6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

Design concepts for data-intensive applications

Attila Adamkó

Department of Information Technology,
Institute of Informatics, University of Debrecen
e-mail: adamkoa@inf.unideb.hu

Abstract

Designing and maintaining Web applications are both major challenges for the software industry and researchers. In the early years of Web development, it was a common practice to approach Web applications by simply "building the application", but now the Internet is ubiquitous, and Web applications are commonplace.
In contrast with the early days of the Web, when most content was served from static HTML pages, today's applications are invariably database-backed and provide mostly dynamic content.

In this paper, we will try to give some basic ideas about Web site development, introducing modeling issues and techniques, presenting the general architecture of Web applications and their different implementations, and the role of XML.
We will consider the current techniques, implementations, tools, and introduce some (conceptual) problems as well.

Key Words and Phrases: Data-intensive Web applications, data models, XML, Web modeling techniques, modeling languages, multi-tier applications

1. Introduction

Web applications are growing in demand, complexity and size, thus making it difficult to systematically design and maintain general Web applications. The Web is being used for applications that contain significant business logic and have considerable user interface requirements. There are tremendous demands for Web applications to deliver highly complicated, dynamic and interactive services to users. The Web was not designed for the deployment of large applications that contain significant business logic, but several enhancements have been made to support this functionality of Web applications. A family of technologies such as CGI and JavaScript and a series of revisions of HTTP have been developed, and innovation

has been conducted to enhance the web for its new role as an application platform. Hence it follows that, the Web is now used for deploying applications that do more than merely deliver information and hypermedia, the purpose for which it was originally devised.
The background of this paper is a development of a Web Information System (WIS) and firstly my main purpose was to get an overview about Web Engineering. Since the nature of WIS differs from the nature of traditional information systems we need to examine the process of WIS design.

This paper is organized as follows. In section two, the general multi-tier Web application architecture is presented. Section three shows the principals of the design process and the main modeling phases. At this point we can now imagine a general web application, so we can take a look at some currently used methodologies in section four. In the last section, some considerations are described, influencing the development process because existing methodologies are not suitable for all cases. Some ideas are proposed that aims to make web applications easier to implement and maintain, and assist in the modularization of a large-scale web application.

2. Architectures for Web Applications

A data-intensive Web application is an Information System, which relies on the Web as the application infrastructure to perform its functionality. Web applications are distributed applications, and hence are at least two-tiered. They act as a special kind of client-server applications, where a large portion of the functionality is "pushed" back to the server side despite the fact that the Web does not define what is behind the server.

A Web application is called "data-intensive" if its primary goal is to make large amounts of data accessible for various kind of users.

Web applications are multidisciplinary (software engineering, database modeling techniques, network computing, and effective interface design). They are built in a constantly changing environment where requirements are unstable and the user community is wider than before. Web applications handle information from various sources (text, graphics, video, audio) dealing with structuring, processing, storing and presenting this information.
A modern web-based application has four layers, as shown in Figure 1:

- a client layer
- a middle tire, which includes:
  - presentation layer
  - business logic layer
- a data layer

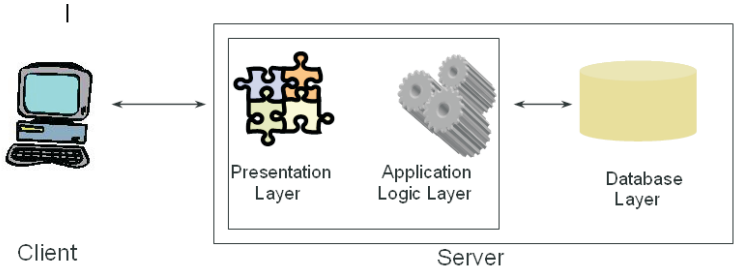


Figure 1: Typical multi-tier architecture

We need to answer two questions at each tier: what tasks can be performed by that tier, and the other is how components interact with the neighbourhood tiers.

Web application architectures are built around the well-known model-view-controller (MVC) pattern. The MVC pattern (shown in Figure 2) is used in web applications in order to separate business logic and data from presentation logic.

