WEB DEVELOPMENT EVOLUTION: FROM STATIC WEB PAGES TO RICH INTERNET APPLICATIONS

The Web has become one of the most important platforms for quickly and effectively transmitting information to people. In 1990, the Web started with HTML technology, which was originally devised to represent static information through a Web browser. The data flow was in a unidirectional way, from server-side to client-side. However, there was no personalization; most Websites did not authenticate users because there was no need.

Over time, Web applications took advantage for business purposes, showing dynamic content to users by enhancing client-side information technologies with JavaScript and bringing components like Applets and ActiveX, or plugins such as Adobe™ Flash™. On the server-side, there were information technologies that provided dynamism to the Web content. One of the first technologies for this purpose was CGI, which in 1993 represented a standard interface for passing dynamic content from server-side to client-side. Three years later, Java Applet emerged besides consolidated technologies such as PHP and Microsoft™ ASP. At that time, Web applications used the .Net as a platform for business, rendering dynamic content to users based on back-end business logic and database content. Developers enhanced the Internet’s capabilities to meet the demands of business applications by including scripts—JavaScript and VBScript—and components, such as Applet and ActiveX™ to the client (user) computer. While this user experience was to some extent enhanced, the fundamental “document-driven” synchronous approach of traditional Internet applications remained the same. As user experience requirements grew in complexity, this approach proved to be a handicap for complex business applications. Though there had been significant progress in server-side implementations, the means of rendering information to the end-user remained the same. As a result, the need to incorporate the rich, interactive, and responsive features of desktop applications was increasingly felt.

At that time, there was also a necessity of incorporating the main features of desktop applications: responsiveness and interactivity. A deterrent to fulfill this necessity was the difficulty of handling multi-transaction business workflows. The one-way nature of traditional Internet applications represented continuous Web page refreshing, making it difficult to represent large amounts of complex data and denying rich user experiences.

The last phase in this Web development evolution is RIA (Rich Internet Application) starting with technologies such as Adobe™ Flex, JavaFX™, or Ajax. Ajax incorporates standards-based presentation using XHTML and CSS, dynamic display, and interaction using the Document Object Model, data interchange, and manipulation using XML and XSLT, as well as asynchronous data retrieval using
XMLHttpRequest and JavaScript binding everything together. Adobe™ AIR™ lets developers use Flex technology to build RIAs that deploy on a wide range of devices such as smartphones, tablets, televisions, desktops, and netbooks. AIR™ applications run across operating systems, and they are easily delivered by using a single installer file. With AIR™, Flex developers can use their existing skills, tools, and code to build highly engaging and visually rich applications that combine the power of local resources and data with the reach of the Web.

WHAT THIS BOOK IS ABOUT

This book is about the last phase in the Web development evolution (i.e., RIAs). In order to cover this topic, the following elements are identified and addressed: 1) development methodologies, 2) development dimensions, 3) application frameworks, and 4) development tools. In addition, some case studies aimed at implementing these concepts are discussed in this book. Therefore, this book is intended to provide a comprehensive view of current practices on RIAs development. RIAs are a new generation of Internet applications that combine behaviors and features of Web and desktop applications such as: 1) client-server architecture, 2) data-intensive handling and business logic execution both on the client-side and the server-side, which results in advanced mechanisms of client-server communication, and 3) highly interactive multimedia content. Therefore, this kind of Web applications allows users to do interactive data explorations through attractive visual interfaces, increasing usability and performance (Martínez-Nieves, Hernández-Carrillo, & Alor-Hernández, 2010). RIAs are the future of today’s enterprise Web applications: they enhance the user experience, deliver functionality of desktop applications, and provide the portability and data reach that enterprise Web applications provide. A RIA changes the way the client Web application interacts with the server, removing or minimizing frequent server transactions from the user experience. RIAs offer the best of both worlds: the rich client and the reach of the enterprise. “Rich” is a term to describe how engaging and interactive a user interface is at the presentation tier. “Reach” describes the connectivity to the other tiers of the application. RIAs promise a presentation tier that all users see in the same way regardless of the platform, coupled with the powerful data-rich environment that enterprise Web applications can deliver.

