SEDNOVE

Sncode/Extenso

pierre.Laplante@sednove.com

2020-08-07

Cours #8

Course #8

- What we have seen in course #7
  - Create a table
  - Create a form
  - Save the form in the table
  - Diaplay the table
- What we will see in this course:
  - Save the form using AJAX
  - Using flush()
  - More on Sncode programming

Ajax

- From wikipedia:
  - Ajax (also AJAX short for "Asynchronous JavaScript and XML") is a set of web development techniques using many web technologies on the client side to create asynchronous web applications.
- jQuery:
  - jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License.
- JSON:
  - JavaScript Object Notation is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format, with a diverse range of applications, such as serving as a replacement for XML in AJAX systems

Widget : modify the widget

- We will use these 3 technologies to save data from the form and re-display the information in the table
- jQuery serialize : *Encode a set of form elements as a string for submission.*
  - https://api.jquery.com/serialize/
- jQuery ajax : *Perform an asynchronous HTTP (Ajax) request.*
  - https://api.jquery.com/jquery.ajax/

Javascript : call save_form

```
<script>
    $("form" ).on("submit", function( event ) {
        event.preventDefault();
        let data =  $(this).serialize();
        $.ajax({
              url : "/save_form.sn",
              dataType: 'json',
              data : data
          }).done(function(response) {
                  console.log(response);
                 $('#table').html(response.output);
          });
    });
</script>

// This form expect a response in JSON
```

Sncode : output

- When you print something in Sncode, everything is put in a buffer
- At the end of the process the buffer is send back to the browser
- If an error occured, the buffer is clear and an error is sent to the browser instead.
- At any moment, you can get the content of the buffer with:
  - get_output();
- You can also clear the output buffer using:
  - clear_output();

Sncode : save_form.sn

- Save data:

```
{{
    cgidata = cgidata();

    // prepare the JSON for return
    json.errcode = 0;
    json.errmesg = "";
    json.cgidata = cgidata();

}}
```

Sncode : save_form.sn

```
 // Save information
    if cgidata.email ne "" then
        res = insert(table:"my_first_form",snc:true,fields:cgidata);
        if res.sqlcode != 0 then
            json.errcode = res.sqlcode;
            json.errmesg = "Error while inserting in table : " .+ res.sqlerr;
        endif
    endif

// Get the table content
    res = sql("select * from my_first_form");
```

Sncode : save_form.sn

```
}}
<table class="table">
        <thead>
            <tr>
                <th scope="col">Email</th>
                <th scope="col">Password</th>
                <th scope="col">checkbox</th>
            </tr>
        </thead>
<tbody>
            {{ for row in res.rows do }}
            <tr>
                <td>{{ row.email }}</td>
                <td>{{ row.password }}</td>
                <td>{{ row.checkbox }}</td>
            </tr>
            {{ endfor }}
        </tbody>
    </table>
```

Sncode : save_form.sn

```
{{
    json.output = get_output();
    clear_output();

    json;
}}
```

Sncode : flush()

- Flush : send the output buffer to the browser and clear the buffer

```
{{
    for(i=1; i<= 100; ++i) do
        i; "<br>"; flush();
        sleep(1);
    endfor
}}
```

Sncode : flush 2

```
<p>Date here</p>
{{
    for(i=1; i<= 100; ++i) do
        date = datetime();
        }}
        <script>
            var para = document.querySelector("p");
            para.firstChild.nodeValue = "{{ date }}";
        </script>{{
        flush();
        sleep(1);
    endfor
}}
```

Sncode : Files operators

- In if you can use the following tests:
  - -e :Check if the file exists (boolean)
- The test is always fron the root of your site
- You can't go over your site like *../../etc/password for example*
- Example

```
if –e "/staging/newfile.txt" then
    ...
endif
```

File operators : all tests

- -z : Check if the file has a size equal to 0 (boolean)
- -r : Check if the file can be read (boolean)
- -w :Check if the file can be open for writing (boolean)
- -x : Check if the file can be executed (boolean)
- -l : Check if the path points to a link (boolean)
- -f : Check if the path points to a file (boolean)
- -d :Check if the path points to a directory (boolean)

File *operators*

- -s : Return the size of the file (in bytes)
- -m :Return the modification time of the file (in secs.) since epoch
- -a : Return the last access time of the file (in secs.) since epoch
- -c : Return the creation time of file (in secs.) since epoch
- -u : Return the owner (user) of the file
- -g : Return the owner group of the file

File operator : side effect

- All file operator astart with '-' followed by a character
- What will the compiler do with the following code

```
a   = 5-e;
```

```
Syntax error in string :In compiler, syntax error, unexpected
SN_T_FILE_EXIST, expecting ';' or SN_T_EOF or SN_T_ENDEXTENSO or ','
```

- Use instead :

```
a = 5 - e; // space is mandatory in this case
a = 5 -ertyuiop; // will cause the same problem
```

Sncode : compound operators

- Example:
- a+=5; // is equivalent to a = a + 5;
- List of compound operators:
  - += Addition assignment
  - -= Subtraction assignment
  - /= Division assignment
  - %= Modulo assignment
  - *= Multiplication assignment
  - **= Power assignment
  - .+= string concatenation a .+= "allo";

Sncode : binary operators

- Or : |

0b100 |

0b001; // return 5 or 0b101

- And : &

0b101 &

0b001; // return 1 or 0b001

Binary operators

- Not : ~

~ 0b001; // return -2 why ?

- **Two's complement** is used in computing as a method of signed number representation.

printf("%x", -2); // return 0xfffffffe

~0b001 & 0b111; return 6 or 0b110

Binary operators

- Xor : ^

0b101 ^

0b100; // return 1 or 0b001

- Shift left: <<

0b001 << 2; // return 4 or 0b100;

- Shift right: >>

0b1000 >> 2; // return 2 or 0b0010

- shift left and shift right are often used to multiply or divide by 2

Compound operators

- |= Bitwise OR assignment
- &= Bitwise AND assignment
- .+= String concatenation assignment
- >>= Left shift assignment
- <<= Right shift assignment

Sncode : Loop for (i=0;i<10;i+=2) do … endfor

- Another example:

```
for(i=0,j=10;i<10;i+=2,--j) do // a=++i; a=i++; --j j--
    i; " ";  j; " ";
     if i == 5 then "\n"; endif
endfor

return
0 10 1 9 2 8 3 7 4 6 5 5
6 4 7 3 8 2 9 1
```

Sncode : Loop for i in variable do … endfor

```
a = [1,2,3,4] ;
for i in a do
  i; " ";
endfor // return 1 2 3 4
for i in { "x" : 1, "y" : 2 } do
  i; " ";
endfor // return
{"key":"x","nbrows":2,"value":1}
{"key":"y","nbrows":2,"value":2}
```

Sncode : Loop for i function(…) do … endfor

- Some function in Sncode can be used as a "callback" function
- The loop will be perform for each row return by the callback function

```
for i sql("select username from sed_login_user") do;
    i; " ";
endfor
```

will return for each user:


```
{"sqlerr":"","error":false,"nbrows":23,
"sqlcode":0,"rows":{"username":"arnaud"},
```