SEDNOVE

Sncode/Extenso

pierre.Laplante@sednove.com

Version 1.2 : 2021-12-31

Course #1

- What you will see in this course:
  - Introduction about Extenso
  - Sednove
  - Extenso – Sncode
  - Architecture
  - Sncode programming

About this course

- Introduction to programming in Sncode
- Goal: Enable non-programmers to learn how to program, in particular with Sncode and Extenso
- What you will learn:
    - Sncode
    - Extenso
    - HTML
    - CSS
    - Javascript
    - Jquery
    - Ajax
    - Websocket
    - WebRTC

What you will need for this course

- HTML
  - https://www.youtube.com/watch?v=BvJYXl2ywUE
  - https://www.youtube.com/watch?v=PypMN-yui4Y
  - https://www.youtube.com/watch?v=1rbo_HHt5nw
  - https://www.youtube.com/watch?v=bFvjE4ZRtSE
- CSS / Bootstrap
  - https://www.youtube.com/watch?v=-qfEOE4vtxE
  - https://www.youtube.com/watch?v=1PnVor36_40
- Javascript
  - https://www.youtube.com/watch?v=cmIkfezTnBE&list=PL9dbBb7Ml9bXwgPTH5STNGEQNQNeCDXdu

What you will need for this course

- jQuery
  - https://www.youtube.com/watch?v=hMxGhHNOkCU
- JSON
  - https://www.youtube.com/watch?v=iiADhChRriM)
- SQL MariaDB/Mysql
  - https://www.youtube.com/watch?v=p3qvj9hO_Bo
- REGEX : Regular Expression
  - https://www.youtube.com/watch?v=rhzKDrUiJVk
- Git
  - https://www.youtube.com/watch?v=IHaTbJPdB-s
- And programming experience in a language...

What you **may** need for this course

- Linux (CentOS). Basic Command line interface (CLI)
  - https://www.youtube.com/watch?v=5jIIOkA0NpI
- Apache configuration
  - https://www.youtube.com/watch?v=rCr3-YlL5S8
- C programming
  - cmake / make / gcc
- Websockets

About this course

- keep your personnal question for later with me directly
- this is an introduction course not an advanced course.
- the course is recorded
- if you have a question, please raise your hand first
- you will need a headphones to speak

Sednove

- Founded in 1997 by Chantal Bilodeau and Pierre Laplante
- Web and mobile applications development
- Technological development
- Branding
- Design

Platinum

- Founded in 1995.
- Software developped in FoxPro
- Front Desk component only until 2000 (scheudle, patient file, transactioons and reports)
- EHR module in 2000 (doctor notes, patient flow with check in and calling patients to the room)
- in USA and around the world since 2002
- Need to move to the cloud to integrate new tools