In Web development, there are three fundamental factors to be considered: 1) Web development methodologies, 2) Web frameworks, and 3) toolkits. Web development methodologies can be categorized into two groups: 1) Software Development Methodologies for Traditional Web Applications and 2) Software Development Methodologies for RIAs. The methodologies of the first group pose some problems accommodating Web-specific aspects in terms of their methods and implementation; however, they were the pioneer for the development of new methodologies or extensions of existing methodologies. Some of these methodologies are WSDM, SOHDM, OOHDM, and UWE. The methodologies of the second group were developed by taking into account existing methodologies in order to satisfy the needs and features of RIAs development. Some examples are RUX Method, OOH4RIA, OOHDM Extension, and UWE-R.

In the case of Web frameworks, the options for developing RIAs can be grouped into three categories: 1) JavaScript-based frameworks, 2) non-JavaScript-based frameworks, and 3) multi-device frameworks. The JavaScript-based frameworks are open source containing a set of pre-written JavaScript code, which help facilitating the development of the JavaScript-based applications. This allows the development of applications by using the same code written in these JavaScript frameworks instead of writing the
same line of code each time individually, which becomes more difficult and time consuming. Some examples of these JavaScript-based frameworks are Dojo, jQuery, Prototype, and Sencha ExtJS. The non-JavaScript-based frameworks are based on proprietary programming languages, and these frameworks are typically used under paid licenses. Some examples are Adobe Flex, JavaFX, Silverlight, and OpenLaszlo, to mention but a few. Multi-device frameworks cover two kinds of applications for mobile devices: 1) native applications that are applications written for a specific device’s hardware and operating system, 2) Web-based applications, and 3) hybrid applications, which are applications built using Web technologies and wrapped in device-specific native application containers. These frameworks have the ability to support multiple mobile platforms such as iOS™ and Android™ at the same time. Some examples of these multi-device frameworks are PhoneGap, iUI, ipFaces, and Sencha Touch.

In the case of Toolkits, there are different Integrated Development Environments (IDEs) for RIAs development. According to its architecture, these IDEs can be classified into the following two major groups: 1) standalone applications and 2) plug-in applications. Some examples of standalone applications for RIAs development are Adobe Flash Builder, NetBeans, and Microsoft Visual Studio, among others. As plug-in applications are SapphireSteel Amethyst plugin for Visual Studio™, E(fx)cclipse plugin for Eclipse™, and Eclipse4SL plugin for Eclipse™.

Figure 1 presents a general perspective about the aforementioned factors.

In this book, a novel view of RIAs development is proposed; it considers some issues relevant not only to rich UI design purposes but to software quality purposes. Thereby, the topics related to RIAs development that are covered in this book are multimedia support, AOP (Aspect-Oriented Programming) support, the use of design patterns, and the use of UI patterns, as depicted in Figure 1.

**Figure 1. A general perspective about the RIA development stack**
Multimedia content is crucial for RIAs. It provides richest content to Web applications by presenting information more dynamically and entertainingly. As examples of multimedia contents are videos and image galleries. The use of image galleries enhances the ability to customize views on the fly and provides real-time visual representations of a selected item. The possibility to represent graphics, audio, and video is considered an inherent ability of RIAs (Rich Internet Applications). In fact, along with UI transformations, visual continuity and temporal are factors affecting one of the distinguishing features of RIAs: the enhanced UI.

AOP allows adding new features to Web applications without changing the original source code. For instance, when a legacy application requires of an authentication method, it is possible to develop a separate method using AOP and without changing the original structure of the application. This is especially relevant for adapting legacy Web 1.0 application UIs to RIA UIs as a common practice in the development of RIAs. Some of the software quality factors that can be achieved in the development of RIAs by using AOP are: 1) Maintainability, 2) Extensibility, and 3) Reusability. Maintainability is very important for RIAs development, since it enables one to make changes as effectively and efficiently as possible. Moreover, the AOP provides a high level for maintainability. Extensibility is another crucial aspect in the RIAs development, because the new functionality sometimes needs to be added to an application that is in operation, and it is necessary to ensure that the application does not fail. Finally, Reusability is important not only in software development but also in the AOP. In fact, Reusability encapsulates the functionality required to be able to reuse it on subsequent occasions. This reduces development time and improves productivity. In the cases of both RIAs and software development, the development time is very expensive.

