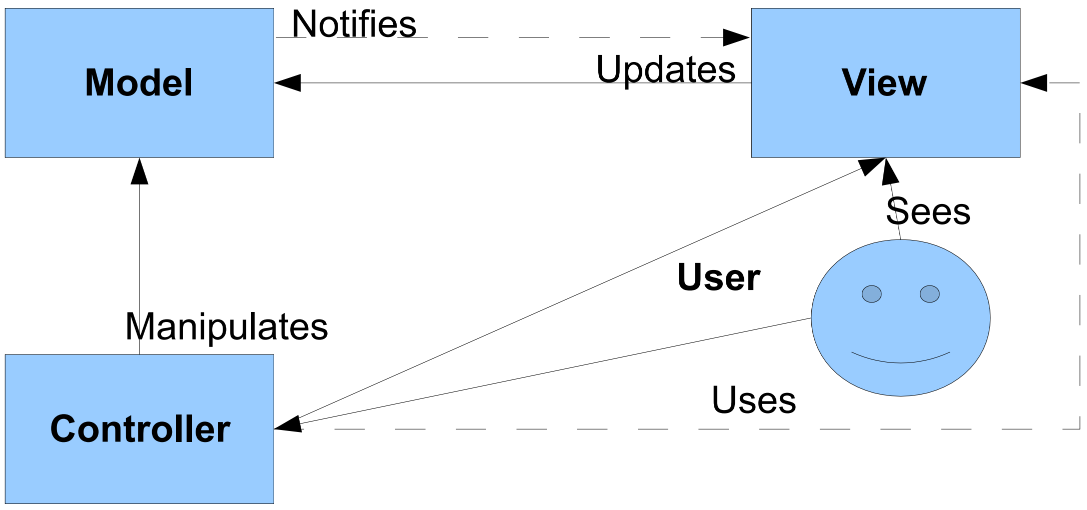

Figure 3.1. MVC pattern diagram

3.1.2. PHP

In the past, PHP was the scripting language that brought web application programming to the masses and it is still enjoying the position of number one web scripting language. According to usage statistics[1] available at PHP home page, it is as of November 2006 installed on over 19 million domains. This number makes it the only web scripting language/framework with massive deployment across web hosting companies - almost every webhoster offers PHP support in their basic package.

Due to popular demand PHP maintains a very short development cycle with numerous new features added each iteration, which was no doubt a reason for its popularity in past. This way, many functions enabling PHP for use with various network protocols, third party databases and application servers have found their way to the main distribution. Most of the time this happened with scant regard to naming consistency. Needles to say, PHP lacks some of the machinery that languages like Java, C# or Python have in place to deal with this kind of problems - name spaces, packages or

[1]<http://www.php.net/usage.php>

modules. Despite the fact that during time PHP evolved from a homepage creation utility to a platform used to develop rather sophisticated applications, it has always remained fairly simple - wide deployment made backward compatibility essential. More than ever the programmers are realizing this and it may be a contributing factor to its downfall in the future.

Lack of name spaces is but one of the points commonly criticised on PHP:

- In the past, the default configuration encouraged poor programming leading to countless security holes (such as publication of request data directly to the global name space).
- Many functions in the standard library have multiple aliases and naming standards are inconsistent between different parts of the standard library. This leads to confusion in team environments.
- It has no native support for unicode or multi byte strings.

Some of these problems are promised to be solved in the future versions. Also, large amount of fairly quality code is available through *PEAR*[2], modeled after Perl's *CPAN* software repository. This code can help overcome some of the deficiencies of the standard library. Among numerous other PHP web application frameworks, there is also an interesting framework (Zend Framework)[3] produced by the Zend Technologies[4] company. Many PHP developers however, are not well acquainted with object-programming concepts, which is why any framework in PHP is unlikely to get much following. Shift from simple web scripting language for the masses to a full featured web development platform may also not benefit PHP. As of writing this thesis, Zend Framework is in its 0.2.0 preview version and it will get some time for it to mature into production.

In short, even if version 6 would solve all of the problems of the previous versions, it probably won't save PHP from gradual demise. Besides, as was once proved during move from version 4 to version 5 (which brought some relatively minor backward incompatibilities - at least compared to what 6 will supposedly do), web hosting companies are reluctant to upgrade their setup. For serious future development, I would definitely not recommend PHP.

[2]<http://pear.php.net/>

[3]<http://framework.zend.com/>

[4]A PHP-focused company founded by Andi Gutman, Zeev Suraski and Rasmus Lerdorf, creators of PHP.

3.1.3. ASP.NET

ASP.NET is a successor to Active Server Pages (ASP) by Microsoft. It is based on the .NET CLR (Common Language Runtime), meaning that programmers can take advantage of .NET class libraries and programming languages. In practice, ASP.NET applications are usually written in C# and Visual Basic languages.

The architecture of ASP.NET is fundamentally different from its predecessor's. In the original ASP, an application consisted of individual ASP files which mixed together business logic and HTML presentation, very much like in PHP. This naturally resulted in spaghetti code difficult to maintain.