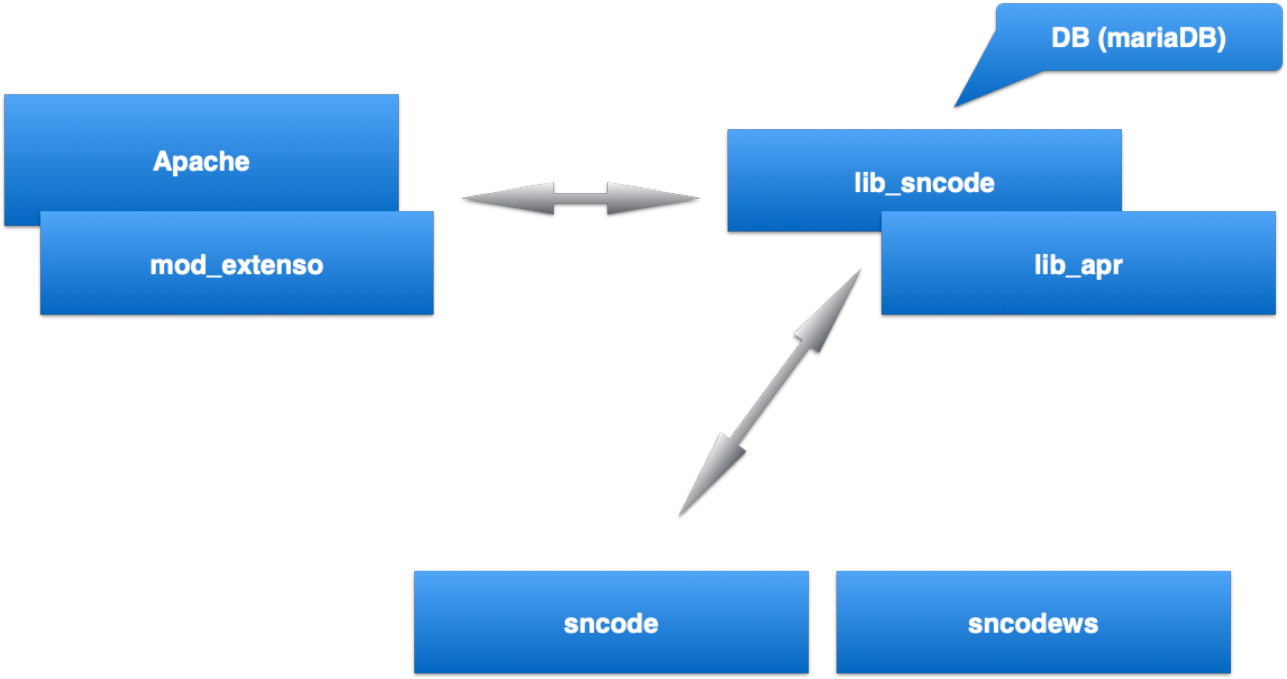
Extenso / Sncode

- Extenso : Clients and programmers interface
- Sncode : Programming language

Sednove's tools

- slack
- clickup
- uxpin
- gecko
- email
  - laplante@sednove.com
- Phone
  - 514-945-1779 (also whatsapp)

Architecture of Extenso



What is different about Extenso ?

- Dynamic / static
- Compile language
- Security with virtual machine
- Grid manager
- Style manager
- Modules manager
- Database manager
- Git/Gitlab managment of modules

Sncode

- Key concepts:
  - Sncode is a compiled language
  - Uses a virtual machine to execute code
  - Rich and extensible library
  - External modules
  - Simple syntax for non programmer
  - Power-full for professional programmer

compile, virtual machine etc.

- Interpreted language, PHP, Ruby
- Compile language : C
- Virtual machine : Java, C#, Sncode
- JIT : Just in Time
- Assembler
- Machine code

Sncode concepts

- File with extention .sn are compile and execute
- File with extension .snc are file already compiled in binary
- Convention use in the naming of site:

  https://ssnode.sednove.com for the staging version
  https://sncode.sednove.com for the production version

Sncode concepts #2

- staging require a login to modify the site
- Production deployment (or publishing) is the process of copying the file from staging to production
- Directory for staging is /staging
- Directory for html is /html

Sncode

- In a page everything that is not between {{ and }} is print as-is
- {{ Start Sncode
- }} End Sncode
- All text outside of Sncode is returned to the browser without being parsed

IDE

- Introduction to IDE in Extenso
- How to execute a program:
  - In staging : https://ssncode.sednove.com/ex.sn
  - In html : https://sncode.sednove.com/ex.sn

Simple example

```
<html>
    <body>
        <h1>{{ "Date is "; datetime(); }}</h1>
    </body>
</html>
```

Exercices

- Exercice #1 : try to reproduce the previous example.
  - Create a file in /staging/ex1.sn
  - Execute it with the URL https://ssncode.sednove.com/ex1
- Exercice #2 Only display the time not the date

  PS : https://extenso.live
- PS #2 https://getbootstrap.com/docs/4.5/getting-started/introduction/

Sncode's documentation

- All documentation under https://extenso.live
- https://module.sednove.com
- Man pages under Linux:

  man sql

Sncode's types

- Integer : `{{ a = 5 ; }}`
- Double : `{{ a = 3.1415 ; }}`
- Array : `{{ a = [ 1, 2.0, "3.1415" ]; }}`
- Boolean : `{{  a = true; }}`
- Null : `{{ a = null ; }}`
- Undefined : `{{ a = undefined ; }}`
- Associative Array / Hash array / Context :
  `{{ a = { "x" : 1.5, "y" : 2.0 } ; }}`
- `a.type()` will return the type of variable a

Integer

- Try This program:

```
{{
      a = 5;
      b = a / 2;
      "b = "; b;
      type(b);
}}
```

Operators

- By order of priority:
  - +, - ,
  - *, /,
  - ** (power), % (modulo)
- Try:

```
2 + 3 * 4 ** 2 => 50
```