Design patterns are important in software development, as well as in the development of both traditional Web applications and RIAs. One of the main uses of the design patterns is for reusability purposes (i.e., for simplifying the work) and, thus, offering a solution that had been previously tested and which was successfully applied to a problem that manifests repeatedly. In fact, RIA technologies are based on the software reusability principle in the sense that UIs are built starting from reusable widgets or UI controls organized in component hierarchies. This is more evident in the case of the non-JavaScript-based RIA frameworks because of the declarative UI definition model based on UI markup languages such as MXML, FXML, and XAML. A well-known Design Pattern on Web applications development is MVC (Model-View-Controller). The MVC pattern separates the domain modeling, the presentation, and the actions into three classes and based on the user input: 1) Model, 2) View, and 3) Controller. MVC pattern is a fundamental Design Pattern for the separation of user interface logic from business logic. Fortunately, the emergence of Web applications has helped resolve some of the ambiguity, since the separation between the view and the controller is apparent.

RIAs design involves two main stages: 1) application structure design and 2) UIs design. This book focuses on the second stage of RIAs design. Rich UIs design typically involves the use of UI patterns to ease the interaction between users and applications. UI patterns enable more intuitive and responsive user experiences (i.e., rich user experiences). Indeed, they allow developers to encourage users to engage with applications. Most of the RIA frameworks offer simple UI controls that natively implement UI patterns (e.g., progress bar [progress indicator pattern] and accordion [expand/collapse pattern] in Adobe™ Flex or JavaFX™). However, some UI patterns require more developmental time and effort in order to be implemented. Therefore, it is not easy to understand the differences between UI controls and UI patterns.

Nowadays, RIAs development demands design principles of Web and desktop applications, which are implemented by the so-called interaction design patterns. Furthermore, mobile devices, such as
smartphones and tablet computers, have also been involved in RIAs development due to the ubiquitous requirements of Web 2.0 applications (Finkelstein, Savigni, Kimmerstorfer, & Pröll, 2002). In this sense, RIAs are known as multi-device RIAs. This term covers RIAs that run as cross-browser Web applications, cross-platform desktop applications, and mobile applications. A Web browser-based RIA is a Web 2.0 application that integrates desktop-like features. A desktop-based RIA is a kind of application that is able to run off-Web browsers. A mobile RIA is a native mobile application with an improved UI.

The new trends in the development of RIAs can be identified by analyzing the steps on the Web evolution, from Web 2.0 to Web 4.0, passing through cloud computing as a trend on the Web 2.0 evolution. Cloud computing could be the best example of where the RIAs development is going both in commercial and academic fields. A new kind of RIA known as semantic RIA has recently emerged in the context of Web 3.0. Semantic RIAs try to solve the issues related to interoperability between systems by using ontologies and linked data principles. In the case of Web 4.0, the necessity of a ubiquitous Web has led to the emergence of context-aware Web applications by taking advantage of RIA technologies in order to offer rich user experiences.

Figure 2 depicts the conceptual map that illustrates the structure of this book.

Figure 2. Conceptual map of the general structure of the book
Hence, this book is structured as follows:

Chapter 1, titled “Basic Concepts on RIAs,” presents an overview of RIA features, and it explains the most important concepts for RIAs development, as well as their benefits and importance in several domains. In this chapter, a standard architecture for RIAs is described. This architecture has three well-defined layers: 1) the client-side that renders the rich user interface, 2) a controller layer where the business logic is executed, and 3) a data transactions manager. Important domains of Web development are presented, and in each one of them, the importance of RIAs is explained emphasizing particular features of each domain. Finally, experiences and successful stories of using RIAs in B2C e-commerce domain are presented. Some of these successful cases mentioned are Kodak EasyShare, MINI USA, FootJoy™, The Broadmoor Hotel™ and BlueGreen Vacation Rentals™, Verizon Wireless™, Yankee Candle Company, and Charles Schwab.

Chapter 2, titled “Frameworks for RIAs Development,” describes the most popular options for RIAs development. For practical purposes, these options are classified into two categories: 1) JavaScript-based frameworks and 2) non-JavaScript-based frameworks; even though there are other classifications reported in the literature, this classification is used in this chapter and throughout the book. In fact, this classification allows for a quick understanding of the technologies for RIAs development by abstracting technical details about intended software architectures.

Chapter 3, titled “Software Development Methodologies for Traditional Web Applications and RIAs,” presents a review of the state-of-the-art on methodologies for RIAs development. For this purpose, methodologies for traditional Web applications development are firstly presented, since, in some cases, methodologies for RIAs development are either extensions of existing Web (and hypermedia) methodologies or new UI design methods used on top of already existing Web methodologies. New approaches covering the RIAs features without relying on legacy Web models are also discussed in this chapter.

