Science of Computer Programming 102 (2015) 1–19

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Analyzing best practices on Web development frameworks: The lift approach

María del Pilar Salas-Zárate [a], Giner Alor-Hernández [b,*], Rafael Valencia-García [a], Lisbeth Rodríguez-Mazahua [b], Alejandro Rodríguez-González [c,e], José Luis López Cuadrado [d]

[a] Departamento de Informática y Sistemas, Universidad de Murcia, Campus de Espinardo, 30100 Murcia, Spain
[b] Division of Research and Postgraduate Studies, Instituto Tecnológico de Orizaba, Mexico
[c] Bioinformatics at Centre for Plant Biotechnology and Genomics, Polytechnic University of Madrid, Spain
[d] Computer Science Department, Universidad Carlos III de Madrid, Spain
[e] Department of Engineering, School of Engineering, Universidad Internacional de La Rioja, Spain

ARTICLE INFO

Article history:
Received 1 October 2013
Received in revised form 18 December 2014
Accepted 19 December 2014
Available online 5 January 2015

Keywords:
Best practices
Lift
Scala
Web frameworks

ABSTRACT

Choosing the Web framework that best fits the requirements is not an easy task for developers. Several frameworks now exist to develop Web applications, such as Struts, JSF, Ruby on Rails, Grails, CakePHP, Django, and Catalyst. However, Lift is a relatively new framework that emerged in 2007 for the Scala programming language and which promises a great number of advantages and additional features. Companies such as Siemens© and IBM®, as well as social networks such as Twitter® and Foursquare®, have now begun to develop their applications by using Scala and Lift. Best practices are activities, technical or important issues identified by users in a specific context, and which have rendered excellent service and are expected to achieve similar results in similar situations. Each framework has its own best practices whose aim is to facilitate the development of Web applications. However, there is no current comparative analysis that identifies the best practices for Web frameworks. Thus, as its salient contribution, this paper identifies a set of best practices for Web frameworks. Afterwards, these best practices were analyzed and discussed in terms of developing Lift-based Web applications. The identification of these best practices would allow developers to construct more interactive and efficient Lift-based Web applications, integrating features of Web 2.0 technologies with less effort and exploiting the frameworks' benefits. In addition, this paper contains a comparative analysis with Web frameworks such as JSF, Struts, CakePHP, Ruby on Rails, Lift, Django, and Catalyst. Finally, as proof of concept, a set of Lift-based Web applications were developed for this paper by applying best practices such as actors, lazy loading, Comet support, SiteMap, Wiring, HyperText Markup Language, version 5 (HTML5) support, and parallel rendering.

* Corresponding author.
E-mail addresses: mariapilar.salas@um.es (M.P. Salas-Zárate), galor@itorizaba.edu.mx (G. Alor-Hernández), valencia@um.es (R. Valencia-García), lrodriguez@itorizaba.edu.mx (L. Rodríguez-Mazahua), alejandro.rodriguezg@upm.es (A. Rodríguez-González), jllopez@inf.uc3m.es (J.L. López Cuadrado).

http://dx.doi.org/10.1016/j.scico.2014.12.004
0167-6423/

1. Introduction

Choosing the appropriate Web framework for Web development, which best fits the developers' requirements, is not an easy task, since there are many frameworks based on different languages. Moreover, selecting an inappropriate framework can lead to 1) wasting time studying the details of another language, 2) failure to meet the required time because developers are not used to the framework, and 3) spending time taking corrective actions to choose a different framework. In order to avoid these problems, it is highly important to know and identify the best practices for Web development.

A best practice is a process, technique, innovative use of technology, or a set of resources with a proven record of success in providing significant improvements in cost, schedule, quality, performance, safety, environment, or other measurable factors that impact on an organization [1]. Best practices on Web frameworks imply reducing development time and effort while saving money, increasing the quality of code, and providing the opportunity to create friendly and interactive applications.

A framework is a high-level solution for the reuse of software pieces, a step forward in simple library-based reuse that allows for sharing common functions and generic logic of a domain application. It also ensures a better level of quality of the final product, since one important part of the application is already found within the framework and, therefore, has already been tested [2]. Many Web frameworks based on different programming languages are now available: JSF and Strutts for Java-based Web frameworks, Ruby on Rails are Ruby-based, Grails is Groovy-based, and CakePHP is for PHP-based frameworks. However, another brand new type of Web framework emerged in 2007. Lift is a Scala-based framework, whose features are based on the experience of David Pollak, its creator, with the mistakes from other Web frameworks [3]. Lift possesses the advantages of Scala's functional programming. Functional programming (FP) is a programming style emphasizing on functions that return consistent and predictable results regardless of a program's state. As a result, a functional code is easier to test and reuse, simpler to parallelize, and less prone to bugs [4]. Scala is a hybrid functional and Object-oriented (OO) programming language, which means getting the power of the higher-level functional languages (such as Haskell, Scheme, among others) while retaining the modularity and reusability of OO components. FP concept of immutability is a specially well-represented feature in Scala, and it is one of the simplest means to provide high scalability. Scala allows for doing more in Lift with fewer lines of code [5].

Existing literature of Lift framework includes merely six books: 1) Simply Lift [6], 2) Exploring Lift [7], 3) Lift in action [8], 4) Lift Cookbook [9], 5) Lift Web Applications How-to [10], and 6) Entwicklung von Web-Applikationen mit Lift und Scala [11]. However, none of this information has yet provided a comparative analysis and a precise detection of Lift best practices in comparison to other Web frameworks.

Thus, this work presents a set of best practices to develop Lift-based Web applications. Moreover, this paper includes a comparison and discussion carried out among other well-known Web frameworks such as JSF, Ruby on Rails, Struts, Grails, CakePHP, Lift, Django, and Catalyst. Finally, the paper presents also the development of a set of Lift-based Web applications in which the best practices were applied and analyzed.

This paper is structured as follows: Section 2 presents the state of the art concerning Lift-based development and the use of best practices on Web development. Section 3 describes and discusses the use of best practices for Web development and presents a comparison table of Web frameworks in the use of best practices. Finally, the fourth section presents a set of Lift-based Web applications applying best practices, while future directions and the concluding remarks are presented in Section 5.

2. State of the art

Research has been proposed in order to identify and obtain the best practices for Lift-based applications development and Web development. Chen et al. [7] discussed the many advantages of using Lift framework and claim that Lift: 1) is a great framework for building compelling Web applications, 2) has been designed to make powerful techniques easily accessible, while keeping the overall framework simple and flexible, and 3) has cherry-picked the best ideas from a number of other frameworks, while creating some other novel ideas. These advantages are a combination of solid foundations and new techniques that make Lift so powerful.

Several research works have addressed some Lift and Scala issues. For instance, Pollak et al. [12] published the development of their multiuser and real-time chat application on Lift. This application provided a single chat server that took chat messages and redistributed them out to all listeners. In the same work, authors also presented Scala's language features such as singletons, pattern matching, traits, and immutable data types.

Wampler [13] equally addressed Scala's support for actors and messages for other Scala-based frameworks that are full-stack frameworks for building multi-tier applications. Some frameworks are "point" tools for specific parts of an application, like template libraries for generating webpages (analogous to Java Server Pages), while others focus on building particular kinds of networked servers like Representational state transfer (REST) response servers that are "headless." However, Play, Scalatra and Finagle service frameworks were only presented and discussed in Wampler's work.

Dong-Hong et al. [14] presented a new framework based on Scala and Lift which tried to solve the conflict of rapid delivery and repeated work. This framework was a new approach for agile software development. It could perform all general functions of information systems, including create, read, update, and delete (CRUD) operations and it had features

such as a good and flexible interface to the specific business logic. Also, the framework possessed Web 2.0 features and provided support for high-level user interactions and all popular database systems.

Several authors have expressed different points of view about which are the best practices. For instance, Stout [15] discussed the best practices for testing a Web application, and how the System Development Lifecycle (SDLC) has been beneficial on the use of these best practices. The work included references to various established processes and offered a brief case study. The author concluded that a successful Web application could be summarized as: usable, secure, scalable, reliable, maintainable and highly available. Also, he mentioned that if a website did not meet the aforementioned criteria, it could be considered as a deficient website. Thus, all these factors are needed to perform a test.

O'Reilly et al. [16] identified eight core patterns that are key to understand and navigate on the Web 2.0 era, and they also presented the problems that each pattern solved. This work provided a thorough analysis of market trends, proven best practices, case studies of industry leaders, and tools for hands on self-assessment. In the end, authors mentioned that in order to compete and thrive in today's world of Web 2.0 technology, decision makers – including executives, product strategists, and entrepreneurs – need to act now.

Hightower [17] discussed guidelines for Struts and related technologies like Validator framework, Java Server Faces (JSF), and Java Server Pages Standard Tag Library (JSTL). The research provided guidelines for using these technologies and techniques in order to develop Struts-based Web applications. The guides included background information, step-by-step instructions, and best practices.

