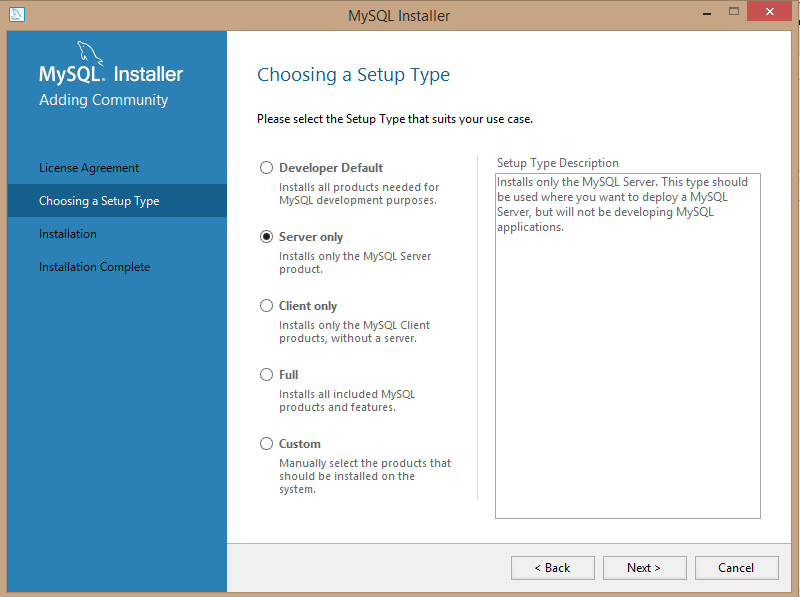
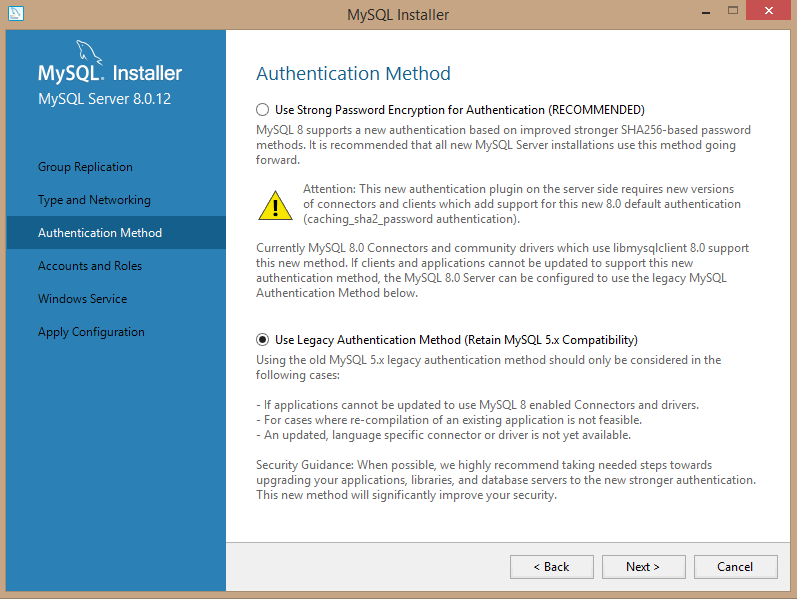
**How to run the ExtJSInAction PHP demos using a standalone PHP exe**

o cd to “CW\_Tools/tools/javascript/extjs/ExtJSInAction2Ed/ch08” and download ext-4.2.0.zip:

|  |
| --- |
| download\_extjs4 |

Extract ext-4.2.0.zip and rename the extracted directory to “ext-4.2.0”.  
  
  
o Install MySQL from https://dev.mysql.com/downloads/mysql/.  
mysql-installer-community-8.0.12.0.msi  
  
**To** keep the install as small as possible, I installed the “Server only” option:  
  
  
**Note:** The “Server only” option also includes mysql.exe (“C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe”).  
  
I selected all the default options, apart from this one, where I selected the legacy option:  
  
The defaults were used for the remaining wizard pages.  
  
  
o Create the demo MySQL database, by importing the “data.sql” file from directory “CW\_Tools/tools/javascript/extjs/ExtJSInAction2Ed/ch08/php”:

|  |
| --- |
| C:\Users\jcdc\Cygwin\home\jcdc\Github\CW\_Tools\tools\javascript\extjs\ExtJSInAction2Ed\ch08\php>"C:\Program Files\MySQL\MySQL Serv  er 8.0\bin\mysql.exe" -uroot -pteneo < data.sql  mysql: [Warning] Using a password on the command line interface can be insecure. |