- Use () to modify the order:

```
((2 + 3) * 4 ) ** 2 => 400
```

SEDNOVE

Sncode/Extenso

pierre.Laplante@sednove.com

Version 1.4 : 2020-07-03

Course #2

- What we have seen so far:
  - Extenso presentation
  - How to use IDE
  - Sncode's type
  - Structure of directory in Extenso

Integer / float

- Try this program:

```
{{
   a = 5;
   b = a / 2.0;
   "b = "; b;
   type(b);
}}
```

Function printf

- printf : print formatted
- `printf("Number = %05d", 10);`
- d : use to print integer
- Try
  - `printf("%7d", 10); a=printf("%x", 10); a;`
  - `printf("%-10s", a); printf("%10.4f", 10.2);`
  - `printf("%010.4f", 10.2); printf("%+010.4f", 10.2);`
  - `printf("%+10.4f", 10.2); printf("%10.1f", 5.17);`
- f : float, s : space, x : hexadecimal

Floating point number

- Example:

```
a = 1.123456789;
a;
```

- By default sncode use: `printf("%.8f", number)`
- According to Wikipedia:

"In computing, **floating-point arithmetic** (**FP**) is arithmetic using formulaic representation of real numbers as an approximation to support a trade-off between range and precision."

Floating point number

- Floating point number are represented as double in C
- Try:

```
{{ printf("%.20f", 0.1+0.2);}}
```

- Check:

https://docs.python.org/3/tutorial/floatingpoint.html

https://doc.lagout.org/science/0_Computer%20Science/3_Theory/Handbook%20of%20Floating%20Point%20Arithmetic.pdf

Floating point number

- Try this program:

```
a = 1/5;
b = 1/5.0;
c = 1.0/5;
"a="; a; ", b="; b; ",c ="; c;
```

Floating point number

- Try:

```
a = 48.0 * atan(1.0/49.0) +128.0 *
      atan(1.0/57.0) -
      20.0 * atan(1.0/239.0) + 48.0 *
      atan(1.0/110443.0);

  printf("%.25f", a);
```

Floting point number comparaison

- Floating point comparaison : operator ==
- try:

```
a=0.15+0.15; // 0.30000000000155
b=0.1+0.2;   // 0.30000000000144
a==b;
```

- return !

```
false
```

Floating point number

```
function compare(a,b)
    //!code Minimal function to compare FPN to 0.0001
    if a==b then
        return true;
    endif

    if abs(abs(a) - abs(b)) < 0.001 then
        return true;
    endif
    return false;
endf
a = 0.15+0.15; b = 0.1 + 0.2; a==b; compare(a,b);
```

Comparaison operators

- < : less than
- > : greater than
- <= : less or equal
- >= : greater or uqual
- <=> : compare 1⇔2; 2⇔1;
- != : not equal

FPN example

```
if 1 == 2 then
     "Oh la la something is wrong here";
else
     "Ok 1 is not equal to 2";
endif
```

FPN compare with string

- String are automatically convert to double

```
a = "0.0001";
b = 0.0001;
if a == b then
    "a is equal to b";
else
    "a is not equal to b";
endif
```

- Test:

```
if "0" == 0 then "true"; else "false"; endif
```

Comparaison operator

- If we do:

```
a = 5 < 6; a;
```

- a is a boolean : true or false

```
if a then "true part"; else "false part"; endif
```

- If we do:

```
a = 1;
if a then "true part"; else "false part"; endif
```

- 1 is true and 0 is false

The art of programming... part #1

- Good indentation
- Use good meaningful variable name :english, lower case, no plural _
- Use comment:
  - `/* ... */`
  - `/*`
  - `...`
  - `...`
  - `*/`
  - `// comment`
  - `# comment`

Comparaison operator : <=>

- Use to compare 2 numbers
- Return
  - -1 if left then
  - 0 if equal
  - 1 if greater than
- Example using a new statement : switch

```
a = 60;
b = 5;
switch a <=> b do
```

Comparaison operator : <=>

- Use to compare 2 numbers
- Return
  - -1 if left then
  - 0 if equal
  - 1 if greater than
- Example using a new statement : switch

```
a = 60;
b = 5;
switch a <=> b do
```

Comparaison operator : <=>

- Use to compare 2 numbers
- Return
  - -1 if left then
  - 0 if equal
  - 1 if greater than
- Example using a new statement : switch

```
a = 60;
b = 5;
switch a <=> b do
```

