Ur/Web:

Функциональный подход к разработке Web-приложений

Сергей Миронов
25.10.2014

Ur / Web

Ur
Статически типизированный язык общего назначения (Adam Chlipala)

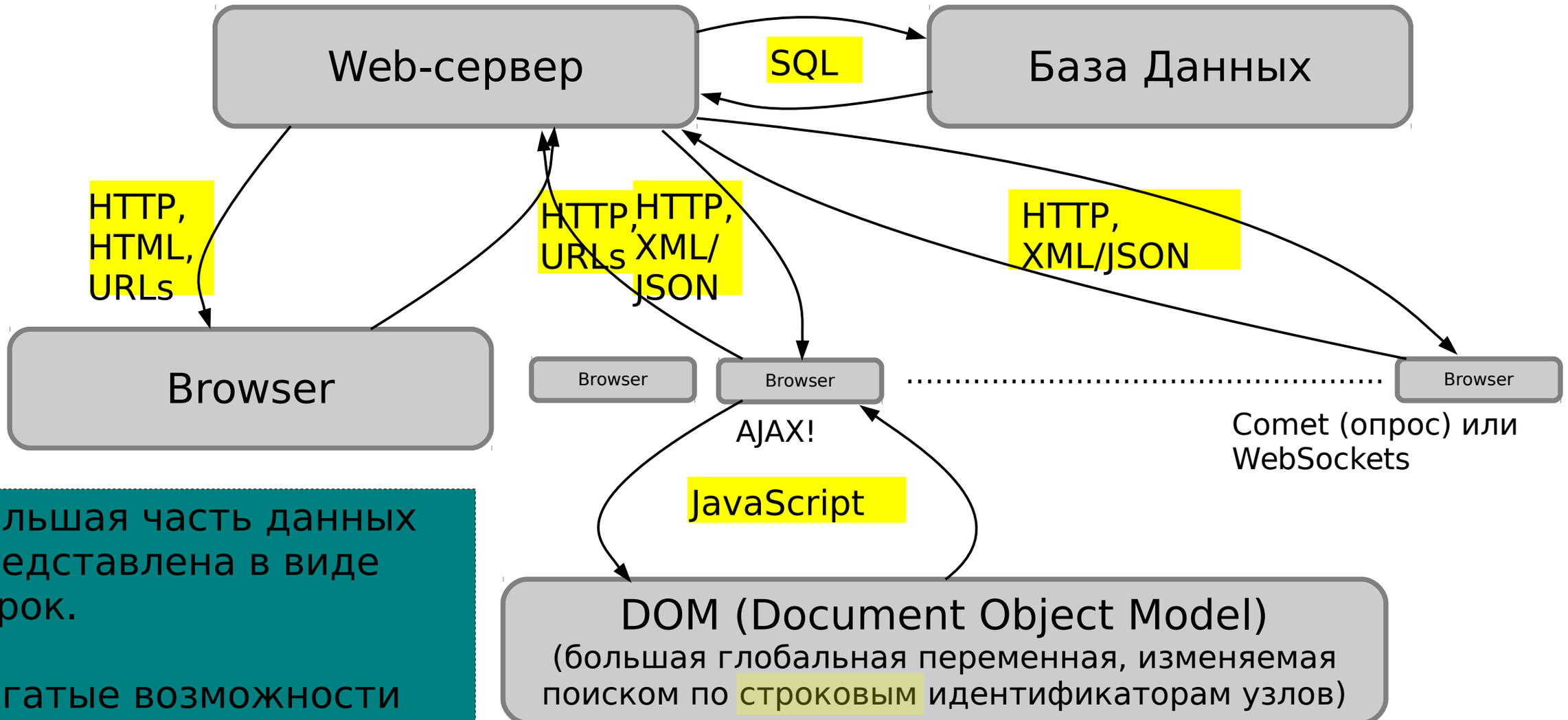
Web
Средства разработки веб-приложений

“Компилятор, который понимает структуру вашего веб-приложения”

Ориентирован на:
- **Удобство разработки**
- **Безопасность**
- **Производительность** (!)

Заимствованные черты:
Haskell: монады, typeclasses, do-notation
ML: структура модулей (structures, functors)

Классическая (?) модель работы Веб-приложения



Большая часть данных представлена в виде строк.

Богатые возможности для Code Injection

Удобство разработки?

