Implementation, Deployment, and Maintenance

During the design phase in the Web or software engineering process, the functional and non functional requirements that have been agreed upon by the customer and the developer are translated into a suitable software solution. Such a solution typically describes the conceptual decomposition of the overall Web application into a variety of constituent parts or modules, the arrangement of application data, the solutions adopted for communication purposes, and so on. The design phase, hence, produces as output a set of logical and conceptual solutions, architectural choices, and design patterns.

Such output is the input of the implementation phase where the elaborated, conceptual solutions are translated into proper application code. Implementing a design of a Web application typically requires the use of different technologies, programming languages, and code libraries; in Chapter 2, we introduced the most important technologies. The decision which technology to chose in a specific implementation situation does not always have an immediate answer. Typically, several (equivalent) implementation alternatives are possible. In this chapter we provide some insight into advanced implementation frameworks and tools (Section 7.2), as they represent best practices in industry and academia. For a better understanding of the frameworks and tools, we first discuss the adopted implementation practices for the presentation layer (Section 7.1).

Once an application has been implemented, its deployment requires some further decisions to be made. Going online with a Web application is not a difficult task, but there are a few issues that need to be addressed before the application can be accessed from the Web. For instance, typical decisions regard the selection of suitable Web server software, the physical location of the server, and the registration of the final domain name. Section 7.3 addresses these issues in more detail.

Finally, as the software life cycle is a continuous process, it requires proper maintenance and evolution efforts even after the final deployment of an application. Maintenance guarantees the continuous availability and functioning of the application; evolution regards the adaptation of the application to newly
emerging or changing requirements. Maintenance and evolution are the subject of Section 7.4.

7.1 Implementing the Presentation Layer

Web applications in general have been evolving steadily since the emergence of the Web in both their technological and their functional aspects. From a strict presentation point of view, we identify the following three steps in the evolution of Web user interfaces that heavily impacted the way a Web application’s interface is programmed:

- **Static HTML**: The first Web “applications” consisted of collections of static Web pages encoded in HTML and interlinked by means of embedded, static hyperlinks. Developing a static Web site was a daunting task, as each update of the site required the developer to manually maintain consistency in the collection of pages.

- **Dynamically generated HTML**: With the emergence of server-side scripting solutions, HTML pages could finally be assembled automatically, starting from a limited set of template documents. Especially, the maintenance of the typically large number of hyperlinks (think, for example, of today’s data-intensive Web applications) benefited largely from dynamically computed URLs.

- **Dynamic HTML and RIAs**: Today’s Web applications tend to augment their user interfaces with dynamic features also at the client side in order to minimize network traffic and to allow for highly responsive interfaces. In contrast to the server-side generation of HTML, the embedding of client-side logic also provides for fine-grained dynamic features, such as the update of single HTML elements.

We assume the reader is familiar with static HTML, and we hinted at dynamic HTML and RIAs in Chapter 2. The following discussion will thus focus on the dynamic generation of HTML markup on the Web server.

7.1.1 Template-Based Layout

The server-side generation of HTML markup is typically achieved by means of languages like JSP, PHP, and ASP.NET (cf. Chapter 2). A Web server that interprets a .jsp, a .php, or an .asp file intercepts the output sent by the program code to the standard output device and assembles it into the response (the page) that is sent to the user. Typically, the languages support not only the imperative, programmed specification of output, but also a hybrid approach, where HTML is interwoven with program code. Consider, for example, the PHP code of the example.php file in Figure 7.1. The code inside the `<?php ...

... ?>` instruction is the business logic of the page; the following lines are