Article
Vitaliy Serdtsev · Nov 17, 2017 6m read

$(REST - CSP - (-HyperEvents) + EasyUI + File Upload). Part 1

This series of articles aims to address the following topics:

- Creation of a web application based on REST pages;
- Overview of some tools for tracing (debugging) HTTP requests;
- Switching from hyperevents to... hyperevents;
- Integration with jQuery File Upload;
- Conversion of JSON from the {id:1,parentId:1} format to the {id:1,children:[{}]} format for tree visualization;
- Integration with jQuery EasyUI (using datagrid and tree as examples);
- Other topics.

Introduction

As is well known, the Caché DBMS is a set of technologies for creating web applications (both server- and client-side):

- CSP/CLS
- Zen
- Zen Mojo

Starting from version 2014.1, it supports the REST concept, and this is the concept we are going to view as the basis for building our REST application.

But how are we going to benefit from it, compared with other methods?

1. a laconic and more secure address: no CPS page or CLS class names will be revealed in the address line. That is, instead of http://server[:port]/xxx/yyy.(csp/cls/zen) it will looks http://server[:port]/xxx/
2. development comfort, separation of the client side from the server side, breaking down the business logic into multiple classes, all thanks to the new Map element in the paths map;
3. full control over the request and response format: headers, body, message format, “ tidiness ” of the generated HTML code, error handling and such;
4. as a result – traffic minimization, improved performance and scalability;
5. low entrance threshold for web developers unfamiliar with Caché, since it uses a familiar stack of technologies (html/css/javascript) and frameworks, along with advanced coding and debugging tools.

Let ’ s create our first REST page, but let ’ s prep for it first:

Note:
All our examples use Caché 2015.2 and above.

2. Download the latest versions of the following components
    - jQuery EasyUI
    - jQuery (I will be using version 2.1.4)
    - jQuery JSON
    - jQuery File Upload Plugin

Note:◦ cURL
Everything you need for the examples is will included in the project file at the end of the article (the very last part).

3. install the downloaded files into the static content folder for our **/rest** web application or, even better, into the **broker** folder to allow all application to use them;

Note:
To change the folder in the Portal, temporarily clear the dispatcher class field.

4. fix typos in the *easyui-lang-ru.js* resource file

Now, create the following class:

```
Class my.rest Extends %CSP.REST
{

XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
<Routes>
</Routes>
}

}
```

Note:
For code completion to work, make sure to specify XMLNamespace for XData blocks.

You will probably say "This is not yet a web application at all" and will be almost right.

Let ' s add some~~magic~~ code:

```
Class my.rest Extends %CSP.REST
{

Parameter CHARSET = "UTF-8";

XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{
<Routes>
  <Route Url="/" Method="GET" Call="MainPage"/>
</Routes>
}

ClassMethod MainPage() As %Status
{
  s %response.ContentType="text/html"

  &html<
  <!DOCTYPE html>
  <html>
    <head>
      <meta charset="#(..#CHARSET)#">
      <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
      <meta name="viewport" content="width=device-width, initial-scale=1.0,maximum-
```

```
scale=1, user-scalable=no"/>
      <style type="text/css">
      *{
        font-size:8pt;
        font-family:'MS Sans Serif','Arial',sans-serif,arial;
        /*padding:0;
        margin:0;*/
      }
      </style>
      <title>Home page</title>
    </head>
    <body>
      <button onclick="alert('Hello');">Click on me</button>
    </body>
  </html>
>
  q $$$OK
}
}
```

Our first REST page is now accessible at this address http://localhost:57772/rest/

Since it ' s now a simple application, we will need sessions, so let ' s add the following to our class

```
Parameter UseSession As Integer = 1;
```

Otherwise, **each** request will be executed in a new session. We will also need to redefine the CONTENTTYPE parameter, since we will mostly be working with the JSON format, rather than HTML or JS/XML/TEXT/...

```
Parameter CONTENTTYPE = {..#CONTENTTYPEJSON};
```

Otherwise, we will often need to specify the following:

```
s %response.ContentType=..#CONTENTTYPEJSON
```

Ok. Let ' s add complexity to our page by loading some static files:

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0,maximum-
scale=1, user-scalable=no"/>
  <!—Here's what we added -->
  <link rel="stylesheet" type="text/css" href="easyui/themes/default/easyui.css">
  <link rel="stylesheet" type="text/css" href="easyui/themes/icon.css">
  <script type="text/javascript" src="easyui/js/jquery.min.js"></script>
  <script type="text/javascript" src="easyui/js/jquery.json.min.js"></script>
  <script type="text/javascript" src="easyui/js/jquery.easyui.min.js"></script>
  <script type="text/javascript" src="easyui/locale/easyui-
lang-#(%session.Language)#.js"></script>

  <script type="text/javascript" src="easyui/js/vendor/jquery.ui.widget.js"></script>
  <script type="text/javascript" src="easyui/js/jquery.iframe-transport.js"></script>
  <script type="text/javascript" src="easyui/js/jquery.fileupload.js"></script>
  <!—end of the new code fragment -->
  <style type="text/css">
```

Let ' s compile the class and update the page to see that our stats failed to load (code error 404, which is "Not

Found")

I anticipate the phrase " Hey, I see no error and no code 404", so let ' s skip straight to HTTP.

HTTP tracing

Let ' s take a look at the following three methods:

1. cURL tool You can see the full list of parameters using this command

   ```
   curl -h
   ```

   Possible start examples:

   ```
   a) curl -v -X GET http://localhost:57772/rest
   ```

   In this case, we call our page with a GET method and show additional information as request and response headers

Note:
Everything preceded with ">" is a request header, everything with a "<" character before it is a response header.

   ```
   b) curl -v -X OPTIONS http://localhost:57772/rest/
   ```

   Here we take a look at the accessible call methods for the specified path. In this case, these are GET and OPTIONS.

   ```
   c) curl -v -X POST -H "Content-Type:application/json; charset=UTF-8" --data-binary @test.txt http://localhost:57772/rest/blablabla
   ```

   Here we send the content of the *text.txt* file by posting the necessary header to the specified address using the POST method.

Note:
More start examples can be found in the REST.DocServer class in the SAMPLES space.

2. HTTP tracing in a CSP Gateway
   Everything is simple here: a) Open the CSP gateway management page: http://localhost:57772/csp/bin/Systems/Module.cxw; b) Go to the " View HTTP Trace " ;

   c) enable tracing and start sending our requests. The request header and body are shown at the top, response parameters are displayed at the bottom. You can use corresponding links to switch between them

In the next part of this article, we will finally deal with static files and more...

This is a translation of an old article. Thanks @Evgeny Shvarov for the help in translation.

#CSP #JSON #ObjectScript #REST API #UI Development #Frontend #Caché

Source URL:https://community.intersystems.com/post/rest-csp-hyperevents-easyui-file-upload-part-1