Client Side and Server Side Validation - a case for both

Posted At : July 6, 2005 1:14 PM | Posted By : Mark Kruger
Related Categories: CFMX 7 Flash Forms, Coldfusion Tips and Techniques

When it comes to form elements, which kind of validation is appropriate? The new CFFORM with the flash format comes with very nice client side validation with highlights and feedback. Is that enough? Should you validate on the client using JavaScript or should you just stick with server side validation? In my opinion you should do both - but if you have to cut corners, make sure and validate *on the server*. Note, by *validation* I'm referring to checking values of a form for the correct type or requirements. For example, you might want to make sure an Social Security Number is 9 digits, or a phone number is 10 digits, or that an email address has been filled in and is the correct format. You get the idea. Here are the pros and cons of both approaches.

lient Side Validation

Validating on the client using JavaScript has the following items in its favor.

- *Immediate Feedback* - The user sees the error right away and can correct it before he or she submits the request.
- *Behavioral Interactivity* - You can create a "responsiveness" to your form that mimics a wizard or "steps" and gives the user the sense of progression. This is especially true of Flash Forms where the Tabbed or paned interface gives a nice "step through" type approach.
- *Maximize Performance* - This is a **great** reason to use flash forms. Since most requests from a form hit the database in some fashion, validating on the client reduces database and web server traffic by intercepting requests before they reach the server. On a busy server this can be very helpful. In addition, the user doesn't have to wait through the page refresh to see the error and correct it.

It's not all a bed a of roses however. Client-side validation has some minuses too.

- *Browser Compatibility Issues* - On a very busy public site you may have to "work around" browsers you plan to support. Not all of them will handle your nifty validation routine correctly. While it's nice to say "most people are on browser *nnn*, you may not be in a position to thumb your nose at the minority still using browser *xxx*. You'll note I'm carefully not taking any sides here (ha). For example, if you are doing a craft site your target audience may be little blue haired grandmothers. If your code generates JavaScript errors on Netscape 3.01 (just an example!) you might expect someone (probably a grandson or granddaughter) is going to be cursing you and ruing the day your were born because he or she is going to spend 2 hours that day on the phone with Grandma. Sometimes it *is safe* to ignore older browsers.
- *Security Settings* - Some folks turn off scripting because they see it as a security risk. For these folks client side validation will simply not work.
- *Easily Thwarted* - This is not a reason to *not use* Javascript validation. It's just something to keep in mind. In most cases getting around validation routines is pretty simple. If someone has malicious intent your client-side validation will not keep them from submitting false data to the server. JavaScript validation should not be the *sole validation* you use for this reason. It's great for enhancing the user

experience and lousy for ensuring that no bad data is ever submitted.

Server Side Validation

You should *always* validate any fields with requirements on the server. If it's supposed to be a date - make sure it is a date - even if it's passed from your cool little widget as a hidden field. If it is supposed to be a number, make sure it is a number. If it is supposed to be *n* characters long, make sure it's *n* characters long. Here are some of the ***bad validation*** routines I've seen on the server.

Checking Required Without Trimming - this is an egregious error. Let's say you have a field that is required. Your validation routine makes sure that the user has filled something in before proceeding. Here is what I often see in this case:

```
<cfif form.fieldname IS "">
   <cfset err = "the field ""fieldname"" is required">
</cfif>
```

What's wrong with that? It can be easily thwarted - even accidentally thwarted - by a space. If the user submits a space then "form.fieldname" Is *NOT* equal to "", even though in your mind it should be, right? Make sure that you checked the "trimmed" value of any required form element.

```
<cfif Trim(form.fieldname) IS "">
   <cfset err = "the field ""fieldname"" is required">
</cfif>
```

Or better yet write a function with a descriptive name to call - an "is" function. I like "isEmpty()"

```
<cfscript>
   function isEmpty(str) {
      if(NOT len(trim(str)))
         return true;
      else
         return false;
      }
</cfscript>
<!--- To use it --->
<cfif isEmpty(form.fieldname)>
   <cfset err = "the field ""fieldname"" is required">
</cfif>
```

I like how readable the "is" syntax is - and how familiar (isNumeric(), isDefined(), *isEmpty()*).

Bad Dates - no I'm not talking about last evening. I'm talking about not being careful with the date values that are submitted. One thing that's often forgotten is validating for range. Can someone submit a ridiculous date to your form? Chances are that CF will say "yep, it's date all right", but it will be badly out of range for what is required. For example, if you are collecting date of birth, can someone submit a date of birth that is in future? Also, since most form values are sent to the database, it is always wise to study up on the date "types" that exist on your RDBMS. For example, if you are using MS SQL's *smallDateTime* data type, you should be aware that the "minimum" date it can hadle is Jan 1, 1753. If need dates before that - you should choose a different data type.

Email Address Validation - it's not "trivial" to check and see if an email exists by querying an exchanger. If your requirements call for that, then it *can* be done (to a point), but it is not trivial, nor is it 100% reliable (as email "standards" are something of an oxymoron). However, there is NO reason not to validate an email address for format. There are numerous functions that do exactly that. They make sure there is a string followed by an "at" sign (@) followed by 2 strings separated by a dot with the last string being one of the root domains (.com, .net, .org, .gov, .biz etc). That's the least you can do. It's also possible to see if a domain exists - and it *is* pretty trivial to do that on a CFMX server, though it may be something of a performance drain if you are doing it constantly, because it uses Java networking classes. Cflib.org has this excellent **UDF** that will handle it for you.

Database Scrubbing - probaly 90% of what is submitted via forms goes into a database somewhere, yet it surprising how often form variables are passed "as is" into a CFQUERY tag. There are lots of obvious reasons to use **CFQUERYPARAM** tag, but preventing malicious SQL injection attacks is perhaps the most important - especially when handling form data.

Hey - if you have a tip for server side validation that you want to add to this post - please do!