Webformer: A Rapid Application Development Toolkit for Writing Ajax Web Form Applications

David W.L. Cheung[1,2], Thomas Y.T. Lee[1,2], and Patrick K.C. Yee[1,2]

[1] Department of Computer Science, University of Hong Kong
[2] Center for E-Commerce Infrastructure Development, University of Hong Kong
{dcheung,ytlee,kcyee}@cs.hku.hk

Abstract. Web forms are commonly used to capture data on the web. With Asynchronous Javascript and XML (Ajax) programming, interactive web forms can be created. However, Ajax programming is complex in a way that the model-view-controller (MVC) code is not clearly separated. This paper discusses about a MVC-oriented web form development called "Webformer" that we develop to simplify and streamline web form development with Ajax. We introduce a scripting language called Web Form Application Language (WebFAL) for modeling web forms. Webformer hides the programming complexity by generating Ajax code directly from the web form models.

1 Introduction

Web forms are one of the mostly used applications for data capture on the web. In early days, a web form was presented as a static HTML page, which requires page reload for the web server to perform data validation. The long page reload time made web applications not as interactive as stand-alone applications. The recent use of the Asynchronous JavaScript and XML (Ajax) [12] programming pattern supports the browser to contact the server to validate the data on a web page dynamically without reloading the whole page. With Ajax, software developers write JavaScript to handle the user interface (UI) events of an HTML component (e.g. when a character is typed in a text box). The JavaScript handler can perform dynamic data validation by sending an `XMLHttpRequest` to the server. A JavaScript callback function is assigned to process the server response and render the validation results on the web page asynchronously. This approach reduces frequent page reloads in web applications and largely improves their interactivity. However, there is a price to pay in using Ajax. The browser incompatibility and the programming complexity are two pitfalls of Ajax commonly recognized [12,14,15,16].

Model-View-Controller (MVC) [13] is a design pattern commonly used for UI application development. MVC factors a UI application into three aspects: (1) the data model that represents the application context, (2) the view of the application that is presented to the user, and (3) the application controller that controls the user interactions. Although an Ajax application has the code for handling MVC interactions, the code is not modularly separated for the three aspects, i.e. the MVC code in HTML and JavaScript is mixed together in a web document. Also, Ajax applications are programmed in different languages, e.g. HTML, JavaScript, Java, etc. This makes developing and maintaining Ajax applications very difficult [12].

We have designed a rapid application development (RAD) toolkit called "Webformer", which provides a MVC framework for web forms to simplify the Ajax programming and streamline the development process.

In Sect. 2, we give an overview of the architecture of Webformer. Section 3 elaborates a scripting language called "Web Form Application Language " (WebFAL) that provides MVC specification for web forms. We introduce a tool called "Webformer Compiler" (wfc) in Sect. 4 and outline how a WebFAL script is complied into a web form runtime. The related work is discussed in Sect. 5. Section 6 concludes the paper.

2 Webformer Architecture

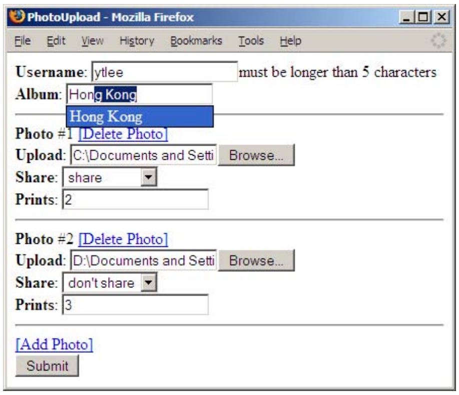
We have designed an XML-based scripting language called "Web Form Application Language" (WebFAL) for modeling web forms. A web form model written in WebFAL is complied by our "Webformer Complier" (wfc) to generate the JavaScript/HTML source to run on the browser. Webformer also provides a JavaScript engine called "webformer.js" that provides the common JavaScript routines to handle the MVC interactions, e.g. data validation and auto-completion, XML document object model (DOM) management, and Ajax messaging.

There are two types of message exchange between the server and the browser: Ajax messaging and web form submission. The Ajax messages will be exchanged when a server validation event is triggered. When a web form is submitted, the form data will be packaged in an XML document conforming to the XML Schema Definition (XSD) file generated from the WebFAL script at the design time. While the message formats are pre-defined, a Java library called "webformer.jar" is provided for parsing and composing the messages in these formats to ease the programming the Java Servlet handlers on the server-side.

The life-cycle of developing a web form using Webformer consists of two phases. The software developer first specifies the model and controller of a web form in WebFAL. The model is then compiled into an HTML template for the web designer to enhance the UI view. This two-phase life-cycle facilitates the segregation of duties between software developers and web designers to streamline the web development process.

Sample Web Form Application. We use a simple web form application as an example to illustrate the use of WebFAL and other Webformer components. This application provides a web form for the user to upload photo files to an online album. A use case of the application is described in Fig. 1.