Programming in ASP.NET very much resembles the way desktop applications are developed. A page in ASP.NET consists of a number of controls or components that can react to events - this is called an event-driven GUI model. Some of the basic components bundled with ASP.NET include buttons, labels, data grids with support for listing pagination or calendar. It is possible to create custom controls or download third-party pre-made controls for free or for a fee. Presently, the market for ASP.NET controls is much larger than that for Sun's JSF components (see next section).

To make the event-driven model work, ASP.NET tracks information of the state of various controls on page in a ViewState variable, which tracks the state of controls and actions of the user. It is handed between pages as a hidden form field and the form must be submitted using POST method, due to ViewState's possible size. Sometimes even with ordinary hyper links must be triggered using Javascript code which makes them less accessible and impossible to be followed by web crawlers.

ASP.NET encourages separation of concerns using a concept called code-behind. Code-behind files are files that contain logic (among other the event handlers for various controls on the page), while ASP pages take care of presentation.

3.1.4. Java Platform, Enterprise Edition (Java EE)

Java EE, formerly known as J2EE, is one of the best known and most mature enterprise web technologies. First betas were released as early as in 2001 and it has enjoyed broad support by developers since then. Wide availability of software tools such as IDEs, application servers, frameworks and libraries is one of the main reasons for its popularity. Large base of existing Java code and platform independence are the factors when big corporations make decisions on web application development technology to use.

Basic components of every Java web application are Servlets and JSPs[5]. There are two commonly used design strategies used to develop applications in Java EE called model 1 and model 2. Model 1 is intended to be used only in very simple applications. Requests are directed either to a Servlet or a JSP which takes care of all necessary actions such as input data validation and communication with a data source. In practice however, model 2 - designed originally to be used in medium to large applications - is used almost exclusively. It comprises a combination of Servlets, JavaBeans and JSPs in a MVC setup, which separates content from presentation. Requests are dispatched to the Servlet (controller) which selects the view (JSP) that invokes methods of JavaBeans (model) to gather necessary data from databases or other sources. Important role in model 2 also play the so-called taglibs, which remove the need to embed Java code directly into JSPs and help factor it out into Java classes. There are numerous taglibs available for download that help easily accomplish common tasks.

In 2004 an interesting addition to the Java EE called JavaServer Faces (JSF) has been developed through the Java Community Process[6]. JSF is a component-based web framework very much like ASP.NET in its characteristics and can be thought of as a successor to pure JSP applications. It features two sets of taglibs to take care of user interface generation in HTML or other language as needed, a server-side event model and state management model. This makes it possible to program applications in a way akin to how traditional desktop applications are made. JSF also includes tools for form validation, error displaying, conversion of string request data into various other data types as needed and for internationalization. It makes building complex forms easy by providing a set of prebuilt complex HTML-oriented components such as date pickers or file upload fields. Possibility to customize renderers of the individual components can be used to add AJAX capability to forms with fallback capability where user agent doesn't support Javascript.

[5]Java Server Pages

[6]A formalized process that involves interested parties such as developers or software companies in the shaping of the Java platform's future.

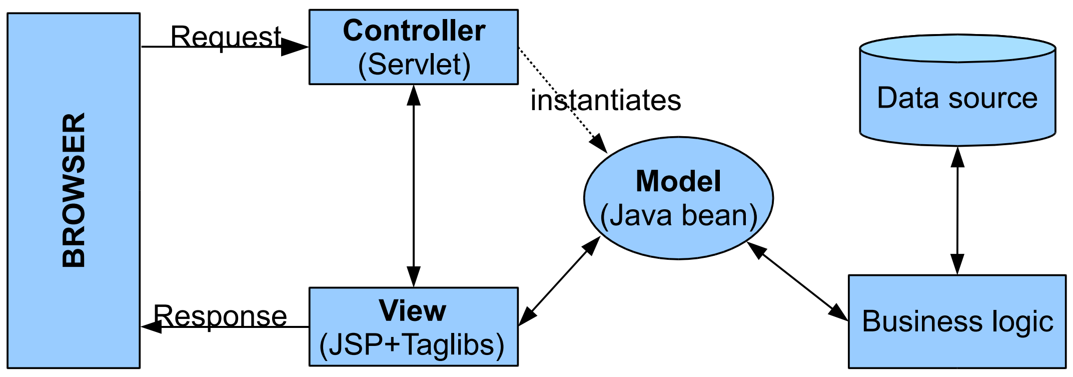

Figure 3.2. J2EE model 2 approach

For all greatness of JSF there is a price to be paid. First, the learning curve is relatively steep for an average developer - much to blame for this is the lack of comprehensive documentation. Resulting HTML code of the components is less transparent to the coders. This only adds up to the overall complexity of the platform. In terms of accessibility, JSF applications tend to suffer from same problems as ASP.NET applications. On the other hand, if the development team invests significant and succeeds in adopting JSF, it is rewarded with consistent MVC structure of the resulting application.

