Ocsigenによる Webアプリケーション

今井 敬吾

OCaml-Nagoya

2009/8/30, OCaml Meeting Tokyo 2009
@ 東京大学本郷キャンパス山上会館

Ocsigen：OCamlの Webアプリケーションフレームワーク

- Webアプリの表現力と安全性の向上を目指す
  - できるだけ静的型付けで安全性を保証
- 論文が ICFP 2009 に出てます (*)
  - これからも注目されるでしょう
- http://ocsigen.org/ - Ocsigen - fresh air in Web Programming

(*) V. Balat, J. Vouillon and B. Yakobowski, Experience Report: Ocsigen, a Web Programming Framework, International Conference on Functional Programming, ACM, 2009.

Ocsigen と Eliom

- Ocsigen
  - OCamlで書かれたWebサーバー
- Eliom
  - Ocsigenで動くWebアプリフレームワーク
  - (Eliom = フランス語で ヘリウム (Helium))
- Eliom について話します

Eliomのポイント

1. XHTML (XML) 型付け
   - valid な XHTMLのみを生成する
2. サービスとリンク・フォームの型付け
   - リクエストパラメータの不整合がない
3. DBアクセスも型安全
   - PG'OCaml を使った型安全DBアクセス

XHTML型とXHTML.Mモジュール

- XHTMLの要素を返す関数が提供されている
- 型は XHTML の DTD を(だいたい)忠実に再現
- 構造多相がある OCaml ならでは
- 例

```
html (head (title (pcdata "Hello")) [])
     (body [h1 [pcdata "Hello,World!"]])
```

多相バリアントと XHTML型

- 例えば，**body** 関数は

```
- : [< `Address | `Blockquote | `Del
    | `Div | `Dl | `Fieldset | `Form
    | `H1 | `H2 | `H3 | `H4 | `H5 | `H6
    | `Hr | `Ins | `Noscript | `Ol
    | `P | `Pre | `Script | `Table | `Ul ]
   XHTML.M.elt ->
  [< `Body ] XHTML.M.elt
```

…という型をもつ.

- [< ...] は 高々 ... しか含まない, という型.

EliomのCamlp4拡張：
XHTML構文

- << ... >> 部分にXHTMLリテラルを配置
- XHTML 中 $ ... $ 部分に OCaml の式を配置

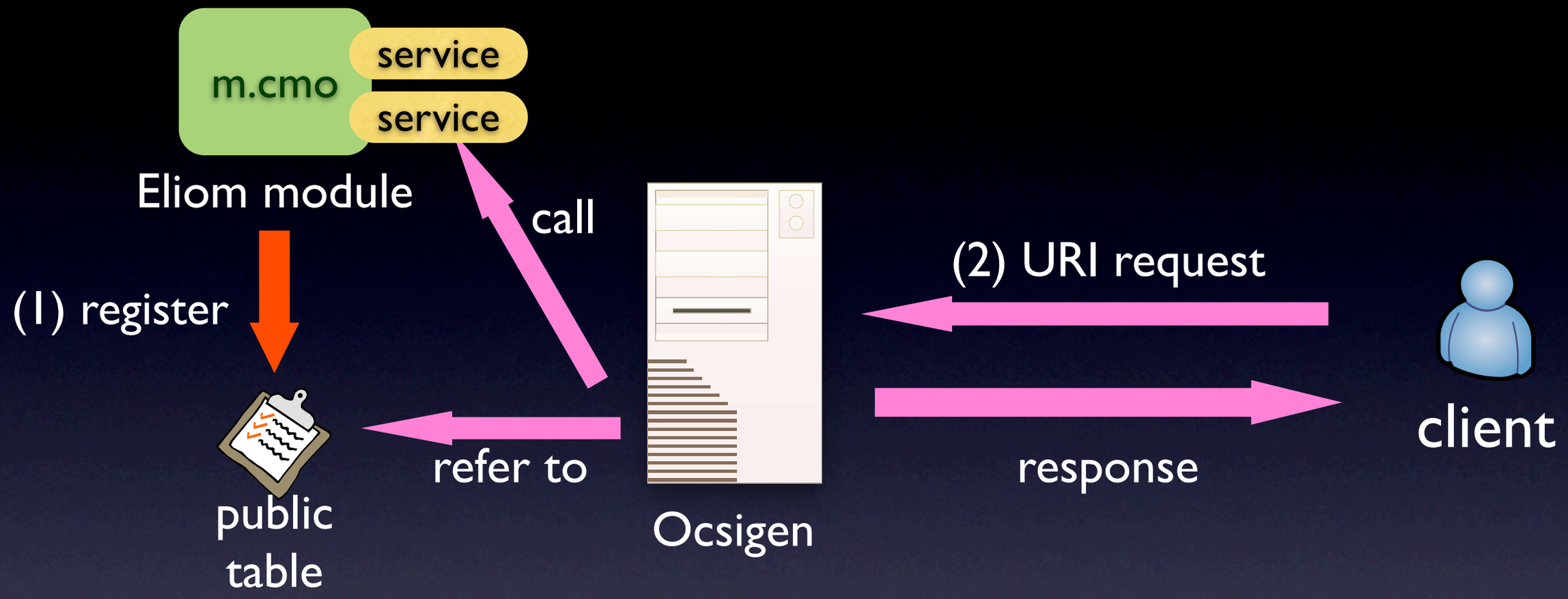
- 例

```
<<   <html><head><title>Adder</title></head>
       <body>
         <p> $ pcdata (string_of_int (a+b)) $ </p>
       </body>
     </html> >>
```