**Note:** The first two lines in “data.sql” had to be edited to the following in order for the above command to work:

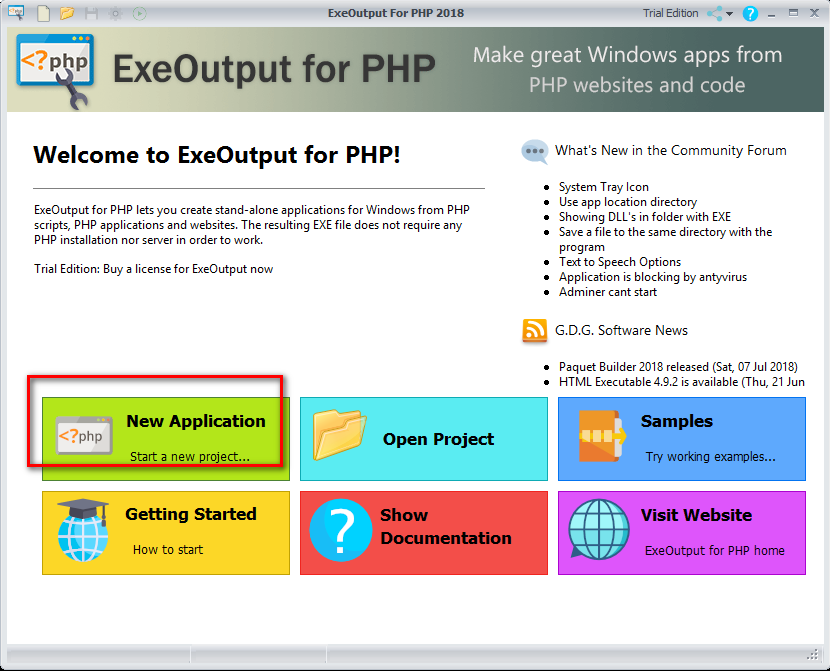
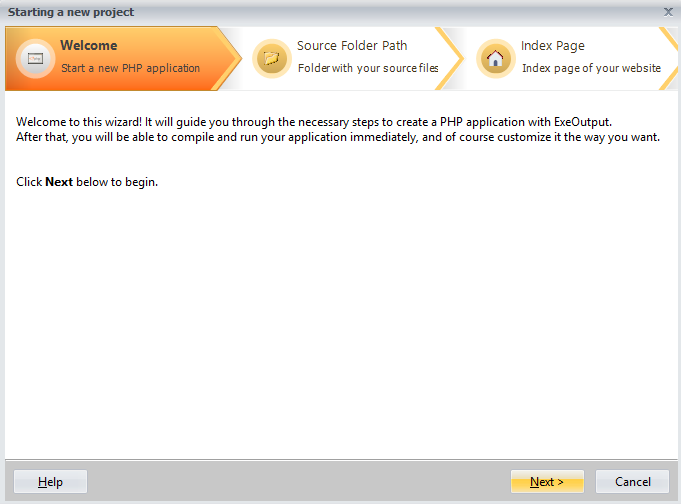
|  |
| --- |
| drop DATABASE if exists `extjsinaction`;  CREATE DATABASE `extjsinaction`; |

