JavaScript Forms Validation Processing with NET+OS's Advanced Web Server

1 Document History

Date	Version	Change Description	
02/11/08	V1.0	Initial entry/outline	
2/13/08	V1.1	Fill in sections	
2/15/08	V1.2	Add section explaining application	
2/21/08	V1.3	Add in edits	
3/24/08	V1.4	Redact to include new knowledge to UseJavaScript option	
3/25/08	V1.5	Pull in additional edits	

2 Table of Contents

3 Introduction

This document describes a method for validating forms served by a NET+OS Advanced Web Server (AWS) using JavaScript.

3.1 Problem Solved

You have created a web-based system, based on web pages and served by an AWS-based web server. This web server is running on one of Digi International's embedded devices. Before any of the forms in the application are posted to the server, you'd like to have the page validate the form's fields. Validated might include any of the following:

- Ensure that certain required fields are not blank
- Ensure that certain invalid combinations are not selected
- Ensure that certain text fields do not contain any invalid characters

Further, you'd like instant feedback sent to the web user, allowing them to correct the issues, thus allowing the form to be posted to the web server.

This paper describes a method for doing all of this along with some recommendations on debugging these types of applications.

3.2 Audience

This document is intended for users with experience developing applications using either of Digi International's NET+OS development environments. This includes both the Green Hills-based and the GNU-based environments. The user should have some experience in developing web-based NET+OS applications using AWS and its pbuilder utility. Additionally to fully utilize this document the user should have at least a basic understanding of writing web pages using HTML and writing JavaScript routines.

3.3 Assumptions

This document assumes that the user has access to a Digi International NET+OS-based development environment and embedded development board for trying out the recommendations outlined in this document. You will be able to get some knowledge from reading this document only, but actually developing a test application and deploying to an embedded device will afford the reader a deeper understanding of the material presented herein.

The validation techniques discussed in this document utilize the JavaScript programming language. The aforementioned techniques do not in any way, shape or form utilize the java programming language. The reader needs to remember to keep JavaScript and java separate and understand that this document discusses JavaScript as opposed to java.

3.4 Scope

This document presents techniques for validating forms served by a Digi International NET+OS-based AWS-containing web server. The techniques contained herein are presenting alternative ways of using existing technologies.

The following items are outside the scope of this document:

- A tutorial on JavaScript programming
- A tutorial on HTML (web) programming
- A tutorial on C programming
- A tutorial on developing applications using make and cvs
- A tutorial in tcp/ip techniques
- A tutorial on web design
- Forms validation using the Basic Web Server (BWS)

3.5 Theory of Operation

Many devices and applications today, include one or more web pages as a method for collecting user input. In the case of an embedded device, this input might be used to control the operation of an electrical, mechanical or electro/mechanical device that is linked to the embedded device. Conversely, the user input might be used to set informational fields in the embedded device (ip address, serial number, serial port name…etc.). Given the criticality of the user input, the application writer might want to validate and/or edit the input of the user before submitting the information to the device. In this way, only quality data is sent on to the device.

The JavaScript functions and code described in this document are used to access web pages, forms on those web pages and elements (fields) on the forms. After accessing the elements, additional JavaScript code is used to compare the contents or settings of the fields under scrutiny against known valid contents and settings. Additionally the JavaScript code can compare the contents or setting of one field against that of another, thus ensuring that no logical incompatibilities exist between fields. After all fields, that need validation are validated, the JavaScript function calls the web page's submit method thus causing the page to be posted to the web server. If the validation fails, the web page is not posted. Additionally, if fields were set in an invalid way, the JavaScript functions can set the fields back to some known and valid state, allowing the user to rethink their submissions.

There are two levels of validation (my terminology) involved in web-based applications. One is at the field level. In this case, your application might want to put limits on the contents of fields, on their own. That is without comparing relationships with other fields. The second might be a forms-level validation, where relationships between various fields are compared and contrasted. The end result is that you might not want to allow a user to submit the form, to the web server, until and unless both levels of forms validation have passed successfully.

4 Basics

Given the fact that some of the techniques need to be performed differently under AWS vs native HTML, this document spends a little time and space first discussing how these techniques might be done using native HTML. Then this document discusses how these techniques are modified to work under AWS. Last this document points out the distinction between AWS stub functions and JavaScript functions.

4.1 How might you do this in native HTML

For this description, I am using a simple HTML form to show the basics for HTML forms validation. To see more extensive use of JavaScript, please take a look at the application that accompanies this document.

The web application that accompanies this paper, implements a simple grocery store purchasing application. It allows you to select from a small number of items and select a delivery mode. The number of each item is validated (field level validation). In addition, the type of transportation selected relative to the items selected is also validated (forms-based validation. The form is not submitted until both types of validations have been run and have passed successfully.