Chapter 4, titled “Important Factors on RIAs Development,” discusses some software quality metrics such as usability, scalability, and reusability of RIAs development. The chapter describes the importance of using multimedia content and UI (User Interface) patterns not only for improving the appearance of RIAs but also for delivering rich user experiences. Likewise, this chapter depicts the importance of leveraging AOP (Aspect-Oriented Programming) capabilities and implementing design patterns to ease the RIAs maintenance and enable RIAs reusability.

Chapter 5, titled “Multimedia Support for Native/Embedded Video Playback on Frameworks for RIAs Development,” discusses the capabilities of RIA frameworks in the context of multimedia content support. For this purpose, several alternatives for implementing video playback functionality are presented by using both JavaScript-based RIA frameworks to and non-JavaScript-based RIA frameworks. Examples of JavaScript-based RIA frameworks having multimedia content support are Dojo, jQuery, Prototype, and Mootools. Examples of non-JavaScript-based RIA frameworks are Adobe Flex™, JavaFX™, Silverlight™, and OpenLaszlo. For each case study the mandatory files are mentioned. The chapter also shows a screenshot where video player is displayed and code snippets that were used are presented. Finally, a comparative analysis of video playback support for each framework is presented in terms of video container formats and encoding types.

Chapter 6, titled “Aspect-Oriented Programming (AOP) Support on RIAs Development,” emphasizes the importance of employing Aspect-Oriented Programming (AOP) on software development, especially in software engineering. Some advantages in the development of RIAs by using AOP are Maintainability, Extensibility, and Reusability. This chapter presents a review of several success stories of AOP implementation in real world development projects and discusses the lessons learned in these projects.
The works analyzed in the state-of-the-art are classified by Web development, Usability Engineering, and other related perspectives. Finally, the chapter also addresses AOP support between JavaScript-based RIA frameworks and non-JavaScript-based RIA frameworks providing either native or third-party AOP facilities. Some code snippets depicting the use of these facilities for implementing AOP concepts are also presented.

Chapter 7, titled “Design Patterns Support for RIAs Development,” presents a review of Design Patterns proposed by GOF classification is presented. GOF classifies Design Patterns in Creational Patterns, Structural Patterns, and Behavioral Patterns. Some implementation examples of GoF Design Patterns are presented by using both JavaScript-based frameworks and non-JavaScript-based frameworks for RIAs development. Additionally, the chapter also presents a comparative analysis that summarizes the review of the capabilities of the RIA frameworks in the context of GoF Design Patterns support between JavaScript-based RIA frameworks and non-JavaScript-based RIA frameworks. Finally, this chapter also mentions how to solve different programming problems by using Design Patterns.

Chapter 8, titled “UI Patterns Support on RIAs Development,” a review of UI patterns supported by using non-JavaScript-based frameworks and JavaScript-based frameworks for RIAs development is presented. For this purpose, the UI Patterns are classified in Transitional Patterns, Lookup Patterns, and Feedback Patterns. Additionally, this chapter presents a series of applications samples depicting the use of not only UI controls but also non-visual functions or classes (as applicable) for implementing the supported UI patterns. These applications are intended to clarify the differences and similarities between UI controls and UI patterns. Finally, the chapter presents a comparative analysis that summarizes the review of the capabilities of the RIA frameworks in the context of rich UI design.

Chapter 9, “Case Studies Using JavaScript-Based Frameworks,” discusses the development of four different SOAP Web services-based thin-client applications using jQuery, Prototype, Dojo, and JSP. The chapter, indeed, addresses the development of a world atlas application, a document indexing a search system, an e-procurement system for the healthcare domain, and a meta-search engine for eBay and Amazon products. These case studies, as well as case studies presented in subsequent chapters, exemplify the use of some UI patterns for accomplishing rich design principles such as stay on the page and use transitions. Similarly, reviews of some similar real world implementations publicly available on the Internet are provided for each case study presented in this chapter. Finally, the chapter presents a review of the support for invoking SOAP-based Web services using Java. This is intended to be a preamble for the case studies and the use and application of SOAP-based Web services.

Chapter 10, titled “Case Studies Using Adobe Flex™,” presents two case studies about the development of REST (Representational State Transfer) Web services-based RIAs using Adobe Flex™ along with HTML and JavaScript. The chapter addresses the development of a meta-search engine for YouTube and Vimeo videos, as well as the development of a forecast weather system based on Yahoo! Weather. In addition, the chapter also offers a review of the support for consuming RESTful Web services in Adobe™ Flex™ as a preamble to these case studies.