O'Reilly et al. [18] focused on experiences related to J2EE Application Programming Interface (APIs). The J2EE APIs included acronyms such as Enterprise JavaBeans (EJB), Java Database Connectivity (JDBC), Java Remote Method Invocation (RMI), eXtensible Markup Language (XML), and Java Management eXtensions (JMX), which were also discussed in the authors' paper. Since J2EE is the most popular Java-based platform, a set of experiences from programmers was also presented. The research contained information about Java-based Best Practices, published by O'Reilly, which covered Java 2 Platform, Standard Edition (J2SE) APIs such as Swing, the collections classes, performance tuning, and NIO.

Ford [19] examined the important areas of architecture, design, and effective best practices or techniques that are based on the knowledge gathered. Some frameworks such as Struts, Tapestry, WebWork, Velocity, and Cocoon were evaluated in order to describe their use. The author also presented a comparison of the Web frameworks.

Golding [20] discussed the use and advantages of using frameworks, as well as the main features of the CakePHP Web framework. The author addressed aspects such as how to effectively build, structure, and organize Cake-based applications, and how to build more extensive Web applications. The paper equally explored Cake's built-in helpers, including the Asynchronous JavaScript And XML (AJAX) helper and explained how it worked with more advanced features.

Shenoy [21] explained and discussed the main foundations of Struts. The author emphasized on the importance of Struts from a usage perspective. The research also addressed a great number of practical issues and described some best practices and strategies. Certain topics such as components, actions, form validation, tag libraries, internationalization and exception handling were also presented in Shenoy's work.

Rao [22] discussed that the progress in the Java programming language has been slow over the last few years. On the other hand, he mentioned that Scala programming language was emerging as one of the possible successors for Java with features such as type inference, higher order functions, closure support, and sequence comprehensions. This allows for an object-oriented, yet concise, code to be written by using Scala. While Java-based Model-view-controller (MVC) frameworks are still prevalent, Scala-based frameworks along with Ruby on Rails, Django, and PHP, are emerging as competitors. Scala has a Web framework called Lift that borrows the advantages from other frameworks while keeping the code concise. In the end, Rao's work presented a Web framework called "Scala Server Faces," which provided a way of writing Scala-based Java Platform Enterprise Edition (Java EE) applications.

Hazrati [23] presented an example for single request processing. The Lift-based source code, running inside Tomcat, ran four times faster than the Rails-based source code running inside Mongrel. However, the Central Processing Unit (CPU) utilization was less than 5% in the Lift-based version, while for the Rails-based version it was 100% (on a dual core machine).

Ghosh et al. [24] discussed a common problem found in current Web programing models, which is the use of String values. Interactions between Web applications (including browsers) and Web servers occur primarily via textual representations – such as JavaScript Object Notation (JSON) – and via name-value String pairs. As a result, sometimes developers have to write a lot of source code dedicated to validate unstructured Strings at multiple layers of interaction. Authors stated that in these kinds of situations, a framework like Lift, which is able to abstract the request cycle via typed representations, could help solve the problem. Finally, the work also described the advantages of the Scala and Lift programming languages.

Galpin [25] developed a real-time Web auction by using advanced AJAX libraries such as jQuery. This technology made easier to write Comet-based applications on the client side, but the scalability on the server side was still a challenge. In this situation, Scala programming language and Lift Web application framework could step in and deliver a scalable back end for the Comet application.

Chen et al. [7] discussed that based on their Web development experience, Lift really solved problems more effectively than other frameworks. Moreover, the work presented the following key Lift's strengths: separation of presentation, content and logic, templating system, boot and schemifier, and snippets.

Although they are all important and comprehensive, the aforementioned works, lacked of two important aspects when they addressed Lift Web framework: a) the analysis of different Web frameworks based on their best practices and b) the

inclusion and discussion of Lift's best practices for Lift-based applications, since so far only those best practices of Java-based frameworks are well-known, and thus, able to be discussed. Therefore, the present paper seeks to address the aforementioned deficiencies by providing an overview of the best practices on Web development including Lift's and implementing examples.

3. Best practices for Web development and comparison of Web frameworks in the use of best practices

A best practice is the best way to achieve an outcome. It is well documented, used widely by the community, and continually evolving. Moreover, it promises quality, consistency, efficiency, and flexibility in engineering software systems [26]. In order to be considered a best practice, a method or tool must have some quantitative proof that it actually contributes to the improvement of quality, productivity, maintainability, or some other tangible factors [27].

The main purpose of this paper is to gather and present a list of the most relevant engineering practices that exist in the development community today by analyzing the best practices reported by others studies of Web frameworks such as 1) Books [18,19,21,28] and 2) Web-development community [29–32], and identifying those practices that enable three of the most important quality criteria for the success of Web applications. These criteria are 1) Reliability, 2) Usability [33], and 3) Security [34].

Table 1 introduces a list of best practices for Web applications development.

3.1. Best practices for lift-based Web application development

This section provides an analysis about how Lift supports the best practices summarized and listed in Table 1.

- **AJAX support**: Lift offers support for asynchronous client-server interaction.
- **Cloud computing**: Cloud foundry is the name of the new platform as a service (PaaS). It already supports Lift and Scala.
- **Comet support**: Lift has Comet support that allows a Web application to push messages from the server to client side using Web 2.0 features with little effort [7].
- **Custom Error Messages**: Lift provides a unified model for such messages that can be used for static pages as well as for AJAX and Comet calls.
- **Customization and Extensibility**: Lift offers more customization of this snippet than just emitting some XHTML. By specifying some prefixed attributes on the tag itself, attributes can be added to menu elements. Lift also supports modules e.g. PayPal, Facebook, Open Authorization (OAuth), and a module containing many jQuery widgets e.g. autocomplete, calendars, and progress bars, among others [80]. One common customization of widget would be to override the Check Cascading Style Sheets (CSS) used. To do this, Lift provides its own style.css file.
- **Debugging**: Lift applications can be easily debugged by using the Maven and SBT build tools, since they allow for the addition of all necessary dependencies into the Lift projects. The debugging process can be performed through the Scala console or an IDE, such as Eclipse or IntelliJ IDEA, in addition to the plug-in corresponding to each tool [81].
- **Documentation**: Scaladoc is a documentation generation system which specially reads formatted comments in Scala source code and generates compiled documentation. It is typically used to produce API documentation in the form of HTML Web pages.
- **Forms validation**: The forms are an important item on Web applications. Therefore it is necessary to validate data input that users provide. If the forms pass validation, an action is performed. If the forms do not pass validation, the user receives a message highlighting the fields that caused the validation problems. Then, the system gives the user an opportunity to fix them. Lift provides a single-screen input/validation mechanism called LiftScreen and a multi-page input/validation mechanism (with stateful next/previous buttons) called Wizard. Lift also provides forms validation by using Record or Mapper objects.
- **HTML5 support**: Lift supports parsing HTML5 input files and rendering HTML5 to the browser in addition to Lift's XHTML support. It offers features such as HTML 2D (canvas and Scalable Vector Graphics (SVG)), HTML 3D (WebGL), geolocation, audio, and video, among others.
- **Internationalization**: One of the best aspects about Lift is its flexible template and resource localization system. Lift supports a new type of resource bundle for I18N, covered in localization. The new resource bundles can be global and per-page
- **JavaScript-based frameworks support**: Lift provides techniques to simplify and abstract access to JavaScript on the client side.
- **ORM**: The ORMs included in Lift are Mapper and Record. Mapper is an ORM system for relational databases that lets query database and represents data in Scala objects. Record is a thin layer over a persistence mechanism to persist objects.
- **Parallel rendering**: Lift allows for the execution of multiple snippets in a parallel way during the rendering of a Web page.
- **Platform support**: Applications developed in Lift can run on platforms such as Windows, Linux, and OS X without modifying the source code.

Table 1
Best practices for Web development.

AJAX support
Description:
The development of interactive web sites requires using a combination of technologies, which involves time and effort in development. However, AJAX is a means to use existing standards in order to create interactive Web contents. It allows for the exchange of data with a server and the update of parts of a Web page without a full-page reload. Today, this option is quite popular to produce fancy visual effects and to provide advanced user interfaces [35].
Example
Movies selection on the Cinepolis® website [36].

Cloud computing
Description:
Cloud computing is widely used among large and small organizations, since it provides several benefits such as reduction of costs, universal access, up to date software, and flexibility. Cloud service providers allow for the deployment of applications in the cloud on different programming technologies. This enables users to choose the option that best meets their needs.
Cloud computing is a promising computing model that enables convenient and on-demand network to access a shared pool of configurable computing resources [37].
Example
Online storage services from Humyo® [38], SugarSync® [39], and SkyDrive® [40] websites.

Comet support
Description:
TA Web page is usually delivered to the client upon his/her own request. Updating data on the side requires the user to refresh or change the Web page, which consumes a great amount of both time and bandwidth. Comet is a solution allowing the server to send a message to the client when an event has occurred, without the client requesting it explicitly. The client can thus focus on anything else while expecting new data from the server [41].
Example
Chat applications from Yahoo® [42] and Microsoft® [43].