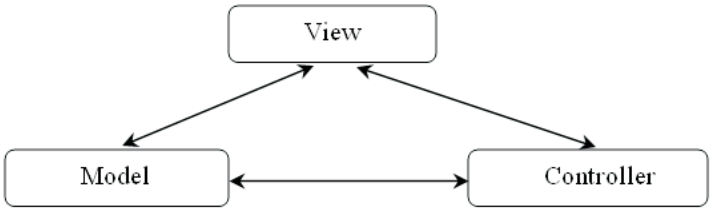


Figure 2: The MVC pattern

2.1. The Client layer

The client layer of a Web application is usually implemented as a web browser running in the user's client machine (nowadays the client could be a WAP–enabled mobile phone or a PDA).

The client browser requests "resources" located in the server, identified inside the system with a URL. The server is responsible for delivering that resource back to the requesting client and the main task in the client side is to displaying that information. Usually this resource comes back to the client as an HTML-formatted document that can be easily rendered by the browser, but some systems send back XML documents that can be transformed with style sheets.

The client can perform only a small portion of a task, usually related to user interactions and sending back user inputs to the server-side. Moreover the client

side cannot receive any messages from its server-side counterpart, and actually they are not working in a request-response manner at the top level. Thus we can conclude that the client tier of a web-based application is "thin".

2.2. Presentation Layer

The presentation layer is responsible for page generation including dynamic content. The dynamic content typically originates from a database. The other major task of the presentation layer is to "decode" the information coming back from the client side (e.g. find the user-entered data and pass that information to the business logic layer).
Web pages come in two flavours: static and dynamic. Static pages do not perform any server-side processing before going to the client. Dynamic pages do execute logic on the server and are often the principal triggers for server-side business logic processing in the system. In these case the final output text (HTML, WML, ...) is not stored on the server, but it is generated at runtime. This process may involve translation as well, for instance to translate XML code into HTML using XSLT style sheets.
The Presentation layer can be built in a number of different tools, e.g. Microsoft Active Server Pages (ASP) or Java Server Pages (JSP). These tools make it easy to embed dynamic content inside static HTML page templates, aiming at providing frequently needed functionalities to developers.

2.3. The Business Logic Layer

The Business Logic layer contains the determinant part of the application logic. It includes:

- performing all required calculations and validations
- managing workflow
  - state management: to keep track of application execution
  - session management: to distinguish among application instances
  - user identification
  - service access: to provide application services in a consistent way
- managing all data access for the presentation layer

The Business Logic layer is generally implemented inside an Application server (like Microsoft Transaction Server, Oracle Application Server, and IBM WebSphere). The Application server generally automates a number of services like transactions, security, persistence, connection pooling, messaging and name services.

2.4. The Data Layer

The data layer is responsible for managing the data. The job of the data layer is to provide the business logic layer with required data when needed and to store data when requested. It is generally a relational database, and connections are made by ODBC, JDBC or other interfaces used by different programming languages.

3. Design Principles

In this section some guiding principles are discussed for a general Web application, which is an Information System supporting user interaction through web-based interfaces. This interaction can be decomposed into three steps:

- Request: the user sends a request to the server, usually via a web page
- Processing: the server receives the request, and performs various actions. Then the result (usually a new page) is transferred to the client.
- Answer: the user's browser renders the results of the request.

The architecture of a Web application is a special kind of the server-client model where the clients are "thin" and the server-side is responsible for managing the whole functionality of the application. Implementing a Web site managing large amount of data, the following design principles should be considered:

3.1. Data-intensive Web applications should be based on a conceptual schema

This conceptual schema describes the information content of data resources and their semantic relationships regardless of their storage structures. The conceptual model provides a better understanding of the information content and improves the development process, usability, maintenance and interoperability of other data sources. It helps to create a suitable user interface for the application and helps in the content analysis and query formulation.

In the creation of the conceptual model some well-known data-modeling techniques can be used, for instance the Entity-Relationship (ER) or the extended EER model, or the Unified Modeling Language (UML) and their extensions. Nowadays the object-oriented approaches are suggested, using the well-known OO primitives (classes and relationships) and abstraction mechanisms (aggregation, inheritance).

3.2. Data derivation should be supported

Derivation is the process of extracting data from various sources and producing data from this information. There are two kinds of derivation:

- *External derivation*: building the data content as specified in the structural model. Derivation could be fully automatic if the data source is a single DBMS and the database schema is automatically derived from the conceptual schema.