1. The user enters his username. The entered username is validated on-the-fly against the server database.
2. The user enters the album name while the server suggests the possible names that match what the user types.
3. The user can upload multiple photos. He can click on the [**Add Photo**] or the [**Delete Photo**] link to add or delete a upload entry.
4. In each upload entry, the user specifies the photo file name, whether he wants to share the photo, and the number of prints of the photo he wants to order.
5. The user clicks on the **Submit** button to send the form data to the server.


PhotoUpload - Mozilla Firefox
File Edit View History Bookmarks Tools Help

Username: ytlee must be longer than 5 characters
Album: Hong Kong
Hong Kong

Photo #1 [Delete Photo]
Upload: C:\Documents and Setti Browse...
Share: share
Prints: 2

Photo #2 [Delete Photo]
Upload: D:\Documents and Setti Browse...
Share: don't share
Prints: 3

[Add Photo]
Submit

Fig. 1. Use case of the sample web form application

3 Web Form Application Language (WebFAL)

WebFAL is an XML-based scripting language to express the MVC specifications for web forms. A WebFAL document has a root element `<WebForm/>` and consists of five kinds of child elements: `<Model/>`, `<Validation/>`, `<Selection/>`, `<Event/>`, and `<SubmitUrl/>`.

`<Model/>` Element. The `<Model/>` specifies the data model of the web form, which is a hierarchical structure of data groups (`<Group/>`) and data fields (`<Field/>`). Each `<Field/>` is associated with a name and a data type. Possible data types include: `String`, `Number`, `Date`, `DateTime`, `Time`, `File` and `Boolean`. A `<Group/>` contains a list of `<Field/>`s or other `<Group/>`s. The minimum and maximum occurrence for a `<Field/>` or `<Group/>` on the web form can be specified. See Fig. 2 for the sample code.

`<Validation/>` Element. A `<Validation/>` specifies a set of validation rules for a `<Field/>` or a `<Group/>`. There are two types of validation rules: browser validation rules and server validation rules. Browser validation rules are some static constraints for `<Field/>`s that can be checked by the browser without contacting the server. Available static validation constraints are `<RegExp/>`, `<Length/>`, `<MinLength/>`, `<MaxLength/>`, `<TotalDigits/>`, `<FractionDigits/>`, `<MinInclusive/>`, `<MaxInclusive/>`, `<MinExclusive/>`, and `<MaxExclusive/>`. An error message may be specified so that that message is reported on the HTML page when the validation fails. A server validation is dynamically performed by the server. The `<HandlerUrl/>` specifies the URL of the server handler that performs the validation. At runtime, when the server validation event is triggered, an Ajax request containing the data field content is sent to the specified handler. The server validates the content dynamically and responses to the browser with the validation result. See Fig. 2 for the sample code.

```
<Model>
    <Field name="Username">String</Field>
    <Field name="Album">String</Field>
    <Group name="Photo" minOccurs="1" maxOccurs="unbounded">
        <Field name="File">File</Field>
        <Field name="Share">String</Field>
        <Field name="Prints">Number</Field>
    </Group>
</Model>
<Validation id="ValPrints">
    <FractionDigits errorMsg="must be integer">0</FractionDigits>
    <MinInclusive errorMsg="must be greater than 0">0</MinInclusive>
    <MaxInclusive errorMsg="must be less than 10">10</MaxInclusive>
</Validation>
<Validation id="ValUser">
    <MinLength errorMsg="must be longer than 6 characters">6</MinLength>
    <MaxLength errorMsg="must be shorter than 20 characters">20</MaxLength>
    <HandlerUrl>/ajax/userval</HandlerUrl>
</Validation>
```

Fig. 2. Sample code for `<Model/>`'s and `<Validation/>`'s.

`<Selection/>` **Element.** A `<Selection/>` provides either a static set of coded values for a `<Field/>` or specifies the URL of a server handler that provides some suggested values. Coded values are a set of permissible values for a data field. Each code value must be unique within a `<Selection/>`. A `<Code/>` can be optionally given a `text` attribute, which is a description for presenting on the web page. Suggested values are dynamically generated by a server handler through an `XMLHttpRequest`. `<HandlerUrl/>` specifies the URL of the server handler. See Fig. 3 for the sample code.

`<Event/>` **Element.** An `<Event/>` specifies an event that can take place in a `<Field/>` in order to trigger some validation or selection operations. Each `<Event/>` binds a `<Field/>` to one or more `<Validation/>`s or `<Selection/>`s. The reference to a `<Field/>` is the path of `<Group/>` and `<Field/>` names,

```
<Selection id="SelShare">
  <Code text="Share">public</Code>
  <Code text="Don't share">
    private
  </Code>
</Selection>
<Selection id="SelAlbum">
  <HandlerUrl>
    /ajax/albumsuggeset
  </HandlerUrl>
</Selection>
```

```
<Event>
  <FieldRef>Username</FieldRef>
  <ValidationRef>ValUser</ValidationRef>
  <Trigger>FocusOff</Trigger>
</Event>
<Event>
  <FieldRef>Album</FieldRef>
  <ValidationRef>SelAlbum</ValidationRef>
  <Trigger>KeyUp</Trigger>
</Event>
```

Fig. 3. Sample code for `<Selection/>`s and `<Event/>`s

delimited by a dot, from the root of the `<Model/>` to that `<Field/>`, e.g. `Photo.Description`. A `<Trigger/>` specifies the event type of the data field that triggers the specified validations/selections. For example, a text field can be validated while when a pressed key is released (i.e. `KeyUp`,). See Fig. 3 for the sample code.