Custom Error Messages
Description:
Feedback is an important aspect to the user. An application must be able to notify errors, warn the user of potential problems, and notify the user when system status has changed. Two empirical studies with 77 and 90 participants have found evidence that the best way of presenting error messages is to provide the erroneous fields after users have completed the whole form, since users often ignore the messages displayed on the screen and continue completing the form as if nothing had happened [44].
Example
The creation of a new user account on the Liverpool® website [45].

Customization and Extensibility
Description:
Adding extended functionality, such as dynamic elements, to a Web application requires a great effort during the development process. However, adding functionality to the application with plugins or other options decreases both effort and development time.
Example
Netvibes® website [46].

Debugging
Description:
Reading through the code to find a bug is a difficult task for developers. Debugging can be of great help to reveal false assumptions the developer made about the state of the code.
Debugging is the task of eliminating errors, or bugs, from an application. While it may refer to any process to correct bugs, it generally means using a debugger to find and identify bugs in the code [47].
Example
"Hello" World in codeanywhere© [48].

Documentation
Description:
Documentation is a significant part of development and maintenance of Web applications. However, developers rarely document their code. For this reason, it can be a tough task for developers to make a change on the application [49].
Example
WordPress.org [50].

Forms validation
Description:
Users frequently fill Web forms with necessary information. However, they often make mistakes during this process. Thus, it is necessary to ensure that the user provides necessary and properly formatted information to successfully complete an operation. Forms validation is an important factor on Web development that solves this problem. The client is usually expected to enter different type data (phone number, name, ZIP-code, or e-mail address). The most frequent solution is to determine on the server side whether the submitted data has the required form, which is known as server-side input validation. If some data are invalid, the parts are presented once again along with suitable error messages [51].
Example
The online registration process on the InfoQ© website [52].

(*continued on next page*)

Table 1 (*continued*)

HTML5 support
Description:
Currently, the goal of developers is to create interactive websites and, in order to do so, they need to include fluid animations and videos, play music, and include Social Network features into the websites. So far, developers can choose to integrate these features with the help of tools such as Flash or Silverlight, Flex or JavaScript. However, these options are time consuming and make Web applications more complex. However, by using HTML5 it is possible to embed video, audio, high quality drawings, charts, animations, and other rich content without using any plugins or third party programs.
HTML5 is mostly about creating a declarative hypertext language in support of application like functionality. It is made to be dynamic and interactive to incorporate many of the aspects of AJAX into a holistic format focused on moving the Web to and application level construct [53].
Example
hootsuite® [54], twimbow® [55] websites.

Internationalization
Description:
The aim of businesses is to expand and participate in international trade. Therefore, having a website in different languages provides the opportunity for these businesses to reach an international audience. However, content, support for standards of each country, and the translation of the website's interface lead to intensive work for developers. Fortunately, internationalization makes this task easier. Internationalization is the design and development of a product, application, or document content that enables easy localization for target audiences that vary in culture, region, or language [56].
Example
CNN® news website [57].

JavaScript-based frameworks support
Description:
Incorporating JavaScript into a Web page allows for the improvement of the users experience by converting a static page into an interactive one. JavaScript code runs on the client side, i.e. that the code is executed on the user's processor instead of the Web server. This saves bandwidth and strain on the web server. It is widely used for Web programming and, increasingly, for general computing purposes. Improving the correctness, security, and performance of JavaScript-based applications has been the driving force for research in type systems, static analysis, and compiler techniques for this scripting language [58].
Example
Dell™ [59] and IBM® [60] websites using jQuery.

Object-Relational Mapping (ORM)
Description:
The traditional way of storing a data from the programming languages into the database involves great effort in terms of development. However, ORM reduces the complexity of the code that needs to be written. It allows developers to declaratively define the mapping between the object model and database schema and expresses database-access operations in terms of objects [61].
Example
Foursquare® website [62].

Parallel rendering
Description:
Web applications containing information of multiple external resources usually have a slow loading time. However, parallel rendering enables accessing multiple external resources in a parallel way. Snippets can be loaded and rendered in parallel in different threads. Parallel Page Rendering can really come in handy when there are simultaneously multiple high-computational snippets to render, like certain heavy database queries in a Customer Relationship Management (CRM) system [63].
Example
iGoogle [64].

Platform support
Description:
Before developers begin writing code, they need to consider the demands that will be placed on the hardware and the operating system (OS). It is pointless to invest a lot of time and money in configuration and coding only to find that the server's performance is poor because the platform is not suitable. Web frameworks are designed to work on a variety of platforms such as Windows, Linux, and Os X.
Example
etherPad© [65].

REST support
Description:
The development of Web services has involved technologies such as Web Services Description Language (WSDL), Universal Description, Discovery and Integration, and Simple Object Access Protocol (SOAP). However, nowadays, Representational State Transfer (REST) has emerged as a new approach to develop Web services. REST is an architectural style for distributed hypermedia system. REST-based Web services provide advantages such as: 1) requests and responses shorter than those of SOAP, since SOAP requires an enveloped XML-based format in every request and response; 2) lower bandwidth required to transport requests and responses if compared to SOAP 3) less memory and processing time required in order to process requests compared to those of SOAP [66].
Example
Twitter® [67] and Flickr® [68] websites.

Scaffolding
Description:
During Web development, developers usually carry out the four basic functions of persistent storage. Scaffolding is a great way to automatically generate basic applications that create, read, update, and delete objects (CRUD). At the same time, advantages are obtained, such as saving time and effort on the Web developed.
Example
Production websites as ValueClick, Inc [69].

Table 1 (*continued*)

Security
Description:
With the adoption of Web applications as a means to do business and deliver service, the industry now pays especial attention to the security of data shared through these applications. Attackers can use different ways to access information, such as cross-site scripting (XSS) and SQL injection. Open Web Application Security Project (OWASP) is a community that helps organizations to conceive, develop, operate, and maintain reliable applications. OWASP published a top 10 threats against Web applications [70]. Thanks to the security support provided by Web frameworks, developers can focus their efforts on developing other features instead of developing defense against the top-10 OWASP and other vulnerabilities.
Example
The protection of personal information on the ACM® [71], Liverpool® [45], C&A® [72] websites.

SiteMap
Description:
Designers frequently have problems with the global navigation system. For instance, there are usually too many tabs, everything is wedged into a tree, users cannot find things, and it takes too many clicks to access the appropriate screen. However, the automatic creation of menus (SiteMap) can help avoid these problems. SiteMap provides basic menu generation functionality, the grouping of menu items, nested menus, and access control mechanisms that not only deal with whether a menu item is visible or not, but also with whether the page it points to is accessible [7].
Example
StackMob® website [73].

Template framework
Description:
The design of both the structure and style of a Web application involves loss of time. However, using a template that provides these two features streamlines the development process of a Web application.
Templates are pieces of HyperText Markup Language (HTML)-based code common to a set of Web pages usually adopted by content providers to enhance the uniformity of layout and navigation of websites [74].
Example
Earthlink.net® website [75].

Testing
Description:
Programmers often spend more time and effort finding and fixing bugs than writing new code. However, software testing is an activity aimed at evaluating the quality of a program and also improving it by identifying defects and problems [76].
Example
"Hello World in Cloud9© [77].

Use actors
Description:
Processor manufacturers have turned towards multi-core processors, which are capable of doing multiple calculations in parallel. Hence, there has been an increasing interest in software engineering techniques in order to fully utilize the capabilities offered by these processors. The Actor model of concurrency offers a different and generally superior mechanism to do multithreaded and multicore coding in order to obtain maximum efficiency and avoid temporary blockages in Web applications. This is due to the fact that the single-thread programming method is used [78].
Example
Foursquare® website [62].

Use Lazy Loading
Description:
In a Web application, data retrieval of other services is sometimes slow. Thus, users may feel that the website is slow if the web page is deployed only after the data is fully processed.
Nevertheless, using Lazy Loading allows the page to render and defer only a small portion until the request is ready. It then pushes the complete content at a later date. This happens in order to provide a better user experience.
Example
Netvibes® website [46].

Use pattern matching
Description:
Designing patterns is a general reusable solution to a recurrent problem in software design. A design pattern is not a finished code, but rather a template for solving a problem; it can be used in many diverse situations.
Patterns help prevent subtle issues, improve code readability, and speed up the development process.
Pattern matching provides a powerful tool to declare business logic in a concise and maintainable way [78].
Example
Twitter® website [67].

Wiring
Description:
Web applications have many interdependent components; for example, there is the case of a shopping cart that contains items and quantities. Keeping track of all these dependencies is hard work. When it comes to updating the site, developers must remember the location of all of the items and how to update them.
Wiring provides a simple solution to manage complex dependencies on a single page and on multiple tabs. It enables to declare the relationships among cells (like a spreadsheet) and the user interface components associated with each cell [7].
Example
The online shopping cart from Forumsport® website [79].

