

A Layered Distributed Application: An Experience Report

Klaus Wolfmaier, Thomas Ziebermayr
Software Competence Center Hagenberg
e-mail: {klaus.wolfmaier, thomas.ziebermayr}@scch.at

Abstract – Web applications depend on many different infrastructure components such as Web Servers, Firewalls, or Application Servers. Furthermore these applications are accessed from different browser platforms on different hardware. Changing the infrastructure components affects the Web application architecture itself. Maintainable Web applications require a well-established practice to divide an application architecture into different layers (or tiers). This experience report presents an architecture for a Web application, that can be distributed over the Internet, based on SOAP for communication between client and server part. Advantages and shortcomings of the presented architecture are briefly described as well.

Keywords – Multi-Layer Architecture; SOAP; Web Applications, Distribution Pattern, Automatic Distribution

1 Introduction

The lifetime of a Web application is characterized by changing requirements for the application architecture, the user interface, or the system environment. Several reasons lead to changes; the application functionality has to be extended; the existing functionality has to be altered; or the underlying hardware and software components change and thus force to change the application implementation.

All these changes do not affect the application as a whole but only parts of the application (e.g. user interface, business logic, access to persistent storage). Ideally, only the changes for the directly affected parts are implemented and the whole application then can use the altered application parts. If the changes affect a client part that is distributed to many different locations, the distribution of the altered application parts is automated.

Constructing an application with independent parts requires a well-

established practice to divide an application into several independent layers. This separation ensures that one layer only contains specific parts of an application (e.g. persistent storage, business logic, user interface). Renzel and Keller [1] identify various layers splitting an application. Figure 1 illustrates five different distribution patterns that are described in [1]. According to these patterns, an application consists of five independent layers.

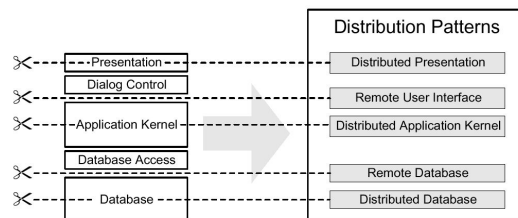


Figure 1: Distribution Patterns [1]

In the following, we describe the layers we implemented for a prototypical Web application in the banking domain and we describe the advantages and shortcomings of the implemented architecture. Experiences gained during the implementation of the prototype are described. The prototype presented in

this report is the result of an industrial research project carried out together with a large Austrian banking institute.

2 Application Architecture

The requirements for the prototype include several restrictions that have to be fulfilled by the prototypical implementation. A persistent storage for the application is needed, business logic must be implemented with *Enterprise JavaBeans (EJB)* – see [2] for further information on EJBs, and the application must be accessible over the Internet on geographically distributed workstations. Furthermore, distribution of the client part to the workstations should be automated.

Figure 2 depicts the architecture we implemented in the prototype. The architecture consists of the following layers: Backend, Business Logic, Communication, Distribution, and Client layer.

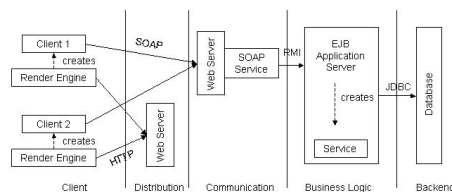


Figure 2: Implemented Architecture

Backend and Business Logic

A database is used as persistent storage. This database is accessed via *Java Database Connectivity (JDBC)* [3] that defines an abstract interface to a database. The business logic is implemented based on EJB technology.

Communication

Communication between client and server is implemented with the *Simple Object Access Protocol (SOAP)* [4]. SOAP is a platform-independent, XML-based communication protocol that is independent from specific programming languages.

Distribution and Client Layer

The user interface for the prototype is described in XML. A program (in the following called render engine) interprets the XML file and generates a user interface from the XML description. Such a render engine is installed at the client computer. The XML file can be downloaded from a web server, thus distribution of the client software is automated. A user interface described in XML is independent from a specific technology. To port client software to a different platform or a different device only an appropriate render engine on this device is required and automatically all applications that are developed in the described way are available on the new platform or device as well.

See [5] for a description of the user interface description in XML.

3 Experiences

The business logic implemented with EJBs accesses the persistent data via JDBC. Using JDBC provides an abstract interface to the database; concrete implementations of a database can be exchanged.

EJB technology, that is used to implement business logic, provides well-defined interfaces both to the backend and to the frontend. JDBC offers the abstract interface to the backend; the usage of SOAP for communication offers an implementation independent interface to the client part. If the implementation of the business logic is changed (e.g. for performance improvements) these changes do not affect the other layers of the application as long as the interfaces are not changed.

When the backend of an application or the component model is changed, these changes usually include a major change in the application architecture and functionality. Based on our experiences these changes occur less frequently.

Considering our experiences, changing the client application of a Web application occurs more

frequently. Changing the user interface of the application in the described architecture is little effort. The XML description of the user interface is altered and the new version is distributed to all clients the next time the client application is launched. These changes affect the user interface description and the SOAP communication layer.

To port the client application to a different platform on different hardware is also a task that might occur. To port the client application does not affect the Web application directly, but there is the need to provide a new render engine on the new platform that is able to interpret the XML user interface description.

The application architecture presented in this paper offers advantages for applications that run in heterogeneous software and hardware environments that are altered frequently. The client applications are implemented independent from a programming language and platform. To alter a client application, only an XML file has to be edited. For minor changes, no programming is necessary. The distribution of a client application to the client computer is done automatically via the Internet.

The main drawback for the described flexibility is restricted performance. SOAP is a text-based, programming language independent protocol. Optimizations for SOAP are needed to make it efficient (e.g. compression of SOAP messages). The user interface is generated at runtime (when the XML description is interpreted) every time the application is launched. Here again, optimization is necessary to speed up launching the client application.

Nevertheless, in our point of view the presented architecture is a possibility to design distributed applications that include the Internet for communication and that are distributed to a large number of wide spread client computers.

4 Acknowledgements

The authors gratefully acknowledge support of the K plus Competence Center Program, which is funded by the Austrian Government, the Province of Upper Austria, and the Chamber of Commerce of Upper Austria.

5 References

[1] Klaus Renzel, Wolfgang Keller; <i>Client/Server Architectures for Business Information Systems</i> ; Proceedings of the 4 th Pattern Languages of Programming Conference 1997, Illinois, USA
[2] Richard Monson-Haefel; <i>Enterprise JavaBeans</i> ; O'Reilly, 1999; ISBN: 1-56592-605-6
[3] Sun JDBC; http://java.sun.com/products/jdbc/ , July 2001
[4] Don Box et al.; <i>Simple Object Access Protocol (SOAP) 1.1 W3C Note 08 May 2000</i> , http://www.w3.org/TR/2000/NOTE-SOAP-20000508 , July 2001
[5] Altmann J., Wolfmaier K., Ziebermayr T., <i>Flexible Thin Client Architectures: An XML-based Approach</i> ; Proceedings of the International Conference on Internet Computing 2001, Las Vegas, USA