`<SubmitUrl/>` Element. The `<SubmitUrl/>` specifies the URL of the server handler that processes the data submission. When a user clicks on the submit button in the web form, the form data will be re-validated, packaged in an XML document, and submitted to that URL.

4 Compilation of WebFAL Scripts

The Webformer Compiler (wfc) compiles the WebFAL script into an JavaScript/ HTML file that provides the basic layout of a web page. The wfc picks the HTML component that best represents a `<Field/>`. For example, a `String <Field/>` is translated into a text box, or a selection box (when the `<Field/>` is bound to a code selection). JavaScript code is embedded to handle the validation and selection events for the component. The web designer may modify this HTML file to enhance the view and content of the web page. The wfc also generates an XSD file from the WebFAL script that specifies the XML format of the web form data for submission to the server. The `<Model/>` and `<Validation/>`s are converted into XSD types and elements. In the `<Model/>`, the structure of `<Group/>`s and `<Field/>`s determines the XML structure while the data type of a `<Field/>` determines its XSD data type. For example, the WebFAL `Number` type is mapped to the XSD `xs:decimal` type. Moreover, the `<Validation/>` definitions are translated into the restriction facets of the XSD type.

5 Related Work

XForms. XForms is a W3C specification [9] for representing web forms in XML. XForms provides a sophisticated processing model, a large set of data types based on W3C XML Schema [8], and a rich collection of form controls and user interface. Addressing the same problem of web form modeling, XForms takes a different approach. Instead of providing a separate language, XForms is built on top of XHTML [6] in a way that XForms mark-ups are used with the XHTML mark-ups in the web document. The advantage of this approach is that the XForms statements do not require compilation and can directly run on an XForms-enabled browser. However, XForms also has some shortcomings. Firstly, XForms statements cannot be natively interpreted by popular browsers. Installation of an XForms plug-in is required to enable the browsers to run XForms. Secondly, Instead of being a self-contained language, XForms is used with other standards, such as XHTML, XPath [11], and XML Schema, which results in mixing different language constructs within a web document. This potentially leads to more implementation complication [10] and browser incompatibility.

Ajax frameworks. A variety of frameworks have been developed to simplify Ajax programming. Google Web Toolkit [4], ASP.NET AJAX [1], Yahoo UI Library [7], Dojo [2], Prototype [5], and DWR [3] are among the popular Ajax frameworks. These frameworks typically provide the JavaScript engines and server libraries to save programming efforts in coding sophisticated web UIs and Ajax messaging. In contrast, Webformer focuses on RAD of web forms using its own WebFAL language.

6 Conclusion

The key contributions of our research are the WebFAL web form modeling language, and the wfc compiler that generates the runtimes of the web forms from their models. Moreover, Webformer provides the webformer.js and webformer.jar libraries, as well as the standard Ajax message formats, to simplify Ajax programming. We also propose a two-phase web development life-cycle to segregate the duties of software developers and web designers and streamline the development process. We anticipate that Webformer will serve the developer community as a useful RAD toolkit for interactive web form programming.

References

1. ASP.NET AJAX Website. http://ajax.asp.net
2. Dojo Website. http://dojotoolkit.org
3. DWR Website. http://getahead.org/dwr
4. Google Web Toolkit Website., http://code.google.com/webtoolkit
5. Prototype Website., http://www.prototypejs.org
6. XHTML2 Working Group Home Page., http://www.w3.org/MarkUp
7. Yahoo User Interface Library., http://developer.yahoo.com/yui
8. Biron, P.V., Malhotra, A.: XML Schema Part 2: Datatypes Second Edition. W3C Recommendation (28 October 2004) 2 (2001), http://www.w3.org/TR/xmlschema-2/
9. Boyer, J.M. XForms 1.0 (Second Edtion). W3C Recommendation 14 March 2006 14 (2003), http://www.w3.org/TR/xforms/
10. Cagle, K.: Understanding XForms: In: The Model. O'REILY, xml.com (2006)
11. Clark, J., DeRose, S.: et al. XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November 1999 16 (1999), http://www.w3.org/TR/xpath
12. Garrett, J.J.: Ajax: A New Approach to Web Applications. p. 22 (2005), http://www.adaptivepath.com/publications/essays/archives/000385.php
13. Krasner, G.E., Pope, S.T.: A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. system (1998)
14. Mesbah, A., van Deursen, A.: An Architectural Style for Ajax. Arxiv preprint cs.SE/0608111 (2006)
15. Paulson, L.D.: Building rich web applications with Ajax. Computer 38(10), 14–17 (2005)
16. Smith, K.: Simplifying Ajax-style Web development. Computer 39(5), 98–101 (2006)