- **REST support**: One of the key features of REST-based Web services is the explicit use of Hypertext Transfer Protocol (HTTP) methods. Lift provides support for REST-based Web services. RestHelper provides useful helper methods that make the code more concise, readable, and maintainable. Lift RestHelper extracts XML or JSON code from POST or PUT request.
- **Scaffolding**: Lift offers two forms of scaffolding, 1) Prototype traits for inheriting common functionality into the implementation, 2) CRUDify for generating CRUD-style interfaces for common use cases [8].
- **Security**: Lift-based Web applications are resistant to common vulnerabilities. SiteMap is a part of the security approach in Lift. It is a unified access control. The developer defines the access control rules for each page by using Scala code, if(User.LoggedIn) or if(User.superUser). These access control rules are applied before rendering any page. Lift offers advantages of access control, and helps resists the OWASP top 10 security vulnerabilities.
- **SiteMap**: The Lift SiteMap is not just a menu giving users access to important pages on a website, it is also a comprehensive representation of every Web page of a website and the access rules that govern whom can access what and when [63].
- **Template framework**: Lift uses a powerful template system based on the view-first approach, which allows for the modification of pages and the reuse of page components much simpler. This approach does not allow for business logic in the view. It starts with the eXtensible HyperText Markup Language (XHTML); then, the controller (Snippet in Lift) can fill in the data in the desired places [82].
- **Testing**: Lift provides many ways to make testing easier by using ScalaTest, JUnit, Mocking, or Selenium.
- **Use actors**: Lift uses actors and immutability as part of its implementation framework and encourages Web-application development based on events and messages. Comet support is one of the main areas in Lift that uses actors as a basic architectural unit. Actors represent a computing model based on asynchronous message-passing concurrency that does not restrict the message-arrival ordering.
- **Use Lazy Loading**: Lift supports lazy loading of a snippet. Lift lazy loading is based on Lift´s Comet support, which is grounded in long polling that enables to "push" content from the server to the browser asynchronously [7].
- **Use pattern matching**: Lift makes use of Scala's pattern matching to enable matching incoming HTTP requests, extracting values as part of the pattern matching process and returning the results.
- **Wiring**: It permits declaring relationships among the various elements on a Web page. When any of the precedent elements changes, the dependent items are redisplayed on the next HTTP response [63].

3.2. Comparison of Web frameworks in the use of best practices

This section compares certain Web frameworks based on their best practices. These frameworks are Struts, JSF, CakePHP, Grails, Ruby on Rails, Catalyst, and Django. They were selected because of their maturity level as well as their successful use in several projects [83–89], including well known websites built with them [90–93]. In addition, after extensive research, the frameworks were detected among the firsts of the ranking of Web frameworks. The rank relied on the developers' experience while working with them [94,95]. Developers claimed to have selected these frameworks because 1) they had a large and active developer community and 2) they possessed a very low learning curve.

Table 2 presents a comparison of Web frameworks (JSF, Ruby on Rails, Struts, CakePHP, Grails, Django, Catalyst and Lift). This comparison is based on the support of the best practices mentioned in previous sections (see Table 1). Table 2 illustrates that most of Web frameworks support features such as Internationalization, Forms Validation, AJAX support, and ORM, among others. However, some frameworks lack of features such as Actors, Lazy Loading, Comet, Pattern Matching, HTML5, SiteMap, Wiring and Parallel Rendering. Also, the comparative table demonstrates that Lift offers more features than other frameworks, and, provides features such as pattern matching and actors thanks to the Scala programming language. This means that Lift obtains the benefits from the functional programming e.g. concurrency in a transparent way, higher-order functions, among others [96]. Lift and Grails frameworks have the actors feature. Lift is designed to be a high-performance and scalable Web framework by leveraging Scala actors to support more concurrent requests than possible with a thread-per-request server. On the other hand, Grails provides actors through of GPars. It is a Groovy concurrency library.

All Web frameworks contain the AJAX support feature, but merely JSF, Lift, Rails, Django, and Catalyst additionally provide Comet support. This represents a greater advantage since these two approaches (AJAX and Comet) improve the user's experience through dynamic Web pages. Comet applications can deliver data to the client at any time, not only as a response to user input. The data is delivered over a single previously opened connection. This approach significantly reduces the latency for data delivery.

JSF, Lift, Grails, Django, and Catalyst provide HTML5 support that allows users to develop rich and dynamic Web applications. HTML5 is rapidly winning over the Web, providing dynamic and interactive features to develop applications with little effort. In fact, according to a survey of more than 5000 developers and technology executives by Kendo UI, a division of Telerik, a great number of developers from the entire globe utilize HTML5. Developers in North America, South America, Africa, and Australia use HTML5 at a rate of 70%, 61%, 50%, and 60% respectively [97]. Each framework provides this feature differently. JSF introduces the so-called pass-through attributes to support new and modified attributes defined in HTML5.

Table 2
Summary of the best practices supported for Web development in JSF, Ruby on Rails, Struts, CakePHP, Grails, Lift, Django, and Catalyst.

Best practice	JSF	Ruby on Rails	Struts	CakePHP	Lift	Grails	Django	Catalyst
AJAX support	Yes	Yes (Prototype, script.aculo.us, jQuery, among others)	Yes	Yes (Prototype, script.aculo.us, jQuery, MooTools, among others)	Yes	Yes (jQuery, prototype, Dojo, YUI, Mootools, among others)	Yes (jQuery, prototype, Dojo, Mootools, among others)	Yes (jQuery, ExtJS, Dojo, YUI, Mootools, among others)
Cloud computing	Yes (Oracle Public Cloud, Oracle WebLogic Server and Google App Engine)	Yes (Amazon EC2, Linode, Rackspace and Heroku)	Yes (Jelastic and Google App Engine)	Yes (Amazon EC2 and Rackspace)	Yes (CloudFoundry)	Yes (CloudFoundry, Google App Engine, Amazon EC2 and Heroku)	Yes (dotCloud, Google App Engine and Amazon EC2)	Yes (Amazon EC2)
Comet support	Yes (Icefaces)	No	No	No	Yes (CometActor)	Yes (Atmosphere or CometD plugins)	Yes (Orbited)	Yes (Twiggy)
Custom Error Messages	Yes (File properties)	Yes (File yml)	Yes (File properties)	Yes (Model-message)	Yes (Snippet-s.notice, s.error)	Yes (File properties)	Yes (model-validationError)	Yes (Catalyst::Action::RenderView::ErrorHandler)
Customization and Extensibility	Yes (PrimeFaces, RichFaces, ICEfaces)	Yes (Plugins e.g. LessCSS, Authlogic, among others)	Yes (Plugins. e.g. JSCalendar, Guice, among others)	Yes (Plugins e.g. Mandrill, CakeDC, among others)	Yes (Modules e.g. PayPal, Widgets, among others)	Yes (Utility e.g. iCalendar, Smartionary, among others)	Yes (Django-Utils e.g. safestring, translation, among others)	Yes (Plugins-Catalyst::Plugin::AutoCRUD, among others)
Debugging	Yes (ui:debug)	Yes (debug, to_yaml and inspect)	Yes (Struts 2 configuration plugin and debugging interceptor)	Yes (DebugKit plugin)	Yes (SBT and Maven)	Yes (X-Grails-Resources-Original-Src Header)	Yes (Django Debug Toolbar)	Yes (Komodo)
Documentation	Yes (JavaDoc)	Yes (Rdoc)	Yes (JavaDoc)	Yes (phpdomain)	Yes (ScalaDoc)	Yes (grails doc)	Yes (Sphinx)	Yes (Plain Old Documentation, POD)
Forms validation	Yes (JSF standard validators and Bean validation)	Yes (ActiveRecord validations)	Yes (ActionForms)	Yes (FormHelper)	Yes (LiftScreen and Wizard)	Yes (Spring's Validator)	Yes (Django Form)	Yes (HTML::Form-Handler and HTML::Form-Validator)
HTML5 support	Yes (Pass-through attributes)	No	No	No	Yes (Html5Properties)	Yes (Modernizr)	Yes (HTML5 Boilerplate, H5BP)	Yes
Internationalization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
JavaScript frameworks support	Yes (Dojo, ExtJS, jQuery, among others)	Yes (jQuery, script.aculo.us, Dojo, among others)	Yes (jQuery, Dojo, ExtJS, among others)	Yes (ExtJS, jQuery, Dojo, among others)	Yes (jQuery, script.aculo.us, ExtJS, among others)	Yes (jQuery, Dojo, script.aculo.us, among others)	Yes (jQuery, ExtJS, Dojo, among others)	Yes (jQuery, Dojo, ExtJS, among others)
ORM(Object Relational Mapping)	Yes	Yes (ActiveRecord)	Yes	Yes (ActiveRecord and Datamapper patterns)	Yes (Mapper and Record)	Yes (GORM)	Yes (Django ORM)	Yes (DBIx::Class and Rose::DB::Object)

(*continued on next page*)

Table 2 (*continued*)