3.2. Trends

It is can be questioned what a trend is and what is not. I have decided to describe and compare four different frameworks: Google Web Toolkit (GWT), which uses Java as its programming language, Ruby on Rails (Ruby programming language), Turbogears and Django (both using Python). The reason for such decision is the growing popularity of dynamic object-oriented languages like Python or Ruby. I expect that at some point in the future frameworks based on these languages will replace PHP as number one development environment for web. In case of GWT, the reason for including it in my thesis is its innovative approach to Javascript programming (which essentially eliminates the need to write any Javascript code manually).

At the end of the chapter I will evaluate the four frameworks, point out their strengths and weaknesses points and provide guidelines for choosing the right framework for the task.

3.2.1. GWT - Google Web Toolkit

GWT - The Google Web Toolkit[7], a part of Google Code initiative, is a free library built to enable easy AJAX-enabled application development in the Java programming language. From the beginning Google intended to open-source GWT and it has finally done so with the 1.3 Release Candidate on the 12. December 2006 under the Apache license. Google also plans to publish notes from their internal team meetings.

GWT is a component-driven framework that makes heavy use of AJAX and uses a rather novel approach. Because it is quite difficult to unit test and debug Javascript code, engineers at Google have decided to eliminate it from the equation except for the output. GWT way, Javascript is only used as an *assembler for web*, of sorts. Such a step has many advantages for the developer, above all [14]:

- all favourite tools for Java development can be used (Eclipse, Sun Studio, etc.)
- Java language has static typing, therefore typos and type mismatches can be solved at compile time
- advantage can be taken of code-completion and hinting features in IDEs
- refactoring tools for Java can be used
- for most developers, object-oriented model of Java is easier to understand than the prototype-based concept of Javascript

Style of coding applications resembles GUI toolkits in standard Java (Swing). You create instances of UI controls and GWT renders them to the place in the HTML skeleton of your specification. You can manipulate properties of the controls programmatically and react to events on them the way you would normally do in Swing. Code of the event handler is then translated into Javascript including the asynchronous requests.

Naturally, some limitations are posed on the code that is translated into Javascript. Currently, only Java 1.4 is supported, which means you cannot use constructs from Java 1.5 such as generics. You are also limited in the parts of the Java API that you can use. At this point of time, most of java.lang package and parts of java.util package are supported. With time the number of supported packages will hopefully increase and versions of Java higher than 1.4 will also get supported. Javascript code is generated in multiple versions and only the compatible one is sent to the browser.

GWT is packaged with a wide variety of widgets built-in. They range from simple wrappers to basic HTML form controls (text input, password input, etc.) to menus, trees, various dialog boxes, tabbed interfaces and more. Layouts similar to those in

[7]<http://code.google.com/webtoolkit/>

Swing are used to position widgets on page. If they are not be enough you can either developed your own (in that case you cannot escape writing Javascript code) or use some of the free widgets available on the web for free.

3.2.2. Turbogears

TurboGears[8] is a pythonic rapid web-application development framework based on several components from other projects. It uses CherryPy as its application server and a basis for the whole framework, Kid as its templating system (although it can be adapted to use any other templating system of the developers preference) and SQLObject for the Object-Relation mapping (ORM) database abstraction layer. According to authors [3], TurboGears are built around three philosophical principles:

1. **Integration** - Python developer community has builds many good and useful tools - the problem is that they are often not designed to be integrated seamlessly. So, for instance, new CherryPy users are faced with a lot of unnecessary challenge. Not only need have to learn CherryPy itself, but also spend considerable amount of time researching what form validation framework to use, what templating system would best fit their needs and how to tie it all together. During that process they may easily encounter the problem of underdocumentation:

2. **Documentation** - "If it isn't documented, it doesn't exist." Indeed, many great Python libraries and tools have great developers, but they focus chiefly on writing clean and elegant code so when it comes to documenting their work for the potential users they don't have enough time or just don't care. The philosophy of Turbogears explicitly states that what is not documented will not be listed as a feature. Documentation is of prime importance and documentation created in the process of documenting the third-party components of Turbogears is contributed back to the original authors.

3. **Improvements** - Lot of new code and features are implemented in the process of Turbogears development. Authors try to make sure that their improvements find their way back to the original projects so both groups of users can benefit from it.

3.2.2.1. CherryPy - application server (controller)

Core of Turbogears is the application server which comes from an independent CherryPy project [9]. CherryPy is a powerful yet simple framework implementing basic

[8]<http://www.turbogears.org>

[9]<http://www.cherrypy.org/>

infrastructure for web application development. It maps python objects methods and functions to URL's and serves their output using HTTP. In CherryPy terminology, methods and functions must be *exposed* to be visible to HTTP user agents - this is done by means of a functional decorator[10]. An exposed method is called a page handler. Following code example shows this on the Hello World application:

```
1 import cherrypy
2
3 class HelloWorld:
4     @cherrypy.expose
5     def hello(self, name):
6         return "Hello %s!" % name
7
8 cherrypy.quickstart(HelloWorld())
```

First, we need to import the cherrypy to the current name space. Then, we follow by defining classes and their methods which will serve as page handlers. For the purpose of this code example I defined a `HelloWorld` class with a single `hello` method. Line 4 is a function decorator, which marks `hello` as page handler for CherryPy. Hello has one formal parameter called *name*[11]. One fundamental concept of CherryPy is that request parameters are available to exposed functions in a very natural way through parameters. These fall in two groups: named and positional. Named parameters are usually normal parameters from the query string or from body of a POST request. Positional parameters can become part of the URL and serve as a facility for SEO[12] friendly or semantic URL's.