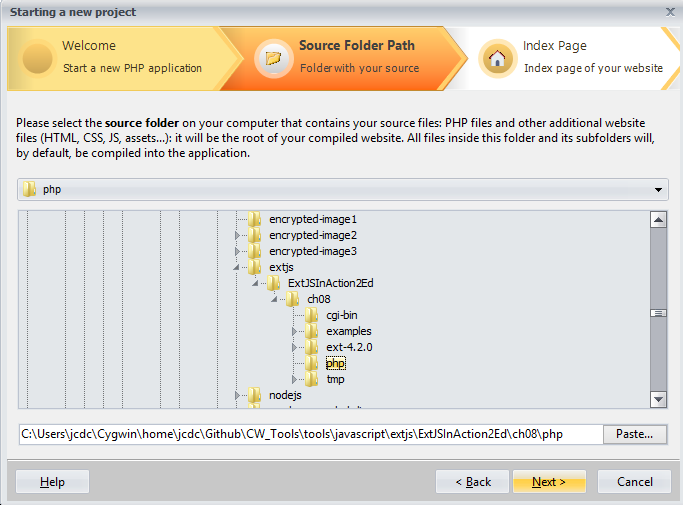
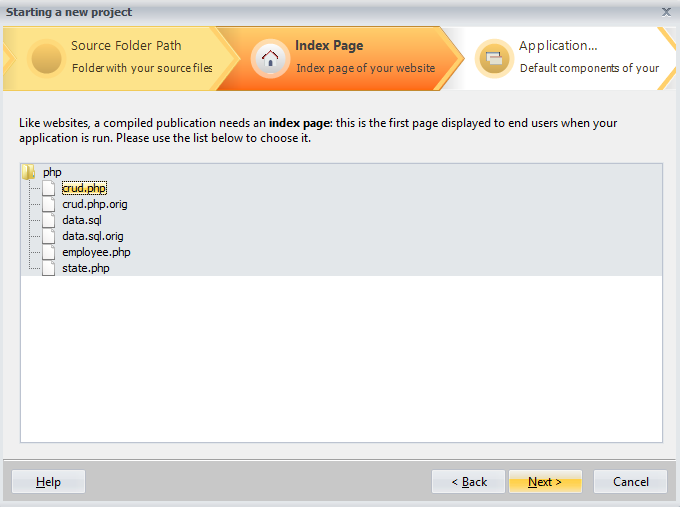
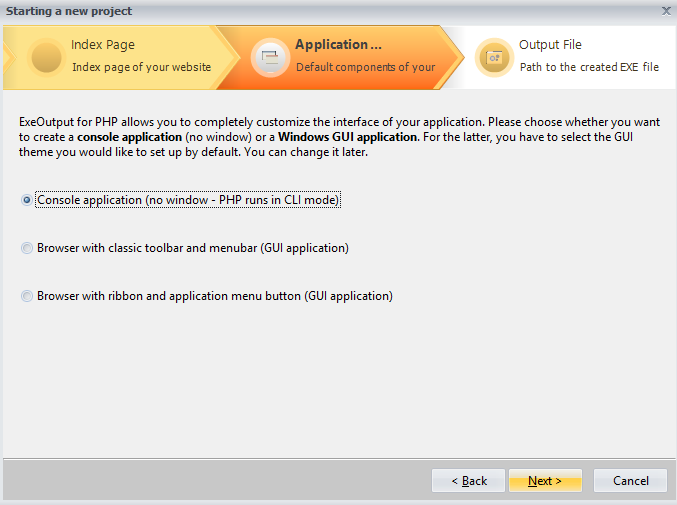
Verify that the data import was successful:

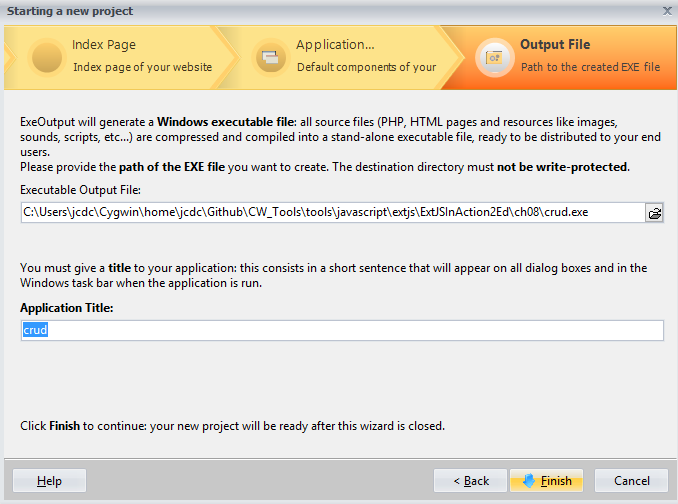
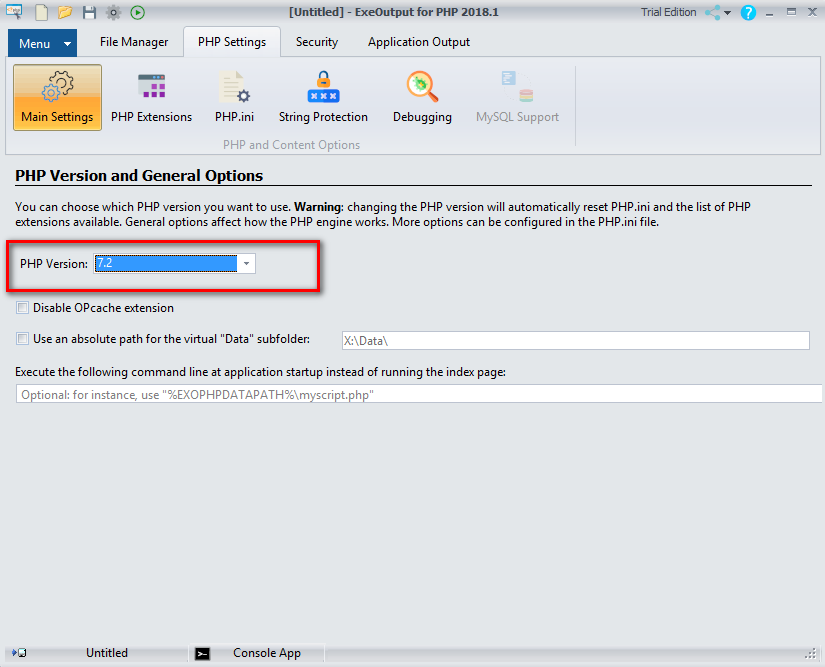
|  |
| --- |
| C:\>"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe" -uroot -pteneo extjsinaction -e "show tables"  mysql: [Warning] Using a password on the command line interface can be insecure.  +-------------------------+  | Tables\_in\_extjsinaction |  +-------------------------+  | departments |  | employees |  | states |  +-------------------------+  C:\>"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe" -uroot -pteneo extjsinaction -e "select count(\*) from employees"  mysql: [Warning] Using a password on the command line interface can be insecure.  +----------+  | count(\*) |  +----------+  | 600 |  +----------+ |