Best practice	JSF	Ruby on Rails	Struts	CakePHP	Lift	Grails	Django	Catalyst
Parallel rendering	No	No	No	No	Yes	No	No	No
Platform support	-Windows (Java Developers Kit (JDK))	-Windows (Rails installer)	-Windows (JDK)	-Windows (PHP 5.2.8 or greater, HTTP Server)	-Windows (Scala and Java Runtime Environment (JRE))	-Windows (JDK and Grails libraries)	-Windows (Python, Setuptools and PIP)	-Windows (perl 5.8.6 or higher, Catalyst::Runtime and Catalyst::Devel)
	-Linux (JDK)	-Linux (JDK)	-Linux (JDK)	-Linux (PHP 5.2.8 or greater, HTTP Server)	-Linux (Scala and JRE)	-Linux (JDK and Grails extensions)	-Linux (Python and PIP)	-Linux (perl 5.8.6 or higher, Catalyst::Runtime and Catalyst::Devel)
	-Os X (JDK)	-Os X (JDK)	-Os X (JDK)	-Os X (PHP 5.2.8 or greater, HTTP Server)	-Os X (Scala and JRE)	-Os X (JDK and Grails libraries)	-Os X (Python and PIP)	-Os X (perl 5.8.6 or higher, Catalyst::Runtime and Catalyst::Devel)
REST Support	Yes	Yes	Yes	Yes (File config-mapResources)	Yes (RestHelper)	Yes	Yes (Django REST framework)	Yes (Catalyst::Controller::REST)
Scaffolding	No	Yes	No	Yes	Yes	Yes	Yes	Yes
Security	Yes	Yes	Yes	Yes	Yes	Yes (Spring security and Shiro)	Yes	Yes
SiteMap or Automatic menu creation	No	No	No	No	Yes (SiteMap)	No	Yes (NAV Menu currently being evaluated)	Yes (MenuGrinder)
Template framework	Yes (Facelets)	Yes	Yes	Yes (Views, elements and layouts)	Yes	Yes (Layout and SiteMesh)	Yes (Django template language)	Yes (Template Toolkit, HTML::Template, HTML::Mason, PHP and any extant Perl template engine)
Testing	Yes (JSFUnit)	Yes (Rspec)	Yes (StrutsTestCase)	Yes (PHPUnit, Fixtures and Mocking)	Yes (ScalaTest, JUnit, Mocking and Selenium)	Yes (GMock and EasyMock)	Yes (Unit test, doctest, nose)	Yes (Supports Perl testing standards)
Use actors	No	No	No	No	Yes (LiftActor)	Yes (GPars)	No	No
Use Lazy loading as part core framework	No	No	No	No	Yes	No	No	No
Use pattern matching	No	Yes	No	No	Yes	No	Yes	Yes
Use Wiring	No	No	No	No	Yes	No	No	No

Lift 2.6 uses the HTML5 parser by default. However, for previous versions of Lift, XHTML is the option for output and input parsing, but by invoking Html5Properties in Boot.scala, the developer can set Lift to full HTML5 support. Grails provides the library Modernizr that helps developers build HTML5 websites. Django provides a django-html5-boilerplate framework that includes HTML5 into Django project. Catalyst also provides a full HTML5 support.

Django, Lift, and Catalyst have the SiteMap feature. It allows for the creation of hierarchical menus in a Web application and the easy definition of navigation rules. Also, it is a mechanism that helps develop applications more safely. Django provides this feature trough of NAV Menu; however, it is currently being evaluated. Moreover, Lift offers a wide range of functionalities to let control site navigation and access trough of SiteMap. Catalyst also offers MenuGrinder, a tool for managing dynamic website menus.

Finally, only Lift provides features such as Lazy Loading, Parallel Rendering and Wiring. These features allow for a better user experience during the rendering of a Web page. David Pollak with his experience in other Web frameworks, claims that this is one of the aspects that distinguish Lift from other Web frameworks, and that these features are crucial to build and maintain interactive and secure websites [63].

Because frameworks are rapidly evolving and adopting features from newer ones, it is difficult to select only one framework to develop Web applications. Nevertheless, authors of this research paper consider that the comparison made above offers developers a way to select the Web framework that best suits their needs.

In the next section, we present a set of Lift-based Web applications that use some of the best practices discussed above.

4. Applying best practices on lift-based Web applications

As a proof of concept, several Web applications were implemented using both best practices present only in Lift – such as Wiring and parallel rendering – and best practices that lacked in most of the other frameworks – such as SiteMap, Actors, HTML5, Comet, and Lazy Loading.

4.1. SiteMap

It allows for the creation of a user menu with options (login, register a new user, and recovery password). These options are added automatically by the “AddUserMenusAfter” function (line 1). If the user logged in, the menu shows an extra option (“Courses”) (line 4).

```
1  def sitemap() = SiteMap(Menu("Home?") / "index" >> User.AddUserMenusAfter,
2  Menu("Courses", "Courses") / "Courses" ) >> IfLoggedIn
```

Fig. 1 presents a Web application with a menu automatically created and easily customized. SiteMap provides access rules to define who can and when this person can access the website.

This screenshot shows the feature SiteMap on Lift-based Web application development. The use of SiteMap allows developers to define the access control rules for each Web page, avoiding the process of updating the XML-based security rules when the Uniform Resource Locator (URLs) or the methods that implement the access control are changed. SiteMap allows for the development of more secure Web applications.

4.2. HTML5 support

Lift templates are based on XHTML by default, although it is advisable to use HTML5 support since JavaScript libraries such as Google Maps do not work on XHTML pages.

The global object that holds all of Lift’s configuration (LiftRules) uses the HTML5 parser by default, which is why setting the parser in the Boot.scala file is not necessary. It is a new feature of Lift 2.6.

The following example shows the use of HTML5 video tag, which allows for the visualization of videos in different formats.

```
1  <!DOCTYPE html>
2  <video width="300" heigth="140" controls>
3  <source src="videos/news.mp4" type="video/mp4" codecs="avc1.42E01E, mp4a.40.2">
4  </video>
```

Fig. 2 presents a Web application with HTML5 support. This is a news Web application, which supports different languages and shows videos of different formats. It shows the functionality of the <video> tag of HTML5 that is used to display a video in a HTML-based document. The screenshot shows another feature of HTML5, the geolocation API that allows users to enjoy the benefits of various location-aware services.

Fig. 1. Use of SiteMap on a Lift-based Web application.

Fig. 2. HTML5 support in a news Lift-based Web application.

HTML5 facilitates the development of interactive Web applications with features that traditional RIA (Rich Internet Applications) frameworks offer. These features are HTML2D, HTML3D, drag and drop, audio, and video, among others. Therefore, HTML5 enables to easily develop Web applications with Web 2.0 features.

4.3. Use actors

Lift simplifies the use of actors by providing the CometActor feature. The class CometActor must be located in a subpackage, configured through LiftRules.addToPackages ("com.myexample") in the class Boot.scala (line 4), and Comet actors are located in com.myexample.comet package.

```
1  class Boot {
2      def boot {
3      ....
4      LiftRules.addToPackages("com.myexample")
5      }
6  }
```

Next, the subclass Message is presented, which extends the class CometActor that is located in com.myexample.comet package. Lift CometActor feature is an extension of Scala Actor feature, which makes an actor easier to use, as a part of a Comet application. An actor is similar to a light-weighted thread, but without shared memory. Thus, synchronization and blocking are never needed.

```
1  class Message extends CometActor{
2      ....
3      ....
4  }
```

Finally, the class Message, located in the Comet subpackage, is invoked in the HTML-based document (line 3).

```
1  <lift:surround with="default" at="content">
2      <center><b> Message s</b></center><br>
3      <div lift="comet?type= Message ">
4      ....
5      </div>
6  </lift:surround>
```

Fig. 3 introduces a Web application by using actors in the development of Lift-based Web application. The figure presents a movies Web application in which the users can type comments about any movie, and these comments are automatically displayed in the browser.

The use of actors on Web applications offers advantages such as unifying threads and events (being efficient and scalable) and automatically mapping to multiple Java Virtual Machine (JVM) threads with the purpose of leveraging multi-core processors.

4.4. Comet

This section presents an example of the chat feature in a Web page. It is an example of asynchronous message passing in which the time, name, and message are showed with bullet points (lines 6–8), making reference to the class ChatLift.scala (line 2).

```
1  <span style="text-align: center"><b>Chat-ITO</b></span><br>
2  <div class="lift:comet?type=ChatLift;ul_id=side_ul_id;li_id=side_li_id">
3      Hi <span name="chat_name">Name:</span>
4      <ul id="side_ul_id">
5          <li id="side_li_id">
6              <span name="when">hour </span>
7              <span name="who">name:</span>
8              <span name="body">Message:</span>
9          </li>
10     </ul>
11 </div>
```

The next code shows a function displaying the messages that the user enters in a list control. It shows time, user name, and message.