サービスとリンク・フォームの型付け

- Eliom において，URI は「サービス」と対応づける
- サービスは，リクエストパラメータの型をもつ
- サービスと，リンク・フォームから送信されるリクエストパラメータとの整合性は型で保証される

Eliomサービスの登録と実行



(1) Ocsigenがモジュールmを動的ロードすると
Mのトップレベルが実行され,
Ocsigenのpublicテーブルに, URIとサービスが登録される.

(2) ユーザが URI をリクエストすると,
public テーブルで解決されたサービスが動作する.

Eliomサービスの登録:
こんな風に書きます

```
let hello =
  register_new_service
    ~path:["hello"]
    ~get_params:(string “name”)
    (fun _ name _ ->
      return
       <<   <html>  …
       <p> こんにちは、$ pcdata name $ さん </p>
            </html> >>)
```

register_new_service で
URI とサービスを定義&登録

→ http://localhost:8080/hello?name=keigoi

register_new_service:
3つのパラメータ

```
let adder =
  register_new_service
    ~path:["adder"]
    ~get_params:(int "a" ** int "b")
    (fun sp (a,b) _ ->
      return
        <<   <html>  ...
          <p> $ pcdata (string_of_int (a+b)) $ </p>
                </html> >>)
```

パス

パラメータ名と型

サービス本体

get_params引数でリクエストパラメータの名前と型を指定

サービス本体は型付きのパラメータを受け取れる

→ http://localhost:8080/adder?a=10&b=20 でアクセス

get_form関数(3引数)で型付きフォーム生成

サービス

[パラメータ名]を引数に,
フォーム内部のタグを
返す関数

```
get_form adder sp (fun (an, bn) ->
 div [
  pcdata "a:";
  int_input ~input_type:`Text ~name:an ()
  br ();
  pcdata "b:";
  int_input ~input_type:`Text ~name:bn ()
  br ()
 ]
```

整数の
<input>タグ

整数の
<input>タグ

[パラメータ名]引数と
型付き <input>タグで
パラメータの型不整合
を防止

a: 100
b: aaa

PG'OCaml : DBアクセスも型安全に

- OCaml に SQL を埋め込み
  -> prepared statement に変換される
- 型安全！
- らくらくDBアクセス！
- O/R mapping なんてヘヴィなものは要らない
- .NET の LINQ to SQL の簡易版のようなもの

まとめ:
Eliomのポイントおさらい

1. XHTML (XML) 型付け
   - valid な XHTMLのみを生成する
2. サービスとリンク・フォームの型付け
   - リクエストパラメータの不整合がない
3. DBアクセスも型安全
   - PG'OCaml を使った型安全DBアクセス

OCamlで表現力が高く、かつ安全な Webプログラミング

もっと知りたい方へ

- 簡単なサンプルを置きましたのでそちらをどうぞ
- サンプル(tar.gz): http://bit.ly/jaPEI
- サンプル解説：http://bit.ly/hhOd1
  - (http://d.hatena.ne.jp/keigoi/20090829/1251533218)
  - “ keigoi の日記 ” で web検索すれば出ます