Important metaphor of the CherryPy design (as of 3.0) are tools. Tools are classes with callback methods that can be attached to different stages of the request dispatch. A lot of basic functionality you expect from a web framework is implemented as tools: session management, input and output character set conversions, support for file uploads or error logging.

Such architecture makes the whole process of request dispatching very transparent to the developer. For instance, a user authentication library can be implemented as a tool to be run before any data are sent. In case a user would fail to authenticate, call to

[10]A syntactic feature in Python language, which allows to wrap a function call in another function (note that despite the similar name decorators in Pythons are not an implementation of decorator pattern).

[11]For those unfamiliar with Python object-oriented concepts, the first method parameter (*self*) references to the current instance of the class much like *this* does in Java and other languages.

[12]Search engine optimization

the exposed method associated with the request URL can be simply bypassed and a page informing the user of the need to log-in to gain access sent instead.

By default, CherryPy only serves contents of the exposed methods, static files such as cascading style sheets or images have to be configured using another tool. This ensures that users can access only the files or directories on the file system that we explicitly allow them to. In a production environment however, it may be better to delegate static file serving to a standard web server, which is better suited to handle large loads. Such approach is very often used with CherryPy applications, mostly in combination with Apache. Two possible methods are using mod_rewrite to proxy request to the CherryPy application server in a separate process or running CherryPy application using mod_python, which is the preferred way.

Out-of-the-box, CherryPy doesn't have much bells & whistles. This may be great if we want put together our own development platform or framework from scratch, but for developers who want to start hacking on an application right away, Turbogears can be a good choice, because they bundle CherryPy with additional libraries and wrap some of its sharp edges in a convenient package.

3.2.2.2. SQLObject - object relational manager (model)

To enable rapid database application development, Turbogears uses the SQLObject[13] library originally written and still maintained by Ian Bicking. SQLObject is an object relational framework not unlike Hibernate for Java, although much less sophisticated. It maps queries to relational database tables to an object-oriented interface. This way SQLObject achieves high level of database abstraction, which in most cases enables database cross-compatibility. Currently, SQLObject supports six database engines: MySQL, PostgreSQL, SQLite (including in-memory tables), Firebird, Sybase and MAX DB. All common features of these databases such as different kinds of joins and transactions are supported. Following example shows how a table is defined, created and how we can store a data in it (note that we haven't specified a primary key - SQLObject creates a synthetic one for us automatically).

```
>>> from sqlobject import *
>>> sqlhub.processConnection = connectionForURI('sqlite:/:memory:')
>>> class Person(SQLObject):
...     firstName = StringCol()
...     lastName  = StringCol()
...
```

[13]<http://www.sqlobject.org/>

```
7 >>> Person.createTable()
8 >>> p = Person(firstName="Vojtech", lastName="Jasny")
9 <Person 1 firstName='Vojtech' lastName='Jasny'>
```

On the first line, we import sqlobject into our current namespace. On the following, we create a connection handle to an in-memory SQLite database. Lines 3 to 5 define a class which represents a relational database table to store data about persons. The table will have two varchar columns and a synthetic primary key. Line 6 sends a `CREATE TABLE` query to the open connection. The penultimate line creates an instance of the `Person` class, which also sends an `INSERT` command to the database. From the final line you can see a string representation of the created object - a primary key value of one has been assigned to it on the relational table.

3.2.2.3. Kid, MochiKit (view)

From the presentation point of view, Turbogears are based on the Kid templating engine. This engine uses XML for template syntax, but can serialize output to a number of formats including HTML 4. This gives the flexibility to have HTML sources in XML language and be able to use all standard tools for XML validation and manipulation while still sending the old HTML format for which browsers still have better support. The XML source can be interspersed with python code using a XML processing instruction. The output of the python code will then be included in the resulting page, very much like with PHP scripts. Of course, best practice is to use SQLObjects for model and write business logic in the page handler, which represents the controller.

Individual templates are assigned to the page handlers declaratively using yet another decorator. The decorator function uses the return value of the page handler, which should be a Python dictionary, to assign variables to the template and then outputs the HTML code. This approach is very elegant, because Turbogears offer you an easy way to switch output from HTML to JSON generated from the returned dictionary. That way we can reuse it to easily build AJAX-enabled applications or to integrate AJAX functionality later.