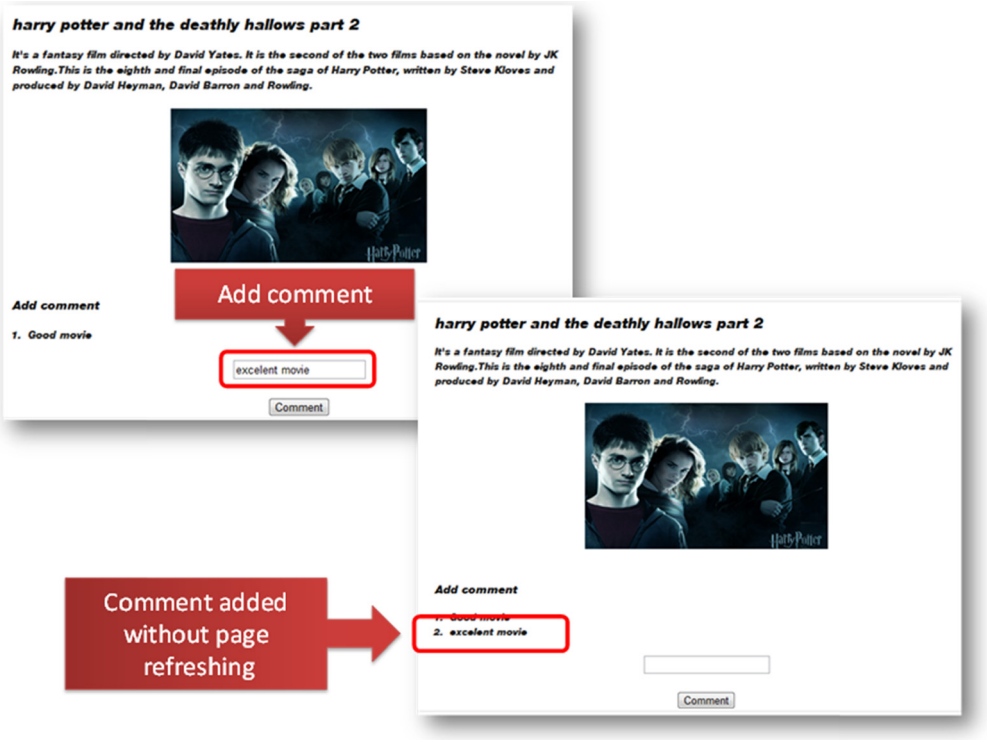

Fig. 3. Use of actors in a Lift-based Web application.

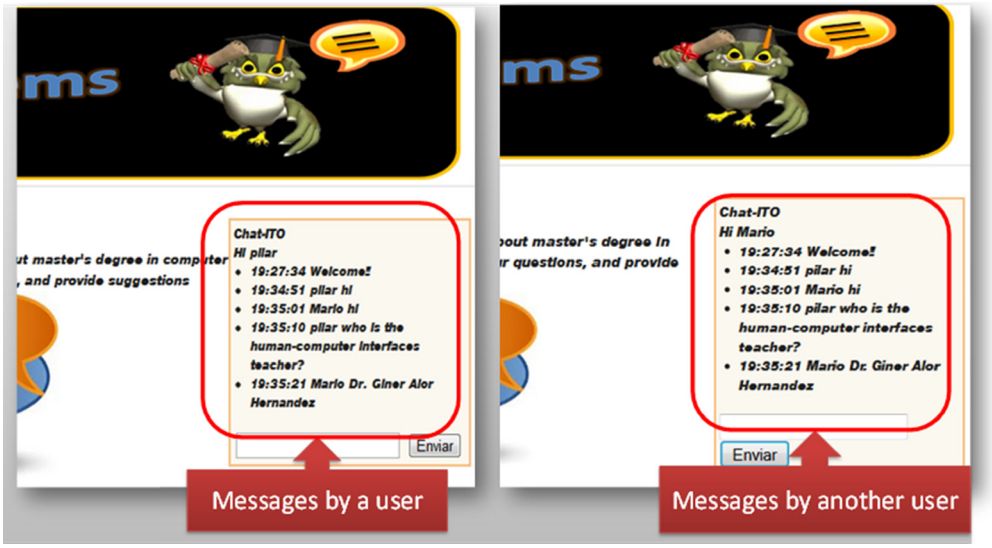

Fig. 4. Comet support in a Lift-based Web chat application.

```
1  private def line(l: ChatLine) = {
2      ("name=when" #> hourFormat(l.when) & "name=who" #> l.user & "name=body" #>
3  l.msg)(li)
4  }
```

Fig. 4 presents a Web application with Comet support. This is a Web chat application that allows students to share ideas or ask questions in real time.

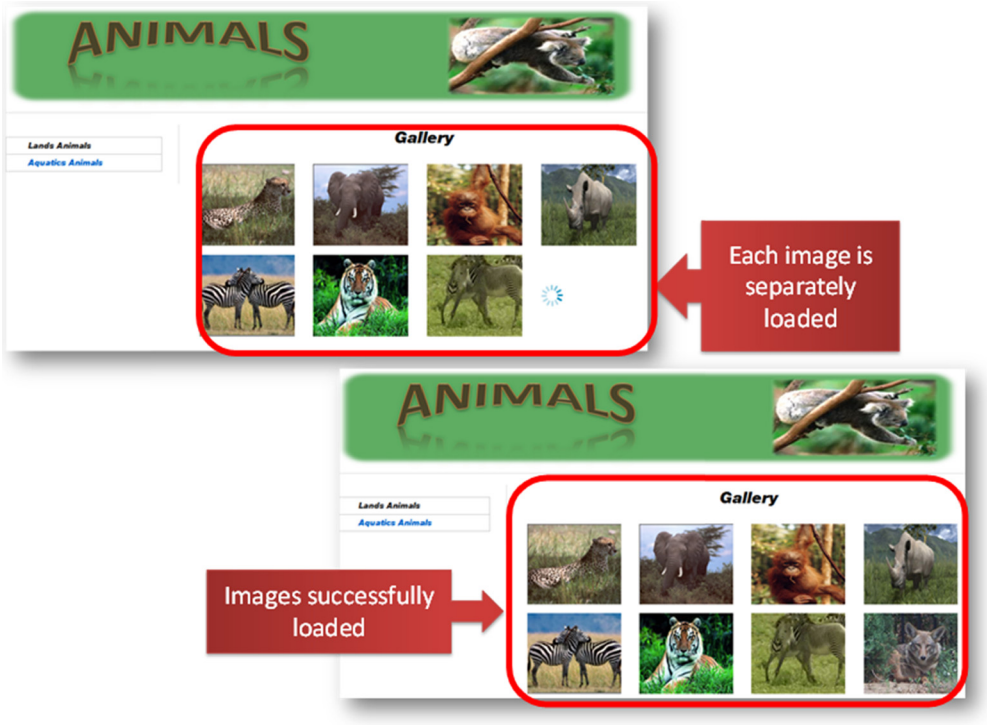

Fig. 5. Lazy Loading in an animal's gallery application.

Lift provides support functions and classes to simplify the development of applications that utilize techniques such as Comet, in which the server sends data to the browser only when IT is needed. It this decreases the latency and obtains asynchronous communication. Comet support allows for the development of more intuitive Web applications.

4.5. Lazy Loading

An example of the use of Lazy Loading is presented in this section. The example enables to render parts of a Web page asynchronously.

```
1  <div data-lift="lazy-load">
2      <div class="lift:Snippet1"></div>
3  </div>
```

Fig. 5 introduces a Web application with Lazy Loading. This Web application shows an image gallery. In the spaces where timeout for showing an image is achieved and the image has not been displayed yet, Lift provides a mechanism for retrying the request until it is successfully achieved.

4.6. Wiring

This example shows three items (subtotal, tax, and total) in a shopping cart. To add or remove a product such elements are modified without reloading the page.

```
1  def subtotal(ob: NodeSeq) = WiringUI.asText(ob, Info.subtotal)
2  def tax(ob: NodeSeq) = WiringUI.asText(ob, Info.tax, JqWiringSupport.fade)
3  def total(ob: NodeSeq) = WiringUI.asText(ob, Info.total, JqWiringSupport.fade)
```

Fig. 6 stands for a Web application with the Wiring feature. This Web application is for an online clothing store in which the users can add items to the shopping cart. When the items are added or removed, the shopping cart is updated. The use of this feature provides a solution for the management of complex dependencies in a single Web page, as in this example.

Fig. 6. Use of Wiring in an online clothing store.

Wiring enables to develop very complex inter-relationships with the elements on the screen. When one is updated all the dependent elements are automatically updated. Wiring helps save time and effort spent and allows for the easy maintenance of the website, since the administrator does not have to know all the dependencies.