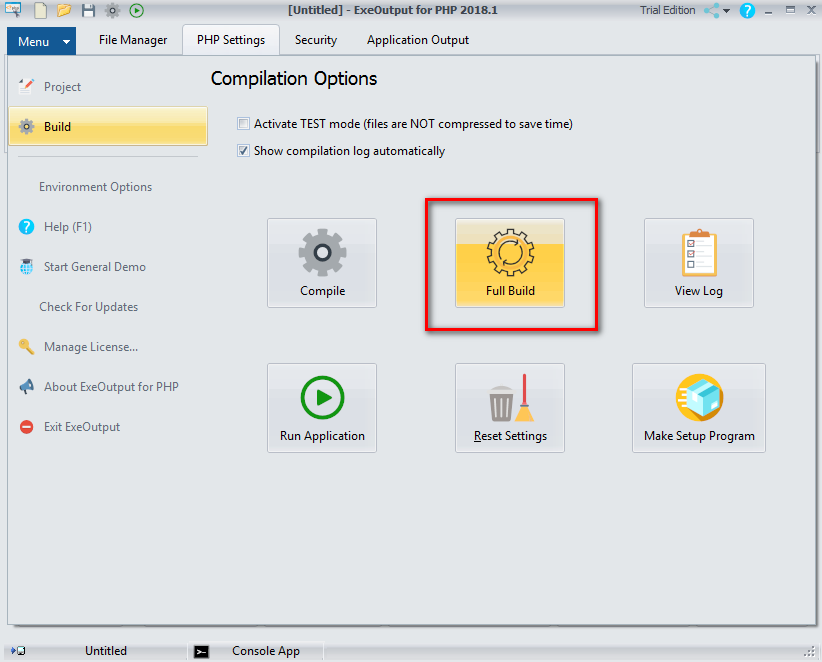
**Note:** Rather than install a PHP web server, we will use “ExeOutput” to compile “crud.php” into a standalone PHP exe called “crud.exe” …

o Install the trial version of Windows software “ExeOutput” from <https://www.exeoutput.com/download>

o Launch “ExeOutput” and select “New Application”:  
  
  
Follow the wizard:  
  


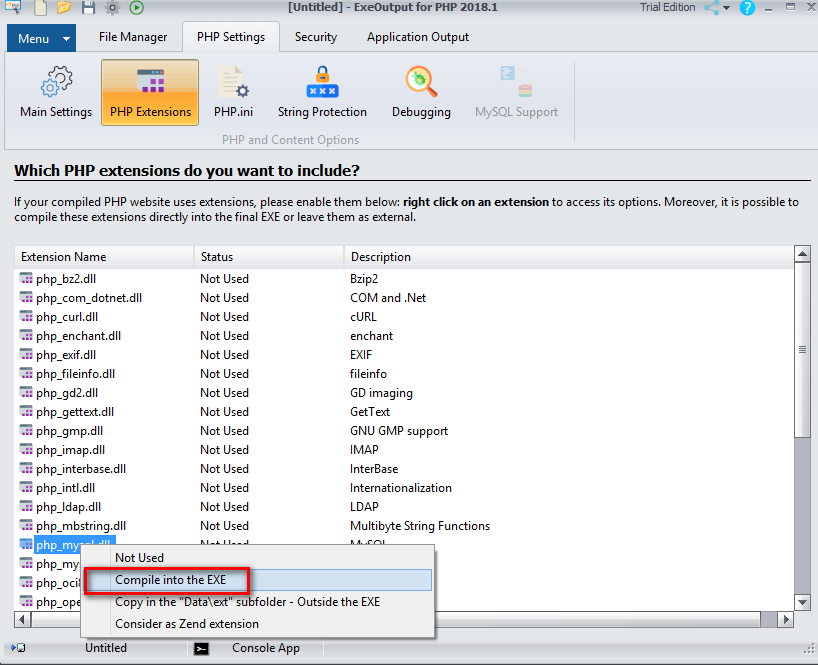
Select the directory containing “crud.php”, which is “CW\_Tools\tools\javascript\extjs\ExtJSInAction2Ed\ch08\php”:  
  