ご清聴ありがとうございました。

おまけ:

説明しなかった機能

- POST
  - POST の URI をGETした時にどのサービスを動かすか(fallback)を指定する必要がある
- Coservice
  - URI に対して別のサービスを割り当てる (セッションで使う)
- セッション
  - 継続サーバーのように(も)書ける

おまけ:

ビルド / 配備

- Findlib / Omake を使えば簡単！
- ocamlfind でコンパイル (私がよく使うもの)

```
ocamlfind ocamlc -c -package
extlib,ocsigen,calendar,pcre,threads,pgocaml,pgocaml.statements,
ocsigen.xhtml.syntax -syntax camlp4o -thread ファイル名.ml
```

- これでXHTML構文/PG'OCaml構文も使えます
- ocsigen.conf (/myapp/ に配備するとき)

```
<site path="myapp" charset="utf-8">
    <eliom module='/path/to/ファイル名.cmo'/>
 </site>
```

おまけ:
OMakefile

```
OCAMLPACKS[] = extlib ocsigen calendar pcre threads pgocaml
pgocaml.statements ocsigen.xhtml.syntax

OCAMLDEPFLAGS += -syntax camlp4o
OCAMLCFLAGS   += -thread -syntax camlp4o

FILES[] = ファイル名1 ファイル名2

LIB = ライブラリ名
.DEFAULT: $(OCamlLibrary $(LIB), $(FILES))
```

- できた ライブラリ名.cma を ocsigen.conf に書く

おまけ:

Java Webプログラマから見た Eliom

- 基本的なエラーはコンパイル時かサーバ開始時に殆ど検出される
- 必要なコード量が圧倒的に少ない！
  - 設定ファイル不要
    - Struts: web.xml, struts-config.xml, ...
    - Ocsigen: ocsigen.conf
- NullPointerException を意識しなくていい！

おまけ:

Eliom の難点 (1)

- 入力値検査 の エラーハンドリングが貧弱
- エラーの場合 フォームの再入力が必要
- (Struts1 の ActionForm のような仕組みが欲しい)
- パラメータが増えるとものすごく面倒

```
let entryform ?exns sp =
  post_form ~sp:sp ~service:entry (fun
      (username,(password,(passwordDummy,
      (kanjiFamilyName,(kanjiPersonalName, (...計40のパラメータ)
```

camlp4
拡張したい！

おまけ:

一見さんお断りな service型の複雑さ

- サービスの型 ... 長ッ！

```
(unit, string * (int32 * string),
   [> `Attached of
        [> `Internal of [ `Coservice | `Service ] * [> `Post ] ]
        a_s ],
 [ `WithoutSuffix ], unit,
 [ `One of string ] param_name *
 ([ `One of int32 ] param_name *
  [ `One of string ] param_name),
 [> `Registrable ])
service
```

おまけ:

service型のパラメー

```
('get, 'post, 'kind, 'tipo, 'getnames, 'postnames, 'registrable) service
```

- ‘get, ‘post .. リクエストパラメータ
- ‘kind ... attachedか否か, serviceかcoserviceか, etc.
- ‘tipo ... URI のサフィクスをリクエストパラメータにするか否か
- ‘getnames, ‘postnames ... リクエストパラメータの名前
- ‘registrable ... register できるか否か

(果たしてこれだけのパラメータが本当に必要か?)

おまけ:

OCamlDuceとの連携

- OCamlDuce：XMLの”型” (DTD, XML Schema, ...) を扱える OCamlの拡張
  - より XHTML DTD に忠実
  - 使ってません… 当面 XHTML.M で十分?
- 利点：XHTMLに対してパターンマッチできる
  - (Eliom の XHTML.M はできない)

おまけ:

PG'OCaml のクエリ例 (1)

```
let insert name age address =
  PGSQL(dbh)
    "INSERT INTO mytable(name,age,address) VALUES ($name,$age,$address)"
```

- コンパイル時に DB に接続し SQL から prepared statement を作る
  - ここでは name, age, address の型が推論される
- NOT NULL でない列は option 型になる,

おまけ:

PG'OCaml のクエリ例 (2)

```
let row dbh = PGSQL(dbh) "SELECT name,age,address FROM mytable"
```

- クエリの結果はタプルのリストで帰ってくる
- この関数の型は
  handle -> (string * int32 * string) list