4.7. Parallel rendering

Lift enables to run multiple Snippets simultaneously to process a Web page. This is a great use if external Web servers need to be queried during the presentation of a Web page. This example shows how the execution of a thread is stopped for ten seconds (line 3) by sending data to the browser, which in this case are text and images (lines 6–11).

```
1  class Example {
2      def render = {
3          Thread.sleep(10 seconds)
4          ".ad" #> Thread.currentThread.getName
5          <div>
6          <a href="http://televisadeportes.esmas.com/otrosdeportes/123165/doblones-columna-
7          carlos-yarza-dic-15"> Doblones, Columna Carlos Yarza Dic 15</a>
8          <br/>
9          <img src="/images/dep1.jpg" height="40" width="40" />
10         Enough!
11         <br/>
12         </div>
13         </div> } }
```

Next, to specify that a snippet be processed in a parallel way, it is necessary to set true the value of the attribute ”parallel” (line 1).

```
1  <div class="lift:Example?parallel=true">
2        server 1: <span class="ad">result</span>
3  </div>
```

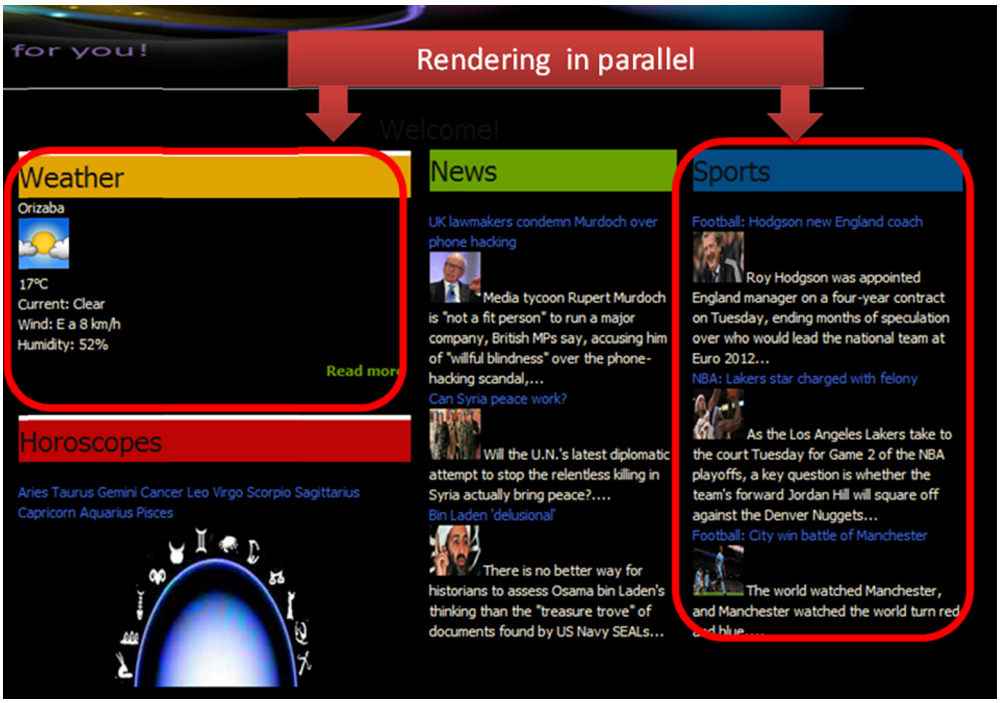

Fig. 7. Use of parallel rendering in a multiservice Web application.

Fig. 7 presents a Web application using of parallel rendering. The application allows for the registration of users. Users can view different services such as weather, horoscope, news, and sports, among others.

Parallel rendering enables to query many external servers while rendering a page. Therefore, users have a better experience to view the Web application.

5. Conclusions and future directions

This paper provides a comparison of eight Web frameworks in terms of their best practices. The use of best practices allows for the development of better and more efficient Web applications. With the use of the best practices analyzed in this work, Web applications were interactively, intuitively, and safely developed, improving the development effort and reducing development time. Therefore, best practices are essential for the software engineering community, since they help decrease errors in the implementation phase. The analysis presented in this work shows that Lift offers more features to develop Web applications than other frameworks. However, it must also be mentioned that the Lift community has currently reported bugs to be fixed and suggested improvements to be included in future versions of the framework. Some features to be improved are: 1) Adding methods to MongoRecord that catch exceptions and return a Box, 2) Fixing MongoListField's setFromJValue to properly handle special mongo data types, 3) Adding a method to suppress warnings when using the BaseResponse class in Lift tests, 4) Adding the ability to get a forced version of a request body as JSON or XML, regardless of whether the specified Content-Type of the request was correct.

As future directions, the authors of this paper consider that it is suitable to analyze new best practices such as CSS selectors, caching, and performance for the different Web frameworks presented in this work. Also, future research will include Lift comparison with other frameworks such as Play and Yii in order to provide a comprehensive analysis. This analysis will allow developers to choose the Web framework that best meets their requirements or the Web framework that provides the best and greatest number of advantages for a given project.

Acknowledgements

This work was supported by the Tecnológico Nacional de México. Additionally, the work was sponsored by the National Council of Science and Technology, Mexico (CONACYT) and the Public Education Secretary (SEP) through PROMEP. Alejandro Rodríguez González's work is supported by Isaac Peral Programme (UPM) and by UNIR Research Support Strategy 2013–2015, under the CYBERSECURITIS.es Research Group [http://research.unir.net]. The Spanish Ministry of Industry, Tourism, and Commerce supported this research paper under the project TRAZAMED (IPT-090000-2010-07), and the Spanish Ministry of Science and Innovation favored it under the project FLORA (TIN2011-27405).