To help getting and parsing the JSON data, Turbogears includes the MochiKit library[14]- which is a relatively light-weight Javascript library, that makes it easy to build dynamic websites. Using the combination of JSON and MochiKit you can create AJAX data tables with support for sorting or drag and drop user interfaces.

[14]<http://mochikit.com/>

3.2.3. Ruby on Rails

Ruby on Rails (RoR as it is often abbreviated) is one of the most trendy web frameworks with relatively large and still growing user base. It is based on the Ruby programming language. RoR mainly focuses on rapid application development for which it offers some interesting aids, such as scaffoldings. Models in RoR are built using an ORM called ActiveRecord. Scaffoldings are a way to easily generate HTML user interface for CRUD[15] operations on the model. You can either let RoR generate a basic scaffolding for you or use a command line utility called ScaffoldGenerator to create templates for scaffolding directly in the file system and modify them yourself. That way it is possible to minimize need to write repeating code for every application. Custom ScaffoldGenerators, which generate AJAX-enabled scaffoldings are available. RoR has relatively good documentation and it surpasses the other framework in terms of marketing.

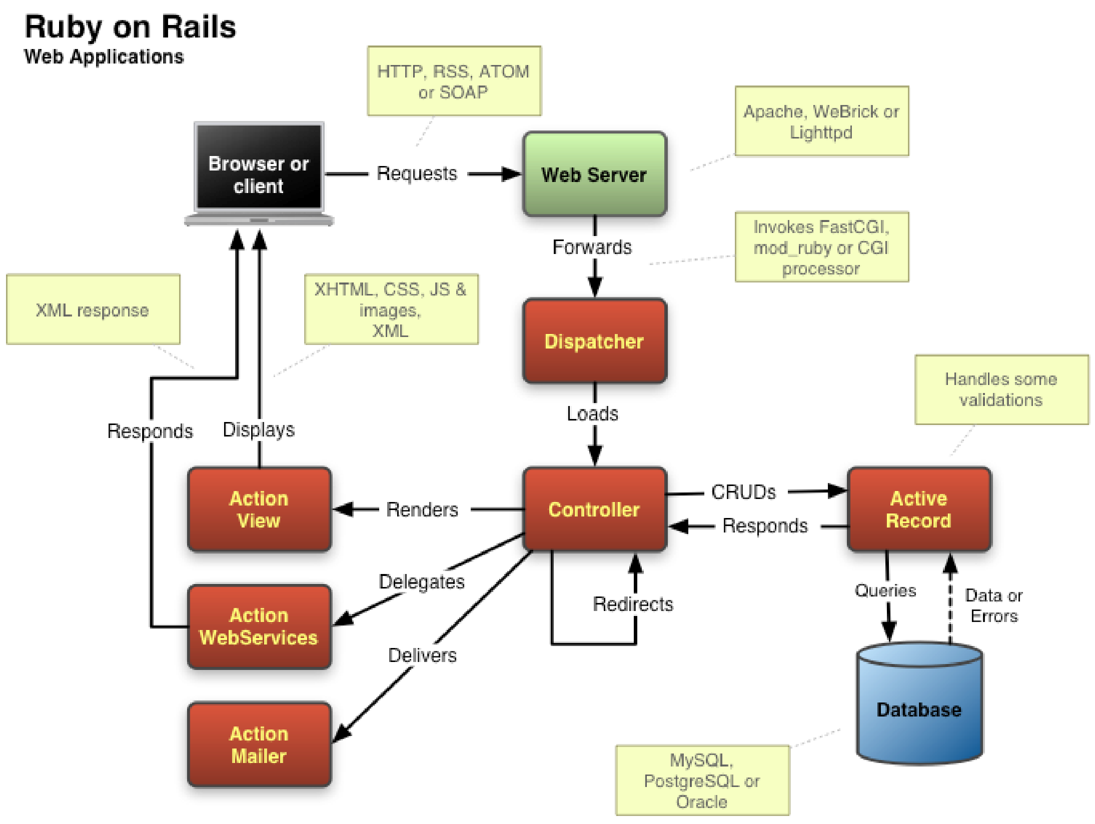

Figure 3.3. Anatomy of a Rails application [12]

Preceding diagram shows in the detail the control flow of a Rails application, but I will only comment briefly on it. First, the browser sends a request to a HTTP server,

[15]Create, retrieve, update, delete - four basic functions of a database.

which can be either a specialized web server such as Apache or WeBrick - a web server built-in Rails. As with most frameworks which incorporate web servers, running WEBrick behind a "real" web server is highly recommended for production environments. The web server forwards the request to the dispatcher, which loads an appropriate controller to handle it. The controller uses ActiveRecord model objects to do data validations and handle errors. After that the controller chooses a correct view and passes it the model to deliver data to the user. `ActionMailer` convenience class can be used to simplify the process of creating mailings.

