Article
Vitaliy Serdtsev · Dec 1, 2017 5m read

$(REST - CSP - (-HyperEvents) + EasyUI + File Upload). Part 3

... but let ' s first look at error handling.

Let ' s create a regula**SrvError** error:

```
XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
<Routes>
  <Route Url="/error" Method="POST" Call="SrvError"/>
  <Route Url="/:hyperevent" Method="POST" Call="RestHyperEvent"/>
  <Route Url="/" Method="GET" Call="MainPage"/>
  <Route Url="/(.*)" Method="GET" Call="StaticFiles"/>
</Routes>
}

ClassMethod SrvError() As %Status
{
  q $$$ERROR($$$GeneralError,"Our error text from Caché")
}
```

And change the button ' s action:

```
<button onclick="$.ajax({url: 'error'})">Click me</button>
```

Now, if we click it, we will see the "Internal Server Error" message, which means that code 500 was returned. But that ' s not enough, since we want to see complete information about the error: code, text~~, developer ' s nam~~e

To do this, let ' s modify the corresponding handler:

```
...
  405: function() {alert('Forbidden method')},
  500: function
(jqXHR,textStatus,errorThrown) {alert($.toJSON(jqXHR.responseJSON,null,2))}
...
```

We can now see everything we need. What cannot be seen isn ' t visible at all.

By the way, the error can be generated differently, - s t=5/0 or throw. Full error information will be sent to the client in any case.

Let ' s jump to the key subject now, but first...

Getting JS code for execution on the client side from the server

Everything is really simple here.

Let ' s add a new method and update the button:

```
<Route Url="/js" Method="POST" Call="SrvGetJS"/>

ClassMethod SrvGetJS() As %Status
{
  s %response.ContentType="text/javascript"
  &js<alert('Hello from the server');>
  q $$$OK
}

<button onclick="$.ajax({url:'js',dataType:'script'})">Click me</button>
```

Integration with jQuery File Upload

Multiple examples on our forum mostly focused on uploading a file and refreshing the page afterwards, which is kind of... ancient these days. But it works.

The planet keeps spinning and the HTML language is also evolving, which resulted in the creation of various plugins addressing the issue above. One of such plugins based on jQuery is called jQuery File Upload.

It ' s fairly well documented, so let ' s just follow the manual:

1. the libraries are already included;
2. let ' s add a method to the URL map and the method itself

```
XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
<Routes>
  <Route Url="/error" Method="POST" Call="SrvError"/>
  <Route Url="/js" Method="POST" Call="SrvGetJS"/>
  <Route Url="/upload" Method="POST" Call="SrvUpload"/>

  <Route Url="/:hyperevent" Method="POST" Call="RestHyperEvent"/>

  <Route Url="/" Method="GET" Call="MainPage"/>
  <Route Url="/(.*)" Method="GET" Call="StaticFiles"/>
</Routes>
}

ClassMethod SrvUpload() As %Status
{
  #dim %request As %CSP.Request
  #dim stream As %CSP.BinaryStream

  if (%request.IsDefinedMimeData("files[]",1) ) {

    s stream=%request.GetMimeData("files[]")

    s file=##class(%Stream.FileBinary).%New()
    s file.Filename="c:\Temp\"_stream.FileName
    d file.CopyFromAndSave(stream)

    s tmp=##class(%ZEN.proxyObject).%New()
    s tmp.name=stream.FileName
    s tmp.size=stream.Size
    s tmp.type=stream.ContentType
```

```
        s list=##class(%ListOfDataTypes).%New()
        d list.Insert(tmp)

        s r=##class(%ZEN.proxyObject).%New()
        s r.files=list

        d r.%ToJSON("aelo")
      }
      q $$$OK
    }
```

Note:
Upon finishing the operation, the server must return a JSON response indicating the status of data saving. Some options are possible here.

3. then add a button

```
...
<button onclick="$.ajax({url:'js',dataType:'script'})">Click me</button>
<input id="fileupload" type="file" name="files[]" data-url="upload"
multiple data-sequential-uploads="false">
...
```

4. and attach a handler to it. The best way to do it is wait until the page has fully loaded, so let ' s update our **WriteCommonJS** method:

```
...
})(jQuery);

$(function(){
  $('#fileupload').fileupload();
});

</script>>
...
```

5. if we go to the page now and try selecting a file or several files, the method on the server will be instantly called, but nothing will be written to the *c:\Temp* folder.

This happens because we are trying to convert the request body from JSON into a string. We need to make an exception for the **upload** method.

Let ' s fix it:

```
...
beforeSend: function(jqXHR,settings) {
  if (settings.url!='upload') {settings.data=
$.toJSON(settings.data);}
// prior to sending, we convert the JSON object into a JSON string, except for upload

},
...
```

Everything should be working fine now.

The plugin offers a lot more, for example: *drag&drop*, upload progress and such. We are using its basic

functionality only.

In the next (final) part of the article, we will take a look at integrating one of the popular UI frameworks into our REST application, but first...

#REST API #UI Development #Frontend #Caché