References

[1] C.U. Smith, L.G. Williams, Best practices for software performance engineering, Presented at the Int. CMG Conference, 2003, pp. 83–92.
[2] R.A. Santelices, M. Nussbaum, A framework for the development of videogames, Softw. Pract. Exp. 31 (11) (Sep. 2001) 1091–1107.
[3] D. Pollak, David Pollak on Lift framework and Scala.
[4] P. Chiusano, R. Bjarnason, Functional Programming in Scala, 1st edition, Manning Publications, S.l., 2014.
[5] D. Chen-Becker, M. Danciu, D. Pollak, T. Weir, Starting with Lift, 2009.
[6] D. Pollak, Simply Lift, 2010.
[7] D. Chen-Becker, M. Danciu, T. Weir, The Definitive Guide to Lift: A Scala-Based Web Framework, new edition, firstPress, Berkeley, CA, New York, 2009.
[8] T. Perrett, Lift in Action: The Simply Functional Web Framework for Scala, Manning Publications, Shelter Island, NY, 2011.
[9] R. Dallaway, Lift Cookbook, 1st edition, O'Reilly Media, Sebastopol, Calif., 2013.
[10] T. Uhlmann, Instant Lift Web applications how-to.
[11] T. Fiedler, C. Knabe, Entwicklung von Web-Applikationen mit Lift und Scala: Einführung anhand einer durchgehenden Beispielapplikation, 1 Auflage, Shaker, Aachen, 2011.
[12] D. Pollak, S. Vinoski, A chat application in Lift, IEEE Internet Comput. 14 (3) (2010) 88–91.
[13] D. Wampler, Scala Web frameworks: looking beyond Lift, IEEE Internet Comput. 15 (5) (Sep. 2011) 87–94.
[14] D.-H. Hu, Y. Xue-Jun, H. Fei, Designing and implementation of agile framework based on Lift, in: 2010 2nd International Conference on Information Science and Engineering, ICISE, 2010, pp. 1–3.
[15] G.A. Stout, Testing a website: best practices, Aug-2001, pp. 3–14.
[16] O'Reilly, Principles and best practices, Web 2.0 report, O'Reilly Media, Incorporated, 2007.
[17] R. Hightower, Jakarta Struts Live, SourceBeat, LLC, Highlands Ranch, Colorado, 2004.
[18] R. Eckstein, J.S. Perry, O. Authors, Java Enterprise Best Practices, 1st edition, O'Reilly Media, Beijing, Sebastopol, CA, 2002.
[19] N. Ford, Art of Java Web Development: Struts, Tapestry, Commons, Velocity, Junit, Axis, Cocoon, Internetbeans, Webwork: Frameworks and Practices, Manning Pubn, Greenwich, CT, 2003.
[20] D. Golding, Beginning CakePHP: From Novice to Professional, 1st edition, Apress, Berkeley, CA, New York, 2008, distributed by Springer-Verlag.
[21] S. Shenoy, N. Mallya, Struts Survival Guide: Basics to Best Practices, Objectsource Llc, Austin, 2004.
[22] V. Rao, Scala Server Faces, Masters Proj., Jan. 2010.
[23] H. Vikas, Overview of Lift Web framework, Presented at the 4th IndicThreads.com Conference on Java, Pune, India, 2009.
[24] D. Ghosh, S. Vinoski, Scala and Lift functional recipes for the Web, IEEE Internet Comput. 13 (3) (May 2009) 88–92.
[25] Build Comet applications using Scala, Lift, and jQuery, 24-Mar-2009 [online]. Available: https://www.ibm.com/developerworks/ajax/tutorials/wa-aj-comet/, accessed: 29-Apr-2014.
[26] M.H. Dodani, The best practice promise and myth, J. Object Technol. 2 (4) (2003) 65–68.
[27] C. Jones, Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies, 1st edition, McGraw-Hill Osborne Media, 2009.
[28] C. Pytel, T. Saleh, Rails AntiPatterns: Best Practice Ruby on Rails Refactoring, 1st edition, Addison-Wesley Professional, 2010.
[29] D. Geary, JSF 2 fu: best practices for composite components, 11-Jan-2011 [online]. Available: http://www.ibm.com/developerworks/web/library/j-jsf2fu0111/index.html?ca=dat.
[30] Testing in Django (part 1) – best practices and examples, Real Python! Blog, 08-May-2013 [online]. Available: http://www.realpython.com/blog/python/testing-in-django-part-1-best-practices-and-examples/#.U3oVWvl_srU.
[31] M. Gray, Catalyst best practices, Catalyst Web Framework, 2010 [online]. Available: http://wiki.catalystframework.org/wiki/best_practices.view.
[32] P. Agarwal, Struts best practices, JavaWorld, 13-Nov-2004 [online]. Available: http://www.javaworld.com/article/2072942/java-web-development/struts-best-practices.html.
[33] J. Nielsen, Usability Engineering, Elsevier, 1994.
[34] J. Offutt, Quality attributes of Web software applications, IEEE Softw. 19 (2) (2002).
[35] L.M. Álvarez-Sabucedo, L.E. Anido-Rifón, J.M. Santos-Gago, Reusing Web contents: a DOM approach, Softw. Pract. Exp. 39 (3) (Mar. 2009) 299–314.
[36] Cinépolis, 2010 [online]. Available: http://www.cinepolis.com/index.aspx.
[37] K. Yang, X. Jia, Data storage auditing service in cloud computing: challenges, methods and opportunities, World Wide Web 15 (4) (Jul. 2012) 409–428.
[38] Humyo [online]. Available: https://www.humyo.com/.
[39] SugarSynck [online]. Available: https://www.sugarsync.com/.
[40] SkypeDrive [online]. Available: https://skydrive.live.com/.
[41] E. Bozdag, A. Meshah, A. Van Deursen, Performance testing of data delivery techniques for AJAX applications, J. Web Eng. 8 (4) (Dec. 2009) 287–315.
[42] Yahoo [online]. Available: http://mx.yahoo.com/.
[43] Microsoft [online]. Available: http://www.microsoft.com.
[44] J.A. Bargas-Avila, G. Oberholzer, P. Schmutz, M. de Vito, K. Opwis, Usable error message presentation in the World Wide Web: do not show errors right away, Interact. Comput. 19 (3) (May 2007) 330–341.
[45] Liverpool, 2014 [online]. Available: http://www.liverpool.com.mx/.
[46] Netvibes [online]. Available: http://www.netvibes.com/.
[47] A. Horovitz, K. Kim, J. LaMarche, D. Mark, Unit testing, debugging, and instruments, in: More iOS6 Development, Apress, 2013, pp. 481–510.
[48] codeanywhere [online]. Available: https://codeanywhere.com/.
[49] N.J. Kipyegen, W.P. Korir, K. Njoro, Importance of software documentation, Int. J. Comput. Sci. 10 (5) (Sep. 2013) 223–228.
[50] WordPress [online]. Available: https://codex.wordpress.org/Main_Page.
[51] C. Brabrand, A. Møller, M. Ricky, M.I. Schwartzbach, PowerForms: declarative client-side form field validation, World Wide Web 3 (4) (Dec. 2000) 205–214.
[52] InfoQ [online]. Available: http://www.infoq.com/.
[53] S. Harper, A.Q. Chen, Web accessibility guidelines: a lesson from the evolving Web, World Wide Web 15 (1) (Jan. 2012) 61–88.
[54] Hootsuite [online]. Available: http://hootsuite.com/.
[55] I. Ricci, Twimbow, blog [online]. Available: http://www.twimbow.com/.
[56] Internationalization and localization markup requirements, W3C, 18-May-2006 [online]. Available: www.w3.org/TR/itsreq/.
[57] CNN, 2014 [online]. Available: http://edition.cnn.com/.
[58] G. Richards, S. Lebresne, B. Burg, J. Vitek, An analysis of the dynamic behavior of JavaScript programs, in: Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation, New York, NY, USA, 2010, pp. 1–12.
[59] Dell [online]. Available: http://www.dell.com.mx/.
[60] IBM [online]. Available: http://www.ibm.com.
[61] C. Richardson, ORM in dynamic languages, Queue 6 (3) (May 2008) 28–37.
[62] Foursquare [online]. Available: https://es.foursquare.com/.

[63] D. Pollak, Lift: the best seven reason, Presented at the GeeCON, 16-May-2011.
[64] iGoogle, iGoogle portal [online]. Available: http://www.google.com.mx/ig.
[65] M. Klehr, Etherpad [online]. Available: http://etherpad.org/#download.
[66] M.A. Paredes-Valverde, G. Alor-Hernández, A. Rodríguez-González, R. Valencia-García, E. Jiménez-Domingo, A systematic review of tools, languages, and methodologies for mashup development, Softw. Pract. Exp. (2013), http://dx.doi.org/10.1002/spe.2233.
[67] J. Dorsey, Twitter [online]. Available: http://twitter.com/.
[68] G. Couturier, Flickr [online]. Available: http://www.flickr.com/.
[69] ValueClick [online]. Available: http://www.conversantmedia.com/.
[70] OWASP the open Web application security project.
[71] ACM, Association for computing machinery, 2014 [online]. Available: http://www.acm.org/.
[72] C&A [online]. Available: http://www.cyamoda.com/.
[73] StackMob [online]. Available: http://stackmob.com/.
[74] K. Vieira, A.L. Costa Carvalho, K. Berlt, E.S. Moura, A.S. Silva, J. Freire, On finding templates on Web collections, World Wide Web 12 (2) (Jun. 2009) 171–211.
[75] EarthLink, 2014 [online]. Available: http://www.earthlink.net/.
[76] S.M. Quadri, S.U. Farooq, Software testing – goals, principles, and limitations, Int. J. Comput. Appl. 6 (9) (Sep. 2010) 7–10.
[77] Cloud9 [online]. Available: https://c9.io/.
[78] D. Pollak, Beginning Scala, Apress, 2009.
[79] Forumsport [online]. Available: http://www.forumsport.com.
[80] R. Dallaway, Modules, Assembla Lift, 2014 [online]. Available: https://www.assembla.com/wiki/show/liftweb/Modules.
[81] H. Seeberger, Debugging [online]. Available: https://www.assembla.com/wiki/show/liftweb/Debugging.
[82] View first, Assembla Lift, 2012 [online]. Available: https://www.assembla.com/wiki/show/liftweb/View_First.
[83] W. Kehe, J. Yajing, H. Xin, J. Yongjun, Design and implementation of fire administrative management system based on JSF and EJB3.0, in: 2013 Fifth International Conference on Computational and Information Sciences, ICCIS, 2013, pp. 555–558.
[84] J.-W. Wang, M. Wang, Research and implementation of information management system of oil field based on Struts framework, Comput. Eng. Des. 13 (2009) 059.
[85] M. Biermann, A simple versatile solution for collecting multidimensional clinical data based on the CakePHP Web application framework, Comput. Methods Programs Biomed. 114 (1) (Apr. 2014) 70–79.
[86] P. Palla, G. Frau, L. Vargiu, P. Rodriguez-Tomé, QTREDS: a Ruby on Rails-based platform for omics laboratories, BMC Bioinform. 15 (1) (Jan. 2014) 1–11.
[87] M.P. Waller, T. Dresselhaus, J. Yang, JACOB: an enterprise framework for computational chemistry, J. Comput. Chem. 34 (16) (Jun. 2013) 1420–1428.
[88] W. Bosl, J. Mandel, M. Jonikas, R.B. Ramoni, I.S. Kohane, K.D. Mandl, Scalable decision support at the point of care: a substitutable electronic health record app for monitoring medication adherence, Interact. J. Med. Res. 2 (2) (Jul. 2013).
[89] N.S.B. Miyoshi, D.G. Pinheiro, W.A. Silva, J.C. Felipe, Computational framework to support integration of biomolecular and clinical data within a translational approach, BMC Bioinform. 14 (1) (Jun. 2013) 180.
[90] What powers Instagram: hundreds of instances, dozens of technologies [online]. Available: http://instagram-engineering.tumblr.com/post/13649370142/what-powers-instagram-hundreds-of-instances-dozens-of.
[91] Success stories and sites using Grails, Grails [online]. Available: https://grails.org/Success+Stories.
[92] Examples of websites built using Struts, Struts Wiki.
[93] At a glance: Hulu hits Rails Conf 2012, Hulu Tech Blog.
[94] Top 20 Web frameworks for the JVM [online]. Available: http://www.infoq.com/research/jvm-web-frameworks.
[95] L. Lancor, S. Katha, Analyzing PHP frameworks for use in a project-based software engineering course, in: Proceeding of the 44th ACM Technical Symposium on Computer Science Education, New York, NY, USA, 2013, pp. 519–524.
[96] D. Wampler, A. Payne, Programming Scala: Scalability = Functional Programming + Objects, O'Reilly Media, Inc., 2009.
[97] Brandon Gaille, 9 developer statistics and trends on HTML5, BrandonGaille.com.