Ocaml Standard Library By Examples

Module

- structure = implementation of module
- signature = interface of module

Struture : Example

```
module MyStack =
  struct
    type t = int list
    exception Empty
    let empty = []
    let push x t = x :: t
    let pop t =
      match t with
        [] -> raise Empty
       |h::t -> (h, t)
    let first t =
      match t with
        [] -> raise Empty
       |h::t -> h
  end
```

Signature : Example

```
module type MYSTACK =
  sig
    type t
    exception Empty
    val empty : t
    val push : int -> t -> t
    val pop : t -> int * t
  end

# module AbstractStack = (MyStack : MYSTACK);;
module AbstractStack : MYSTACK
# AbstractStack.first [1;2;3];;
Unbound value AbstractStack.first
```

Standard Library

- Pervasives
- List
- Set
- Map

Parvasives

- Module already opened when OCaml starts.
- Comparisons
- Boolean, Integer, Float-point, Bitwise operations
- Input / Output
- exit

Set : Example

```
module OrderedInt =
  struct
    type t = int
    let compare = compare
  end ;;

module IntSet = Set.Make(OrderedInt) ;;

let setA = IntSet.add 3 (IntSet.add 6 IntSet.empty) ;;
let setB = List.fold_left Set.add Set.empty [2;4;6;8] ;;

IntSet.elements (IntSet.union setA setB) ;;
IntSet.elements (IntSet.inter setA setB) ;;
IntSet.iter (Printf.printf “%d, “) setA ;;
IntSet.fold (+) setB 0 ;;
```

Map : Example

```
module IntMap = Map.Make(OrderedInt) ;;

let aMap =
  IntMap.add 3 “three”(
    IntMap.add 2 "two" (
      IntMap.add 1 "one" IntMap.empty)) ;;

IntMap.find 2 aMap;;
IntMap.find 2 (IntMap.map (fun x -> x ^ “!!!”) aMap);;
IntMap.find 2 (IntMap.mapi (fun k v1 -> (string_of_int k)^”-
>”^v1) aMap);;
```

More...

- Printf, Scanf
- Stack, Queue
- Array
- Hashtbl
- Random

References

- OCaml manual : http://caml.inria.fr/pub/docs/manual-ocaml/
  - The core library
  - The standard library
- Developing Applications With Objective Caml : http://caml.inria.fr/pub/docs/oreilly-book/index.html
- OCaml tutorial : http://www.ocaml-tutorial.org/