  
  
Select “crud.php” as the “index” page:  
  
  
Select the console application type:  


Keep the default location for the generated “crud.exe” file, and give it a title:  
  
  
**Note:** The default PHP version is 7.2:  


Click on the “Menu” tab and click the “Full Build” button:  


Let’s now run the built “crud.exe” directly, to see what happens:

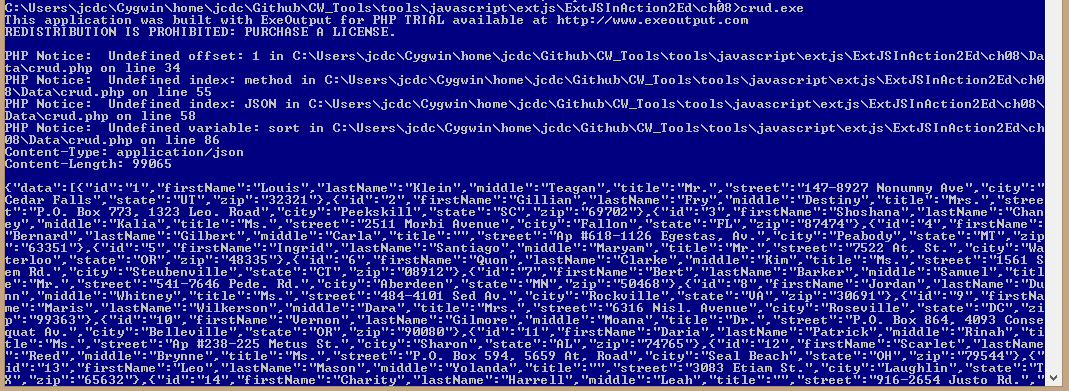
|  |
| --- |
| C:\Users\jcdc\Cygwin\home\jcdc\Github\CW\_Tools\tools\javascript\extjs\ExtJSInAction2Ed\ch08>crud.exe  This application was built with ExeOutput for PHP TRIAL available at http://www.exeoutput.com  REDISTRIBUTION IS PROHIBITED: PURCHASE A LICENSE.  PHP Fatal error: Uncaught Error: Call to undefined function mysql\_connect() in C:\Users\jcdc\Cygwin\home\jcdc\Github\CW\_Tools\tools\javascript\extjs\ExtJSInAction2Ed\ch08\Data\crud.php:3  Stack trace:  #0 {main}  thrown in C:\Users\jcdc\Cygwin\home\jcdc\Github\CW\_Tools\tools\javascript\extjs\ExtJSInAction2Ed\ch08\Data\crud.php on line 3 |

**Note:** The error about “mysql\_connect()”happens because all the “mysql\_\*” functions were removed in PHP version 7 (see <https://stackoverflow.com/questions/34579099/fatal-error-uncaught-error-call-to-undefined-function-mysql-connect>). The quickest solution is to switch to PHP version 5.6 in the ExeOutput project.   
 **Note:** But we still get the same error. This is because we need to add the MySQL extension “php\_mysql.dll” from the “PHP Extensions” page:  
  
  
Next we get the following error when we run “crud.exe”:

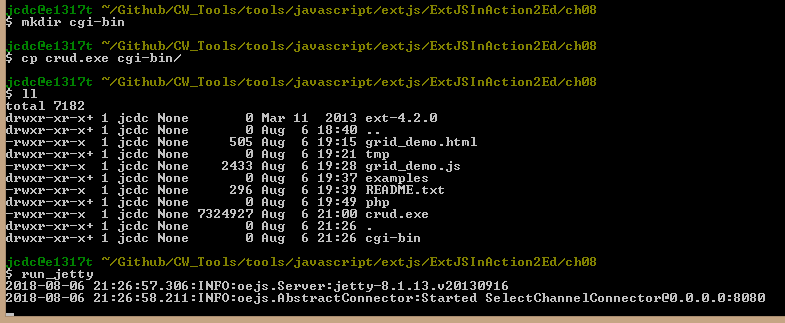
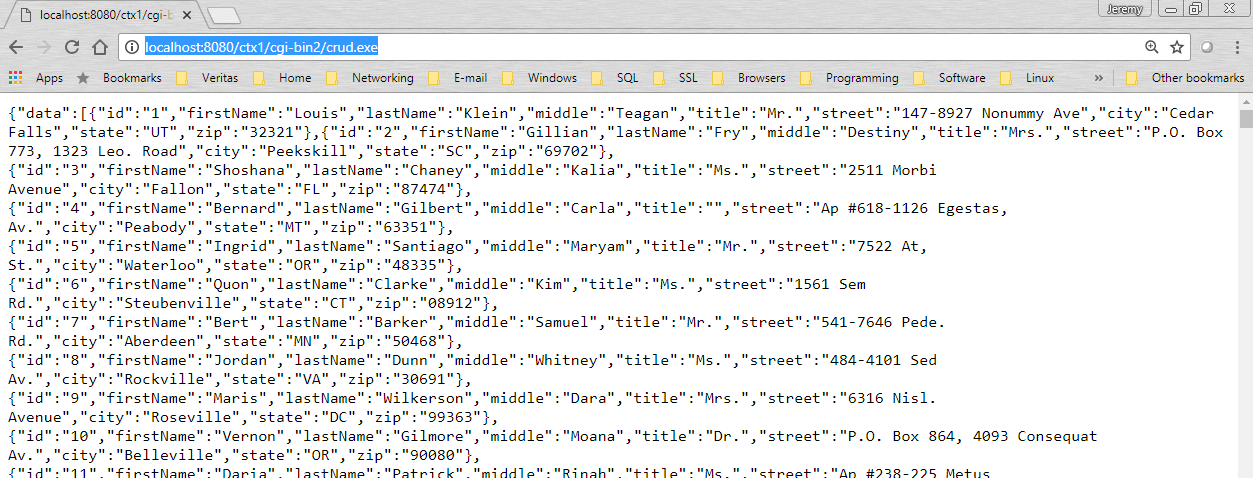
|  |
| --- |
| C:\Users\jcdc\Cygwin\home\jcdc\Github\CW\_Tools\tools\javascript\extjs\ExtJSInAction2Ed\ch08>crud.exe  This application was built with ExeOutput for PHP TRIAL available at http://www.exeoutput.com  REDISTRIBUTION IS PROHIBITED: PURCHASE A LICENSE.  PHP Warning: mysql\_connect(): Server sent charset (255) unknown to the client. Please, report to the developers in C:\Users\jcdc\  Cygwin\home\jcdc\Github\CW\_Tools\tools\javascript\extjs\ExtJSInAction2Ed\ch08\Data\crud.php on line 3  PHP Warning: mysql\_connect(): Server sent charset unknown to the client. Please, report to the developers in C:\Users\jcdc\Cygwin  \home\jcdc\Github\CW\_Tools\tools\javascript\extjs\ExtJSInAction2Ed\ch08\Data\crud.php on line 3  could not connect:Server sent charset unknown to the client. Please, report to the developers |

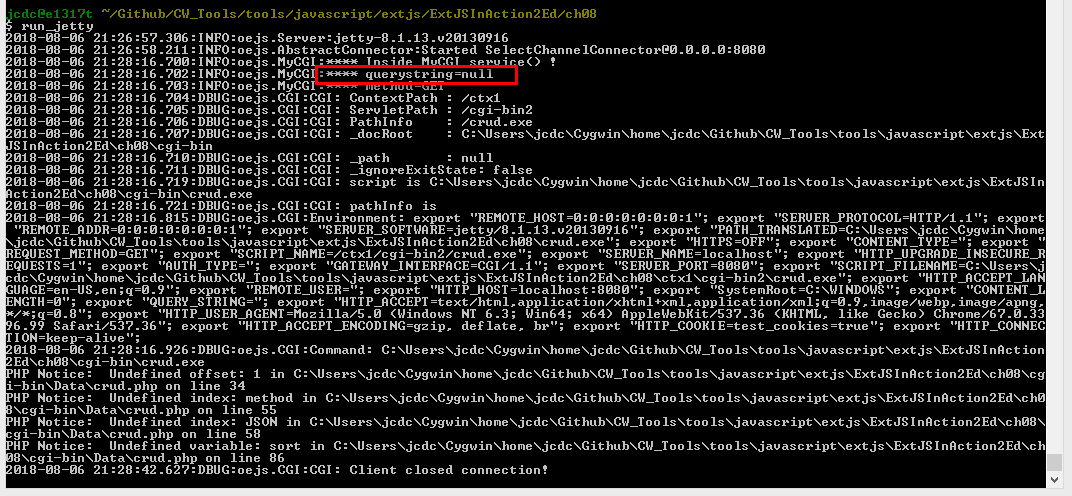
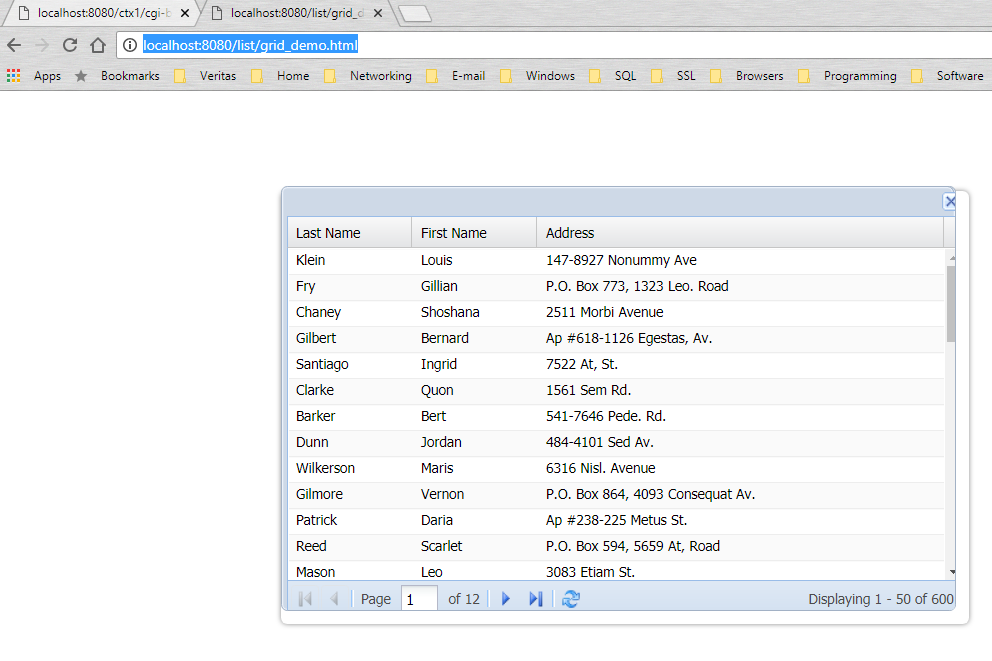
To fix this, add the following to “C:\ProgramData\MySQL\MySQL Server 8.0\my.ini”:

|  |
| --- |
| [mysqld]  # JeremyC.  character-set-server=utf8 |

(see also https://bugs.mysql.com/bug.php?id=85946)  
  
**Running “crud.exe” on the command-line is now successful 😊. There are several “PHP Notice: Undefined” warnings, which can be ignored, then the JSON output is returned:**  
  
(If you examine “crud.php”, you will see that, without any input arguments, the program defaults to the “READ” mode, which returns all the empolyee JSON output.)  
  
**Note:** I’ve made some edits to “crud.php” in order for it to be successfully called from Jetty as a command-line cgi-bin exe. Here is the full diff:

|  |
| --- |
| \*\*\* crud.php.orig 2014-01-15 13:35:28.000000000 +0000  --- crud.php 2018-08-06 19:41:29.319159600 +0100  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\* 1,6 \*\*\*\*  <?php  ! mysql\_connect("localhost", "root", "") or die ("could not connect:".mysql\_error());  ! mysql\_select\_db("extjsinaction");    require\_once("employee.php");  require\_once("state.php");  --- 1,7 ----  <?php  !  ! mysql\_connect("localhost", "root", "teneo") or die ("could not connect:".mysql\_error());  ! mysql\_select\_db("extjsinaction") or die ("extjsinaction database not found:".mysql\_error());    require\_once("employee.php");  require\_once("state.php");  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\* 16,38 \*\*\*\*  };    $param = array();  ! $param["model"] = $\_REQUEST['model'];  ! $param["id"] = $\_REQUEST['id'];  ! $param["detail"] = $\_REQUEST['detail'];  ! $param["parent"] = $\_REQUEST['parent'];  ! $param["callback"] = $\_REQUEST['callback'];  ! $param["start"] = $\_REQUEST['start'];  ! $param["limit"] = $\_REQUEST['limit'];  ! $param["sort"] = $\_REQUEST['sort'];  !    if (!isset($param["model"])) { $param["model"] = "Employee"; }  if (!isset($param["detail"])) { $param["detail"] = false; }    ! $method = $\_REQUEST['method'];  if (!isset($method)) { $method = "READ"; }    ! $datain = json\_decode($\_REQUEST["JSON"]);  $data = array();  $msg = "";  $model = $param["model"];  --- 17,65 ----  };    $param = array();  ! $query\_string = getenv('QUERY\_STRING');  ! if (isset($query\_string)) {  ! // JeremyC  ! // We are running this PHP script as a standalone exe, built using  ! // ExeOutput (https://www.exeoutput.com/download). Because this exe  ! // is being called from Jetty as a cgi-bin program, and not from a  ! // regular PHP web server, the array $\_REQUEST will not be populated  ! // for us. Instead, we need to parse the "QUERY\_STRING" environment  ! // variable that is passed from Jetty (see CGI.java).  !  ! // From http://php.net/manual/en/function.urldecode.php  ! foreach (explode('&', $query\_string) as $chunk) {  ! $param\_pair = explode("=", $chunk);  ! if ($param\_pair) {  ! $param[urldecode($param\_pair[0])] = urldecode($param\_pair[1]);  ! }  ! }  ! }  ! else {  ! $param["model"] = $\_REQUEST['model'];  ! $param["id"] = $\_REQUEST['id'];  ! $param["detail"] = $\_REQUEST['detail'];  ! $param["parent"] = $\_REQUEST['parent'];  ! $param["callback"] = $\_REQUEST['callback'];  ! $param["start"] = $\_REQUEST['start'];  ! $param["limit"] = $\_REQUEST['limit'];  ! $param["sort"] = $\_REQUEST['sort'];  !  ! $param["method"] = $\_REQUEST['method'];  ! $param["JSON"] = $\_REQUEST["JSON"];  ! }    if (!isset($param["model"])) { $param["model"] = "Employee"; }  if (!isset($param["detail"])) { $param["detail"] = false; }    ! $method = $param['method'];  if (!isset($method)) { $method = "READ"; }    ! $datain\_json = $param["JSON"];  ! if (isset($datain\_json)) {  ! $datain = json\_decode($datain\_json);  ! }  !  $data = array();  $msg = "";  $model = $param["model"];  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\* 90,97 \*\*\*\*  $dataout = json\_encode($response);    if (isset($param["callback"])) {  ! print $param["callback"].'('.$dataout.')';  } else {  print $dataout;  }  ?>  --- 117,135 ----  $dataout = json\_encode($response);    if (isset($param["callback"])) {  ! // This request has come from ExtJS, because they've given us  ! // the JSONP callback, that we need to wrap around our response.  ! $dataout\_with\_callback = $param["callback"].'('.$dataout.')';  ! print "Content-Type: application/json\n";  ! print "Content-Length: " . strlen($dataout\_with\_callback) . "\n";  ! print "\n";  ! print $dataout\_with\_callback;  } else {  + // For testing on command-line, where we won't get the JSONP  + // callback querystring as added to the request by ExtJS.  + print "Content-Type: application/json\n";  + print "Content-Length: " . strlen($dataout) . "\n";  + print "\n";  print $dataout;  }  ?> |