Статическая типизация всех ключевых элементов Web-приложения

Компилятор проверяет корректность приложения и сообщает об ошибках соответсвия типов данных

Выразительная система типов, обеспечивает, в числе прочего, проверку корректности HTML кода на этапе компиляции

(к сожалению, полный рассказ о системе типов занял бы слишком много времени)

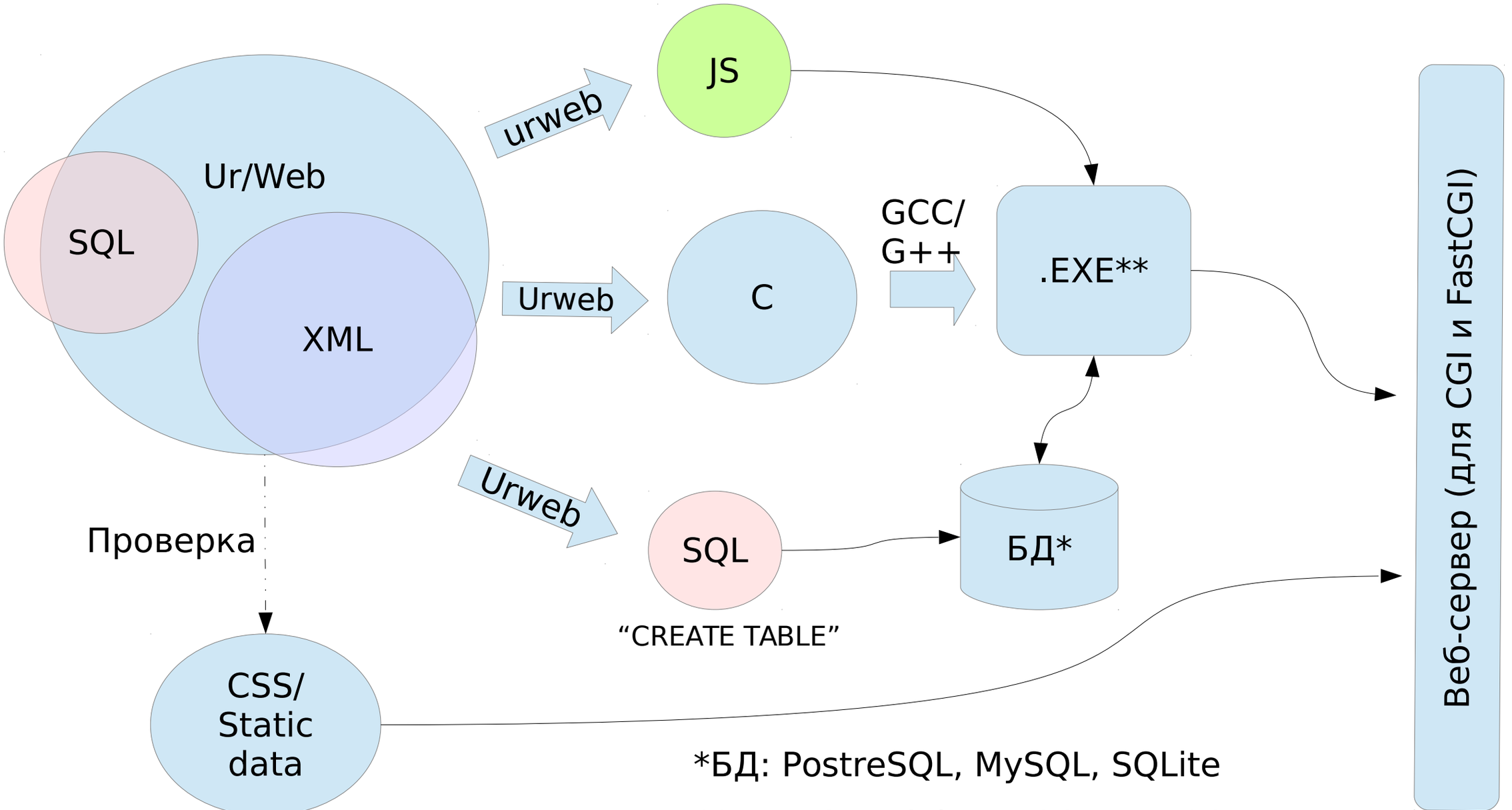
Security?

Распространенные уязвимости заведомо исключены: язык поддерживает шифрование cookie, констролирует используемые url, предотвращает code-injection.

Производительность?

Комиплятор генерирует нативный код C, часто оказывающийся эффективнее, чем аналогичный код, написанный “в ручную”.

*Ur/Web использует простую **транзакционную модель многозадачности**, которая имеет свои “слабые места”.