- *Internal derivation*: producing different viewpoints of the same information. Derivation queries are applied to the conceptual model and define additional concepts, whose content is internally derived instead of being stored in the database.

3.3. Data navigation should be supported

Data navigation means that we focuse on another part of the application content along navigational path coherent to the conceptual structure of the application.
It includes: Sorting / Filtering / Indexing / Accessing / Browsing.
While the information-base consists of data, which can be accessed by users, navigational model describes the way how the users can access those pieces of information. A navigation object plays the role of a view defined on the structural model's objects and links between these navigational objects are specified as views on the relationships of the structural model.

3.4. Flexible page composition and flexible presentation styles should be supported

Page composition is a fundamental process of assigning data targets specified during structure modeling to abstract information nodes. Multiple page types per target allow the developer to represent the information on the same real-world object in different ways, e.g., for serving the needs of different users.

Presentation styles are responsible to construct the (out)look of the pages. The presentation specification should be performed at the conceptual level, independently of the specific language used to render the information content to enable multiple mapping from the same page to different realizations (HTML, XHTML, WML, and XML). The output of actual pages is created from presentation styles, page types and database targets and should be defined in a formal way. We can consider the screen as an abstract grid and construct the page from the above specified informational nodes.

3.5. Personalization could be supported

Personalization makes it possible to present a user – or specific classes of user – different views of the site (with different navigational commands and/or presentation styles). The main goal of personalization is to satisfy the different demands of distinct classes of users.
Personalization is essential in electronic commerce for supporting one-to-one marketing, an approach where each user is served by an apparently individual interface.

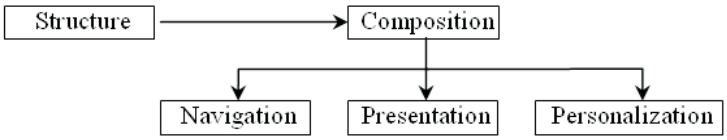
There are two ways to collect information about users or user-groups, and then create the appropriate profile:

- ***Declarative personalization***: static information collection, pre-plant user-specific information typically gathered with user's registration forms.
- ***Procedural personalization***: dynamic information collection about usage, typically monitoring the user's habits.

These two approaches are strictly independent of each other. Declarative personalization is based on static information and used for automatically construct information content and/or optional navigational options relative to user's interests. Procedural personalization use business rules – which is a triple *event-condition-action* – to express domain-dependent rules. These conditions react on page generation; if *event* occurs and *condition* is true then *action* will affect this process.

3.6. Design Dimensions

Web applications have five orthogonal design dimensions: structure, composition, navigation, presentation and personalization.



Each dimension represents a particular view of the application, and the application can be obtained by putting the layers together. If something needs to be changed it only affects a distinct part of the model and the application.

4. Modeling Phases and Methodologies

This section describes the phases of a typical Web Information System design from A to Z. Several research efforts have resulted in a number of proposed methodologies for WIS design, mostly model-driven. Different approaches deal with different models focusing on variant parts of Web applications. Typically, the methodologies consider the design process in terms of process phases and their deliverables, often models.

A typical WIS design methodology has the following phases:

- **Requirement Analysis**: gathering and forming the specification of the user requirements
- **Conceptual Design**: constructing the conceptual model for the domain

- **Navigation Design**: building the navigational model as a navigation view of the application
- **Presentation Design**: defining the appearance of the navigational units and their behaviour during user interaction
- **Implementation**

There are several Web modeling approaches proposed. Due to the close linkage of Web application with hypermedia, earlier Web modeling techniques were based on hypermedia concepts.
Later, it was found that modern software engineering approaches and models could easily be deployed to model most complex and dynamic Web applications.

We can consider various design models – such as OOHDM, RMM, WebML and UWE. These methods evolved out of research into hypermedia applications, tending to have a strong focus on information modeling and the design of effective navigation structures.

For example, OOHDM (which stands Object-Oriented Hypermedia Design Method) defines a four step development process. First, the conceptual design, the problem domain itself is modelled using UML-like graphical notations, providing a clear picture of the information which will be underpin the web site. In the second step, navigation design, representing the navigational structure. The third step of abstract interface design allows the developer to design the interface appearance and behaviour. And the final step, implementation.