o Now we want to call our “crud.exe” program as a Jetty cgi-bin program.  
First, we create a local “cgi-bin” directory and copy our built “crud.exe” file into it. Then we start jetty, by running “run\_jetty” in a Cygwin prompt, from the directory above, which contains “grid\_demo.html” and “grid\_demo.js”, and the “ext-4.2.0” directory:  
  
We can now invoke our “crud.exe” program using the URL of [http://localhost:8080/ctx1/cgi-bin2/crud.exe](http://localhost:8080/ctx1/cgi-bin2/crud.exe%20)   


**Note:** Lot’s of interesting output is displayed in the jetty output, including the GET query-string, which is empty in this example:  
  
(**Note:** This “\*\*\*\*” logging was added by me – see “CW\_Tools/tools/jetty/MyCGI.java” and also the “jetty-logging.properties”file).   
  
o Finally, we want our ExtJS demo (grid\_demo.html and grid\_demo.js) to invoke our “crud.exe” program and allow us to page through the list of employee details.  
To do this, will simply run “run\_jetty” from directory “CW\_Tools/tools/javascript/extjs/ExtJSInAction2Ed/ch08” and then open <http://localhost:8080/list/grid_demo.html>:  
  
That’s it! 😊  
  
  
**Comment about the ExeOutput trial message**This doesn’t seem to cause any problems for this demo. This is because the message is mixed-in with the response header, i.e. not the body content. Here’s an example of the response seen:

|  |
| --- |
| HTTP/1.1 200 OK  **This application was built with ExeOutput for PHP TRIAL available at http: //**[**www.exeoutput.com**](http://www.exeoutput.com/)  **REDISTRIBUTION IS PROHIBITED: PUR**Content-Type: application/json  Content-Length: 8318  Server: Jetty(8.1.13.v20130916)  Ext.data.JsonP.callback1({"data":[{"id":"1","firstName":"Louis","lastName":"Klein","middle":"Teagan","title":"Mr.","street":"147-8927 Nonummy Ave","city":"Cedar Falls","state":"UT","zip":"32321"},{"id":"2","firstName":"Gillian","lastName":"Fry","middle":"Destiny","title":"Mrs.","street":"P.O. Box 773, 1323 Leo. Road","city":"Peekskill"… |

**Note:** The “content-type” is mixed with the ExeOutput trial message, but this doesn’t cause a problem. I expect this is because the receiving end (ExtJS) expects JSON, so it doesn’t matter that the header is ballsed-up.

*JeremyC 05-08-2018***END**