Chapter 11, titled “Case Studies Using JavaFX™,” presents four case studies on the development of JavaFX™-based RIAs built on top of popular social networking Websites. The chapter addresses the development of three photo album applications based on Flickr™, Picasa™, and Photobucket™ Websites, respectively. In addition, the development of a search engine for movies based on Rotten Tomatoes™ Website is finally presented. For each case study, the chapter provides a review of some similar real world implementations publicly available on the Internet.
Chapter 12, “Multi-Device RIAs Development,” introduces the concept of multi-device RIA to describe a kind of RIA that can run on a variety of devices starting from the same code base. This includes not only cross-browser Web applications but also out-of-browser applications, namely cross-platform desktop and mobile applications. Thereafter, the chapter presents an overview of multi-device development frameworks. The supported platforms and the main capabilities are described for each framework mentioned. Furthermore, this section includes both a brief description of the most popular mobile operating systems for mobile devices and a comparison table for summarizing the main features of the multiplatform development frameworks.

Chapter 13, titled “An Overview of RIAs Development Tools,” provides a review of RIAs development tools, including both official and third-party tools. For this purpose, the two major approaches for RIAs development already identified in the literature were considered: 1) framework-based development approaches and 2) MDD approaches. However, this classification was extended by introducing a third category: RAD approaches. Thereby, the chapter reviews not only IDEs for RIA frameworks-based developments but also addresses other support tools for RAD such as code generation tools. Likewise, based on their architecture, IDEs for RIAs development are reviewed and classified into the following two major groups: 1) standalone applications and 2) plugin applications. Some development tools presented are Adobe™ Flash Builder™, NetBeans IDE™, Microsoft™ Visual Studio™, WebRatio™, RUX-Tool™, and AlexandRIA.

Chapter 14, titled “AlexandRIA: A Visual Tool for Generating Multi-Device RIAs,” explains the use of AlexandRIA for developing cloud services-based RIAs by implementing UI patterns. Unlike other RIA development tools described in Chapter 13, AlexandRIA is entirely focused on the fully automatic source and native code generation of RIAs, and it entirely addresses multi-device RIAs development. In this chapter, the use of the UI design process behind AlexandRIA is also demonstrated by means of a sample development scenario addressing the development of a cloud services APIs-based cross-platform mobile RIA. This scenario is further revisited in a case study that addresses the automatic generation of an equivalent application using AlexandRIA. In addition, this section of the book provides a review of both the UI patterns and the cloud services APIs supported by AlexandRIA as a preamble to the case study.

Chapter 15, titled “New Trends on RIAs Development,” presents the new trends on RIAs development by considering the steps on the Web evolution, from Web 2.0 to Web 4.0, passing through cloud-based RIAs development and mashups-rich UIs development as two easily visible trends related to Web 2.0. Semantic RIAs, RMAs (Rich Mobile Applications), context-aware RIAs, and ubiquitous Web applications are some of the academic proposals related to Web 3.0 and Web 4.0 that are discussed in this chapter. Similarly, some commercial efforts from companies such as IBM™, TopQuadrant™ Inc., and Salesforce.com Inc. are also discussed in this chapter.

TARGET AUDIENCE

This book is targeted at professional developers and students wishing to improve their knowledge of Web engineering and of the way it can be applied to the RIAs development. Some of the concepts covered (methodologies, frameworks, tools) will assume a basic level of prior knowledge and understanding. If the reader requires any further reading related to these topics, the authors provide a list of suggested titles at the end of each chapter. If the reader wishes to learn how to develop RIAs, as well as about all the features of this kind of application, this is undoubtedly the book for them.
CREDITS

Whilst some of the topics covered in this book were implemented based on the authors’ personal experiences, many of them have been previously identified and explained by other people. This work is an effort of the combined experience from the authors and other researchers, whose studies are fully covered in the references sections.

_Giner Alor-Hernández_
_Instituto Tecnológico de Orizaba, Mexico_

_Viviana Yarel Rosales-Morales_
_Instituto Tecnológico de Orizaba, Mexico_

_Luis Omar Colombo-Mendoza_
_Instituto Tecnológico de Orizaba, Mexico_

REFERENCES