Another modeling approach for designing Web sites is WebML (Web Modeling Language). WebML applies data oriented aspects using classical notations like the ER model and UML class diagrams. WebML enables the high-level description of a Web site under distinct orthogonal dimensions: its data content (structural model), the pages that compose the hypertext containing the information units (composition model), the topology of links between pages (navigation model), the layout and graphics requirements for page rendering (presentation model), and the customization feature for one-to-one content delivery. The distinct modeling specifications are written in the platform-independent XML language.

Naturally, there are several other available modeling techniques, each having their pros and cons, respecting the focus of the main modeling direction.

5. Special Remarks

Although a large variety of tools and modeling technologies are available to support Web Information System design, many of these heavily concentrating the conceptual layer, leaving untouched the final phase of the modeling process, the implementation. Accordingly in this section some considerations are taken to advance the model-driven Web engineering process.

XML technologies are used extensively in the web publishing world to support both the re-use of content and context-dependent delivery. Content may be represented as XML documents and selected content and presentation generated through

the use of XSLT templates and transformers. Universal client access is achieved by generating different XML formats such as XHTML for desktop browsers, or WML for WAP-enabled phones, and so on.
It could be better to manage information about entities at the conceptual level as database objects and dynamically generate XML documents to represent particular context-dependent views on those entities. This approach facilitates changes, avoids inconsistencies and promotes further forms of modularity and reuse.

On the more technical side, many Web applications are based on dynamic Web pages generated by scripts like PHP, Microsoft's Active Server Pages (ASP) or Java Server Pages (JSP). This introduces a problem into the development process; these scripts usually lead to a mixture of business logic and design elements within one software module. This has to be clearly separated into distinct modules to allow the reuse of both logic and design modules.

This separation between logic and design is an understood thing. However, the similar situation between business logic and data layer is not the same. The developer should aim to have little or no validation/business logic in the data layer since that logic belongs to the business logic layer.
Although, eradicating all business logic from the data tier is not always the best approach. Not null constraints and foreign key constraints can be considered "business rules" which should only be known on the business logic layer. Most would agree that it is safer or better to include such simple constraints in the database, and change them, as the business rules evolve.

Continuing this approach, we could take under consideration that databases systems supports view tables and data manipulation through these views (sometimes the help with trigger functions). Most of the methodologies are not utilizing this possibility, or if it is, then only for data retrieval. Another essential remark, a Web application could provide different interfaces for the Internet and for the Intranet. In this case the business rules must be implemented in two different modules. However, if we could find a tiny division between application logic and storage logic inside the business layer, we could ignore this problem by implementing the storage logic at the database layer, hence concentrating only the pure application logic in the business modules.

6. Conclusions

The first part of this paper serves as an introduction to the possibilities of modeling data-intensive Web applications. It focuses on the architecturally significant components, the design principles and dimensions and the key parts of the modeling process. Detailed information can be found in [1], [2], [5] and [7].

The last section covers some special remarks introducing hints for the development process and raising several implementing issues. Ongoing researches are opens up many interesting directions for further investigation and offers space for detailed analysis and development.

References

[1] **Ceri, S. at al..**: *Design Principles for Data Intensive Websites*, **SIGMOD record, Vol. 28, 1999.**

[2] **Ceri, S., Faternali, P., Bongio, A.**: *Web Modeling Language (WebML): a modeling language for designing Web sites,* **Proc. WWW9, 2000**

[3] **Conallen, J.**: Modeling *Web Application Architectures with UML*, **ACM, Oct. 1999, Vol. 42 No. 10**

[4] **Faternali, P.**: *Tools and Approaches for Developing Data-intensive Applications: A Survey*, **ACM Computing Survey, Vol. 31, 1999**

[5] **Schwabe, D., Rossi G.**: *The Object Oriented Hypermedia Design Method,* **Comm. of the ACM, Vol. 38, Aug. 1995**

[6] **Yogesh, D. at al.**: *Web Engineering*, **Journal of Web Engineering, Vol. 1, 2002**

[7] **Zhao, W., Kearney, D. and Gioiosa G.**: *Architectures for Web Based Applications*, **???**

Postal addresses

Attila Adamkó
Department of Information Technology,
Institute of Informatics,
University of Debrecen
H-4010 Debrecen, PO Box 12
Hungary