3.2.3.1. ActiveRecord (model)

ActiveRecord is an ORM responsible for database abstraction layer in Rails - scaffolds make use of ActiveRecord model classes. Out of the box, ActiveRecord only has adapters for three database engines - MySQL, SQLite and PostgreSQL. If for instance you are developing an application based on legacy database schema on an unsupported database engine, the developers of RoR claim you can write an adapter for it in less than a hundred lines of code. Great in RoR is the Rake command line tool, using which you can keep track of model evolution and synchronize changes with the database.

3.2.4. Django

Django is a Python framework in many aspects similar to RoR. It was first released as open-source in 2005 (a year after RoR) but it has been in development since 2003. Although in a lot of aspects Django surpasses RoR, RoR tends to get a lot more publicity as a sort of Web 2.0.

An application in Django generally consists of models created using the built-in object-relational mapper, callback functions (not unlike in Turbogears), templates for presentation and a set of regular expressions to map the callback functions back to URL's. Oddly enough, the authors of Django insist on calling the callback functions views and argue that the functionality of controller is provided by the framework itself. Therefore, they refer to their framework as MTV - Model-Template-View, instead of MVC. Personally, I do not subscribe to their explanation, because in my opinion the callback functions are clearly Django's controllers and templates are the views. On the other hand I agree with authors that at the end of the day, it comes down to getting stuff done [13]. Subtleties of pattern naming are not so important.

To help rapid application development, Django offers a feature which resembles scaffoldings in RoR - the automatically generated administration site. Using the admin site you can get very quickly from model definition to working interface that can be used to enter data. In a production environment, the programmer can design the model first, activate the admin site and let users responsible for content fill it with data while he is building the publicly visible part of the site. Django follows the DRY principle[16] and unlike ScaffoldingGenerator in RoR it generates no code for the admin site - instead, all code is reused.

In addition to basic functionality, Django offers contributed "mini-frameworks" for authentication, content syndication (RSS, ATOM), page comments and so on. The automatic admin site is also implemented in this way. The contributed frameworks are decoupled from the rest of the framework and it is completely up to the programmer whether to use them or write his own code. In the following text, I will stick to the MTV terminology used by Django authors.

3.2.4.1. Model

It is very easy to define models and their relations using the Django ORM. Currently, the ORM library supports three database engines: MySQL, SQLite and PostgreSQL, with more under way such as Firebird or Oracle. There is also a branch of Django which uses more advanced SQLAlchemy[17] ORM. When completed, it should be possible to switch between the two different database abstraction layers just by importing a different Python package.

With all the models defined, you proceed to creating public interface of the application. In this process you are assisted by data manipulators and form wrappers. A data manipulator is a class that knows how to either create a new object of certain type or modify it and also encapsulates validation. The basic validation such as required fields or length of text fields is based on the definition from the model so there is no need to write duplicate code. A `FormWrapper` is a class that is assigned to the template and knows how to display form fields for a model object instance and validation errors in the data.

In future versions these concepts will be replaced by a new newforms package. This package will contain classes to completely wrap validation and form generation and also provide means to generate these classes from the models directly. As of writing the code is still undergoing changes but has become reasonably stable for production use. However, the documentation is still lacking.

[16]DRY - Don't Repeat Yourself principle

[17]<http://www.sqlalchemy.org/> - ORM database toolkit for Python with support for multiple database engines.

3.2.4.2. View

Views are functions whose output is sent to the user when a URL is accessed. It is mapped to the URL's by a set of regular expressions called *urlpatterns*. Using *urlpatterns*, great level of URL flexibility can be achieved (one view may be accessed using multiple URL's and so on). In the regular expression patterns you use named sub-patterns to pass named parameters to the view functions.

The view is the point where model and form objects get instantiated and are used to validate and modify underlaying databases. At the end of view a template can be displayed, a redirect performed or an HTTP error triggered (such as "404 - Not Found" in case a correct instance of model object hasn't been found).

3.2.4.3. Templates

Templating in Django is made easy by the built-in templating library. It uses its own syntax in which the program flow control structures are delimited by curly brackets and bears a lot of resemblance to Cheetah (Cheetah - the Python-Powered Template engine[18]). Like Cheetah, it supports inheritance of the template files, but there is no need to compile the templates into Python objects before use, which makes the whole development process a lot smoother and faster. In scenarios where performance is critical, you can use of Django caching library. It should also be noted that the whole admin site is based on templates, so it is possible to modify it by extending them.

3.2.5. Evaluation

Comparing different frameworks between each other is difficult for several reasons. Most importantly choice of technology is often dependant on the particular environment. Some of the relevant factors can be: skills of developers in the team, legacy database software (does the framework support it?), legacy web applications (can application in language A integrate with application in language B?), licensing requirements to name a few. Frameworks often offer similar features and so choosing one is a matter of personal preference of the project leader to a degree.

