Chapter 2
Frameworks for RIAs Development

ABSTRACT

Chapter 2 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. In the case of JavaScript-based framework, some frameworks were selected and analyzed such as Dojo, jQuery, Mootools, and Prototype. In the case of non-JavaScript-based frameworks, frameworks selected and analyzed were Adobe Flex™, JavaFX™, Silverlight™, and OpenLaszlo™. For each framework, the architecture, functionality, and properties are described.

1. INTRODUCTION

There are many options for developing RIAs (Rich Internet Application). RIA frameworks have become popular in recent years. “A framework is a defined support structure in which another software project is organized and developed. Commonly, a RIA includes support for programs, libraries, and an interpreted language in order to help develop different components of a project” (Viveros García & García Godoy, 2009).

According to their license type, RIA frameworks can be classified into open source frameworks – such as jQuery – and commercial frameworks – such as Adobe Flex™. The most popular options for RIAs development are described in this chapter in order to help developers and designers in the decision making process about the RIA technology to be used considering which best suits the features of the project to be carried out. This chapter discusses the different technologies for RIAs development.

2. FRAMEWORKS FOR RIAS DEVELOPMENT

In recent years, several classifications for RIAs have been proposed. These classifications address different aspects of RIAs, such as functional-
Frameworks for RIAs Development

ity, target runtime environment and, other more complex issues, such as the software development technology (Toffetti, Comai, Preciado, & Linaje, 2011). Some of these classifications of RIAs are presented below.

Four main aspects of the application development are considered basing on the user’s experience:

- **Rich Presentation**: RIAs offer client-side event-handling and widgets similar to desktop-based UIs. This permits partial page updates, support interaction with visual data representations, and multimedia content (e.g., audio, video).

- **Client Data Storage**: It is possible to store data on the client-side with different levels of persistence (in a temporal way while the application is running).

- **Client (and Distributed) Business Logic**: It is possible to carry out complex operations directly on the client-side, such as data navigation/filtering/sorting with multiple criteria, domain-specific operations, and local validation of data. It is also possible to distribute the Business Logic between the client and the server-side, (e.g., to validate some form fields on the client and others on the server-side).

- **Client-Server Communication**: RIAs support synchronous communication between client and server-side to distribute domain objects, data, computation, and provide server-push (e.g., in collaborative/monitoring applications) (Toffetti et al., 2011).

Depending on each of the application’s functionalities, the features above can be combined to obtain standalone applications, collaborative applications, or simply more appealing UIs (UI stands for User Interface) for existing Web applications. In terms of growing number of features and development complexity, a RIA’s may typically falls into one of the following types of application (that they can be possibly combined to obtain complex RIAs):

- **Traditional Web applications with RIA-makeover**: Where simple isolated RIA capabilities (usually for partial page updates) are added to a traditional Web application (e.g., Facebook™).

- **Rich UIs**: Web applications with widget-based UIs, where the client-side logic is an extension layer over the browser, superseding core browser responsibilities, such as handling events and managing states and the rich user interfaces components work in a coordinate way (e.g., Gmail™).

- **Standalone RIAs**: Web applications capable of running both inside and/or outside the browser in a connected and/or disconnected fashion (e.g., SlideRocket™).

- **Distributed RIAs**: Where the application data and logic are (sometimes dynamically) distributed across client and server-side. Moreover, on-line collaboration is supported and client-server communication is used to fill the gap between objects and events living across the application components (e.g., Google Docs™) (Toffetti et al., 2011).

Currently, RIAs capabilities can be implemented in a number of different client-side technologies. These technologies can be broadly classified into three categories according to the runtime environment:

- **JavaScript-Based**: The client-side business logic is implemented using the JavaScript scripting language (the approach is also known as “AJAX”, which stands for Asynchronous JavaScript and XML). Moreover, UIs are based on a combination of HTML (HyperText Markup Language) and CSS (Cascading Style Sheets). The

Recommend this product to your librarian:

www.igi-global.com/e-resources/library-recommendation/?id=1

Related Content

Future Trends
www.igi-global.com/chapter/future-trends/24668?camid=4v1a

Swarm Intelligence Applications for the Internet
www.igi-global.com/chapter/swarm-intelligence-applications-internet/16909?camid=4v1a

Web Services
www.igi-global.com/chapter/web-services/16922?camid=4v1a

Multicast of Multimedia Data
www.igi-global.com/chapter/multicast-multimedia-data/16870?camid=4v1a