Please review the sample application that accompanies this document. We have documented all code in hopes of making it fairly easy to understand. On the other hand, without some understanding of HTML and JavaScript, I suspect it might be difficult to understand.

4.2 AWS ways of doing things

4.2.1 PbSetup.txt file and the Pbuilder utility

Successfully including JavaScript into an NET+OS-based aws application, may be more dependant on getting this file right then on anything else you do. The Pbuilder utility is the mechanism in AWS for converting your HTML into C code. The problem is that by default, the Pbuilder utility does not support or understand JavaScript. Thus, any JavaScript you add to your web pages, will, in some cases, be excluded from the output file(s). Additionally trying to debug this, without a good understanding of AWS internals is quite difficult. So how do you make this easier? The file PbSetup.txt is a text file that controls the actions of the Pbuilder utility. One of the commands within the PbSetup.txt file is UseJavaScript. By default this line is commented out. If you edit this file, it will look like the following:
//UseJavaScript.
To enable this command, you need to Uncomment this command. To do this, remove the two "/" from in front of the UseJavaScript command. When done it should look like this:
UseJavaScript

The UseJavaScript command, tells the Pbuilder utility to not process any code in the HTML file that it does not understand. Instead, UseJavaScript tells the Pbuilder utility to blindly copy anything you do not understand into the resultant HTML code that will be served by the web server application to any browsers surfing to your device.

4.3 AWS stub functions vs. JavaScript

In AWS, if we are doing forms validation, we are now talking about two sets of functions. It is quite important that we keep them separate. The JavaScript functions run only within the context of the browser. They do not run in AWS and they are not capable of actually updating device data.

AWS stub functions run only on the device. The "get" functions get device data and returns it, through the AWS engine to the browser. The "put" functions, take data from the browser (validated by the JavaScript function(s)) and update the device data variables.

Please notice that there is *NO* direct interaction between the AWS stub functions and the JavaScript functions. They serve two different purposes, in two different places. Further if you want to use AWS and you have device data, your application MUST employ stub functions. JavaScript can NOT be used to update device data. Additionally, if you want to validate your forms and you want to have device data, you MUST employ both AWS stub functions and JavaScript functions.

5 JavaScript functions

5.1 Where to place them

The JavaScript functions, to be accessed by the HTML code must be defined prior to their use. Most of the books and web pages that I have scrutinized, have recommended defining the functions within the head section of the HTML page. You'll notice that in this example, I have followed that method. We recommend that you define your JavaScript functions within the head section of your HTML page.

5.2 Accessing forms

HTML allows you to give all objects in a form a name. This includes the form itself. In addition, forms are accessible (through JavaScript, as an element of the document. They (the form(s) are contained within an array that is part of the document (page). Thus document.forms[0] is one way to access a form. getElementsByName() is another. Please consult either web pages on JavaScript programming or a good JavaScript programming book for more on this topic.

5.3 Accessing forms' elements

Forms elements can have both a name and an id. Thus elements can be access via the getElementById() function call. You will need to access forms elements in order to access their value in order to be able to validate the element.

6 Example Application Explanation

As was stated above, the example application is a simple grocery store web-based order form. The application allows you to select from chickens, turkeys and roast beast. In addition, you can have your order sent via local ground, U.S. Postal Service (USPS), United Parcel Service (UPS) or Federal Express (FEDEX). The application validates the elements by ensuring that you do not order more of an item then is allowed. Additionally the forms validation ensures that you do not select a delivery mode that will not ship an item that you selected. You will notice that the form will not be submitted until all validations have successfully passed.

Though the application is simple, I believe it demonstrates some powerful capabilities that are available to AWS-based applications by adding forms validation via JavaScript code.

7 Conclusion

Through this paper and its associated application, we have described to you a method for performing forms validation using JavaScript functions from within AWS applications. We believe this to be a useful tool in making your web pages more user friendly to your customers.

8 Appendix

8.1 Glossary of terms

AWS – advanced web server. An embedded web server included with Digi International's NET+OS product

BWS - basic web server. A simpler (than the AWS) embedded web server also included with Digi International's NET+OS product

HTML – hypertext markup language – a language using tags and used to create web pages

java – an object oriented programming language

JavaScript – a programming language sometimes used within HTML-based web pages for validating user input

NET+OS – an embedded operating system and development environment, developed and sold by Digi International

Stub Functions – C callback functions that are called by the AWS either when data is updated on a web page and needs to be updated in a device or when a browser needs to get updated information from the device.

Sample Code