To give some overview, I have put together a set of criteria on which I evaluate all four frameworks presented above. I use multiple kinds of criteria: purely qualitative (license), boolean, list (supported databases) and rating (excellent, good, fair, poor, unsatisfactory). It would be gross oversimplification to declare one framework an ultimate winner. Furthermore, all of the frameworks are actively developed, so the situation

[18]<http://www.cheetahtemplate.org/>

can change quickly. Following table should only serve as a basis for further examination of the frameworks:

Table 3.1. Comparison of key framework properties

Criteria	**GWT**	**Turbogears**	**Ruby on Rails**	**Django**
Open source	yes	yes	yes	yes
License	Apache 2.0 License[a]	MIT License[b], LGPL[c]	MIT License	BSD License[d]
Platform	Windows, Unix	Windows, Unix	Windows, Unix	Windows, Unix
Programming language	Java	Python	Ruby	Python
Object orientation	yes	yes	yes	yes
ORM	no	yes, SQLObject or SQLAlchemy	yes, ActiveRecord	yes, built-in, SQLAlchemy integration currently in development
Databases supported	GWT apps use JDBC as their database layer	• Firebird/Interbase • MS SQL • MySQL • Oracle • PostgreSQL • SQLite	• Firebird/Interbase • IBM DB2 • Informix • LDAP • MS SQL • MySQL • Oracle • PostgreSQL • SQLite • SybaseASA	• MySQL • PostgreSQL • SQLite • Oracle and Firebird/Interbase in development
Widgets	yes, widget based	yes	no	no
AJAX support built-in	yes, AJAX widgets	yes, MochiKit	yes, prototype and script.acu.lo.us	no direct support for AJAX in the framework
Caching	no	no	yes	yes
Debugging	excellent	good	fair	good

Criteria	GWT	Turbogears	Ruby on Rails	Django
Documentation	good	good	fair	excellent
Availability of developers	excellent	good	poor	good
Support for unit testing	yes, integration with JUnit	yes, testutil package	yes, test/unit package	yes, using standard doctest or unittest packages
Internationalization	yes, using properties files and interfaces	yes, using gettext	yes, using a gettext plug-in	yes, using gettext
Localization	no (but you can use Java API)	yes, basic - offers a handful of convenience functions for date and number formatting	yes, using a plug in	yes, contrib.localflavor, offers a growing collection of locale-specific validators and variables (lists of states, etc.)
Deployment	using a servlet container	integrated web server, mod_python for Apache, WSGI	integrated web server, mod_ruby for Apache	integrated web server, mod_python for Apache, WSGI

[a]<http://www.apache.org/licenses/LICENSE-2.0.html>

[b]<http://www.opensource.org/licenses/mit-license.php>

[c]<http://www.gnu.org/licenses/lgpl.html>

[d]<http://www.opensource.org/licenses/bsd-license.html>

All of the frameworks are object-oriented, open source and available for both the Unix a and Windows platforms. Aside from GWT they have an ORM database layer, which is the result of GWT not being a full-stack framework. RoR and GWT both have widgets, but GWT is clearly more flexible in this regard, because they are a fundamental part of the framework.

All frameworks but Django have AJAX support built-in. It is currently under discussion in the Django developer community whether to add it. It the meantime, nothing stops you from using a Javascript library of your choice. RoR and Django have a very good support for HTML output caching, which comes in very handy on busy sites. Future version of Turbogears will likely include the caching layer from Django or some other Python project.

For the debugging, documentation and programmer availability criteria I have assigned the values myself. Quality of debugging on a particular platform depends highly on availability of IDEs, which is a strength of the Java platform. There is much less tool support for Python and even less for Ruby, which reflects the overall market share of the programming languages. In GWT debugging is even easier because you can develop client and server side in one environment. Availability of developers is also the result of popularity of the platforms. It can be easily verified on any job-seeker site that Java has the most open positions, followed by Python.

I have researched the frameworks relatively thoroughly in the course of writing my thesis and even developed applications in some of them so I dare to say that Django has the best documentation. Turbogears and RoR are a little behind but still very good and GWT is somewhat lacking. It has a good API documentation but there is not much tutorials available and areas such as deployment are not very well covered.

All frameworks have support for unit testing, internationalization and localization. GWT is deployed through any compatible servlet container, preferred deployment method for the other three frameworks is either through mod_python or mod_ruby Apache web server modules. Since the availability of web hosting is not very high (probably the best for Java for the time being), it is an advantage to have your own server.

Chapter 4

Conclusion

I have picked the most notable trends in web application development on both the client and the server side. Analysis of trends in any area, especially one as turbulent as web development, is always a very unrewarding task, one which necessary leads to a more or less biased view. I have tried hard to retain my objectivity, but I was necessarily influenced by the class of problems and projects I work on in my professional life. By definition of a trend, literature was not much of a help in my goal.

In client development, I have identified AJAX and COMET techniques as the most important trends and described them. I have explained their principles and outlined their strengths and weaknesses. I concluded that use of them without the aid of supplementary Javascript libraries is very complicated, due to great differences between execution environments in modern browsers. Subsequently, I proceeded with describing three Javascript libraries, explaining their unique functions and comparing them against each other.