Comparaison operator : <=>

```
case -1:
    "a is lower than b";
    endc

case 0:
    "a is equal to b";
    endc

case 1:
    "a is greater than b";
    endc

default:
    "Ohh la system error";
    endc
ends
```

SEDNOVE

Sncode/Extenso

pierre.Laplante@sednove.com

Version 1.4 : 2020-07-03

Course #3

- What we have seen in course #2
  - Floating point number
  - Comparaison operators
  - Comments

Boolean

- 2 values: true or false
- Examples:

```
a = true;  b = false;
a != b;
a == b;
a < b;
a > b;
```

String

- A string variable is created using quote or double quote
- Example:

```
a = "Pierre";
a;
a= 'Pierre';
a;
```

- Why single quote or double quote ?

String : double vs single quote

- Double quote support escape sequence
- Ascii characters : `a = "Pier\x43\x44";` return PierCD;
  - 43 is hex value of character C
  - 44 is hex value of character D
- UTF-8 characters: `a = "\ucf80 = 3.1415"; a;`
  return π = 3.1415
- Complete list of utf-8 character:

  http://www.fileformat.info/info/charset/UTF-8/list.htm?start=1024

String : escape sequences

- `\\ : display \`
- `\n : newline`
- `\a : alert beep bell`
- `\b : backspace`
- `\t : tab`
- `\r : carriage return`
- `\v : vertical tab`
- `\o : octal number`
- `\f : Formfeed Page Break`
- `\' ou \" : Display ' or "`

String : when " and ' are not enough

- q(....) : quote is (and)
  - `q( x='"\t );`
- dq((...)) : quotes are 2 characters ((and))
  - `dq([ x='"\t ]);`
- qq(...) : double quote are (and)
  - `qq( x='"\t );`
- dqq((...)) : double quotes are 2 characters ((and))
  - dqq((x='"\t));

String : quote and double quotes

- You can also use the following characters:
- /, #, @, !, $, %, ?, &, \
  - `q/abc/;`
- (and),[and], { and }, < and >
  - `q(abc);`
- /, #, @, !, $, %, ?, &, \ follow by any characters for 2 characters (which in this case is not really clear)
  - `dq/edghe/;`

String comparaison

- `a = "001"; b = "1";`

  `a == b; // will convert a and b to double before doing the comparaison`

  or try

  `a eq b`; // string are compare case sentitive
- Sncode knows that a is a string and b is string or convert them
- `type(a); type(b);`

String comparaison operators

- eq : equal(==)
- ne: not equal (!=)
- lt : less than (<)
- le : less or equal (<=)
- gt : greater than (>)
- ge : greater or equal (ge)
- st : start
- ns : not start
- cmp : compare (⇔)

String comparaison st and ns

- Can you figure out what is the use of st and ns ?

String comparaison st and ns

- st stands for start with
- ns stands for not start with
- Example:

```
a = "/usr/local/website/plv1/staging/tmp";
a st "/usr/local"; // true
a st "usr/local"; // false
a ns "usr/local"; // true
```

String comparaison and date

- SQL usually return date in the military format:
  - YYYY-MM-DD HH:MM:SS
- String comparaison can then be used to compare date and time

```
a = "2020-12-14 18:32:33";
a lt "2020-12-15 19:19:19"; // return true
```

String and sub-string

- You can use the [] operator to get substring of a string:

```
a = "Pierre Laplante";

a[0:5];      // return Pierr
a[1:5];      // return ierr
a[:5];       // return Pierr
a[5];        // return e
a[7:];       // return Laplante
a[:-2];      // return Pierre Laplan
a[-4:-2];    // return an
```

String operator

- To concatenate 2 string use the operator .+

```
a = "pierre " .+ "laplante";
a; // return pierre laplante
b = "pierre " .+ 35;
b; // return pierre 35
```

String exercice

- Write a program to reverse the string a="pierre"; b
- `length(string)` return the length of a string

```
name = "Etienne";

b = name[6:7];
b = b .+ name[5:6]; // b .+= name[5:6] a = 7; a+= 7;
b = b .+ name[4:5];
b = b .+ name[3:4];
b = b .+ name[2:3];
b = b .+ name[1:2];
b = b .+ name[0:1];
```

String reverse

```
a = "pierre laplante";
b = "";
len = length(a);
for(i=0; i<len; ++i) do
    b = a[i:i+1] .+ b;
endfor
b;
```