Key deficiency of current web applications are forms - it requires unnecessary amount of working around and developer time to design user interfaces taken for granted by desktop programmers. I have described XForms and Web Forms 2.0, which in my opinion are likely to become successors to the current HTML forms. It is likely that in the future these two technologies will co-exist depending on complexity of a particular application.

On the server side I have directed my attention towards four different frameworks, each of which has its own target group of users, positives and negatives. As far as it is possible with software as specific as a web framework, choice of which is almost always subject to specifics of the application being developed and beliefs of the person involved in making the choice, I have tried to compare them. To make this goal feasible, I have devised a set of criteria which can possibly influence the decision making process.

Evolution in the problem area of my thesis never ceases and by the time I am finishing this text, web application development may have taken a completely different direction. If not, it is certain that the situation will be a lot different in as little as a couple months, but it is of course difficult to predict in what way. In spite of this I believe that most of the technologies I have written about are viable in the longer run.

Bibliography

[1] PHP - Criticism [online], Available at: <http://en.wikipedia.org/wiki/Php#Criticism>

[2] XmlHttpRequest [online], Available at: <http://en.wikipedia.org/wiki/Xmlhttprequest>

[3] TurboGears: Project Philosophy [online], Available at: <http://www.turbogears.org/about/philosophy.html>

[4] SCHLOSSNAGLE, George: Advanced PHP Programming. 1st printing. Indianapolis (Indiana): Sams Publishing, 2004. ISBN 0-672-32561-6

[5] JSF Tutorial [online], Available at: <http://www.coreservlets.com/JSF-Tutorial/>

[6] Dynamic HTML and XML: The XMLHttpRequest Object [online], Available at: <http://developer.apple.com/internet/webcontent/xmlhttpreq.html>

[7] Introduction to XForms [online], Available at: <http://www.w3schools.com/xforms/xforms_intro.asp>

[8] DUBINKO, Micah: What Are XForms, Available at: <http://www.xml.com/pub/a/2001/09/05/xforms.html>

[9] DUBINKO, Micah: XForms Essentials, O'Reilly, 2003, Available at: <http://xformsinstitute.com/essentials/browse/index.php>

[10] Mozilla XForms Project, Available at: <http://www.mozilla.org/projects/xforms/>

[11] MVC [online], Available at: <http://en.wikipedia.org/wiki/Model-view-controller>

[12] UnderstandingRailsMVC [online], Available at: <http://wiki.rubyonrails.com/rails/pages/UnderstandingRailsMVC>

[13] Django Documentation FAQ [online], Available at: <http://www.djangoproject.com/documentation/faq/>

[14] Google Web Toolkit - Product Overview, Available at: <http://code.google.com/webtoolkit/overview.html>

[15] Dojo Developer Guide, Available at: <http://dojotoolkit.org/docs/book>

[16] The Django Book, Available at: <http://www.djangobook.com/>

[17] XForms 1.1, Available at <http://www.w3.org/TR/xforms11/>

[18] Web Forms 2.0, Available at <http://www.whatwg.org/specs/web-forms/current-work/>

[19] Yahoo! UI Library (YUI), Available at: <http://developer.yahoo.com/yui/>

[20] Yahoo! UI Library: Calendar, Available at: <http://developer.yahoo.com/yui/calendar/>

Glossary

AJAX	Asynchronous Javascript and XML - a technique used in web development.
ASP	Active Server Pages - web application technology by Microsoft, predecessor to ASP.NET
COMET	An AJAX based communication technique enabling server-push communication.
CPAN	Comprehensive Perl Archive Network - large archive of Perl software ranging from utility code to database interfaces.
CRUD	Create, read, update and delete - basic functions of a computer database.
CSS	Cascading Style Sheets - a style sheet language used to define appearance of HTML or XML documents.
GWT	Google Web Toolkit - free toolkit by Google enabling rapid AJAX-enabled web application development in the Java language.
IDE	Integrated Development Environment
JDBC	Java Database Connectivity - abstraction layer for database access in Java.
JSF	JavaServer Faces - a Java component-based web application development framework.
JSON	A subset of the object literal notation of Javascript. It is often used in AJAX systems to exchange data between the client and server programming languages (e.g. from Python to Javascript etc.)
JUnit	Unit testing framework for Java used by many developers.
LGPL	GNU Lesser General Public License - a free software license published by the Free Software Foundation.
mixin	A class in which is not meant to exists independently but instead inherited by other classes to add some functionality

	to them ("mixed in"). They encourage code reuse, but multiple inheritance is necessary to make them work.
MVC	Model-View-Controller - a design pattern in programming that separates business logic from presentation.
ORM	Object-relational mapping - a layer that maps database tables to objects.
PEAR	PHP Extension and Application Repository - a framework and distribution system for reusable PHP components.
PHP	PHP: Hypertext Preprocessor - a scripting language used mainly in web application development.
W3C	World Wide Web Consortium - an organization whose members work together in development of web technology standards.
widget	Reusable user interface component such as a calendar or a color picker.
XForms	XML based language created to enhance and replace traditional HTML forms.
XHR	Abbreviation often used for `XmlHttpRequest` class.