*БД: PostreSQL, MySQL, SQLite

**EXE: Standalone, CGI, FastCGI

Пример: HelloWorld

```
table t : {Id : int, Nam : string}

fun hello () : transaction page =
  return <xml>
    <head/>
    <body>
      <p><a link={world 0}>Hello what?</a></p>
    </body>
  </xml>

and world (id:int) : transaction page =
  r <- oneRow(SELECT * FROM t AS T WHERE T.Id = {[id]});
  return <xml>
    <head/>
    <body>
      <p>Hello, {[r.T.Nam]}!</p>
    </body>
  </xml>
```

- “Синтаксический сахар”

Кортеж с именованными полями (type-level record)

Монада transaction, Аналог IO-монады Haskell

Отложенный вызов ф-ии world

Константа типа [xml a b c]

Пример: Type-level records

```
type myRecord = record [A=int, B=float, C=string, D=bool]
val x:myRecord = { A = 0, B = 1.2, C = "world", D = True }
val v1 = x.A (* == 0: int *)
val v2 = x.C (* == ("hi": string)  *)

type myRecord2 = record [A=int, Z=xbody]
val y:myRecord2 = { A = 5,  Z = <xml><p>Hello, {[x.C]}</p></xml> }
val v3 = y.A (* == 5 *)

fun getA r : int = r.A
val v4 = getA x (* Работает, если убрать следующую строчку *)
val v5 = getA y (* Работает, если убрать предыдущую строчку *)
                (* Две строчки вместе вызывают ошибку проверки типов *)
```

```
fun getA [t::::{Type}] [t~[A]] (r:record (t++[A=int])) : int = r.A
val v4 = getA x (* == 0 *)
val v5 = getA y (* == 5 *)
```

Пример: Type-level records

```
type myRecord = record [A=int, B=float, C=string, D=bool]
val x:myRecord = { A = 0, B = 1.2, C = "world", D = True }
val v1 = x.A (* == 0: int *)
val v2 = x.C (* == ("hi": string)  *)

type myRecord2 = record [A=int, Z=xbody]
val y:myRecord2 = { A = 5,  Z = <xml><p>Hello, {[x.C]}</p></xml> }
val v3 = y.A (* == 5 *)

fun getA r : int = r.A
val v4 = getA x (* Работает, если убрать следующую строчку *)
val v5 = getA y (* Работает, если убрать предыдущую строчку *)
                (* Две строчки вместе вызывают ошибку проверки типов *)

fun getA [t:::{Type}] [t~[A]] (r:record (t++[A=int])) : int = r.A
val v4 = getA x (* == 0 *)
val v5 = getA y (* == 5 *)
```

Desugaring (1)

```
fun hello () : transaction page =
  let
    val a_desugared = tag
      null
      None
      noStyle
      None
      {Href = url(world 0)}
      (a {})
      (cdata "Hello")
  in
  return <xml>
    <head/>
    <body>
      <p>{a_desugared}</p>
    </body>
  </xml>
  end
```

Компилятор восстанавливает XML из функционального представления

Используются примитивы стандартной библиотеки:
tag (7 параметров), **null, a, noStyle, cdata, url**

Desugaring (2)

```
val t : sql_table [Id=int, Nam=string] [] = ???

(*
==typechecks==
val t : sql_table [Id=int, Nam=string] [] = error <xml/>
*)

fun hello () : transaction page =
  return <xml>
    <head/>
    <body>
      <p><a link={world 0}>Hello</a></p>
    </body>
  </xml>
```

Переменная t, представляющая таблицу, имеет тип sql_table [..] [..]. Значение не определено (не может использоваься напрямую).

Desugaring (3)

```
table t : {Id : int, Nam : string}

and world (id:int) : transaction page =
  r <- oneRow ((sql_query
    { Rows = sql_query1 [[]]
        { Distinct = False
        , From = sql_from_table [#T] t
        , Where = sql_binary sql_eq (sql_field [#T] [#Id]) (sql_inject id)
        , GroupBy = sql_subset_all [(_ :: {{Type}})]
        , Having = sql_inject True
        , SelectFields = sql_subset [[T = ((_ :: {Type}), [])]]
        , SelectExps = {}
        }
    , OrderBy = sql_order_by_Nil [(_ :: {Type})]
    , Limit = sql_no_limit
    , Offset = sql_no_offset
    })
  );
  return <xml>
    <head/>
    <body>
      <p>Hello, {[r.T.Nam]}</p>
    </body>
  </xml>
```

“Внедренная” переменная

Итоговый запрос SQL реконструируется компилятором из функционального представления (тот же метод, что и в случае XML)

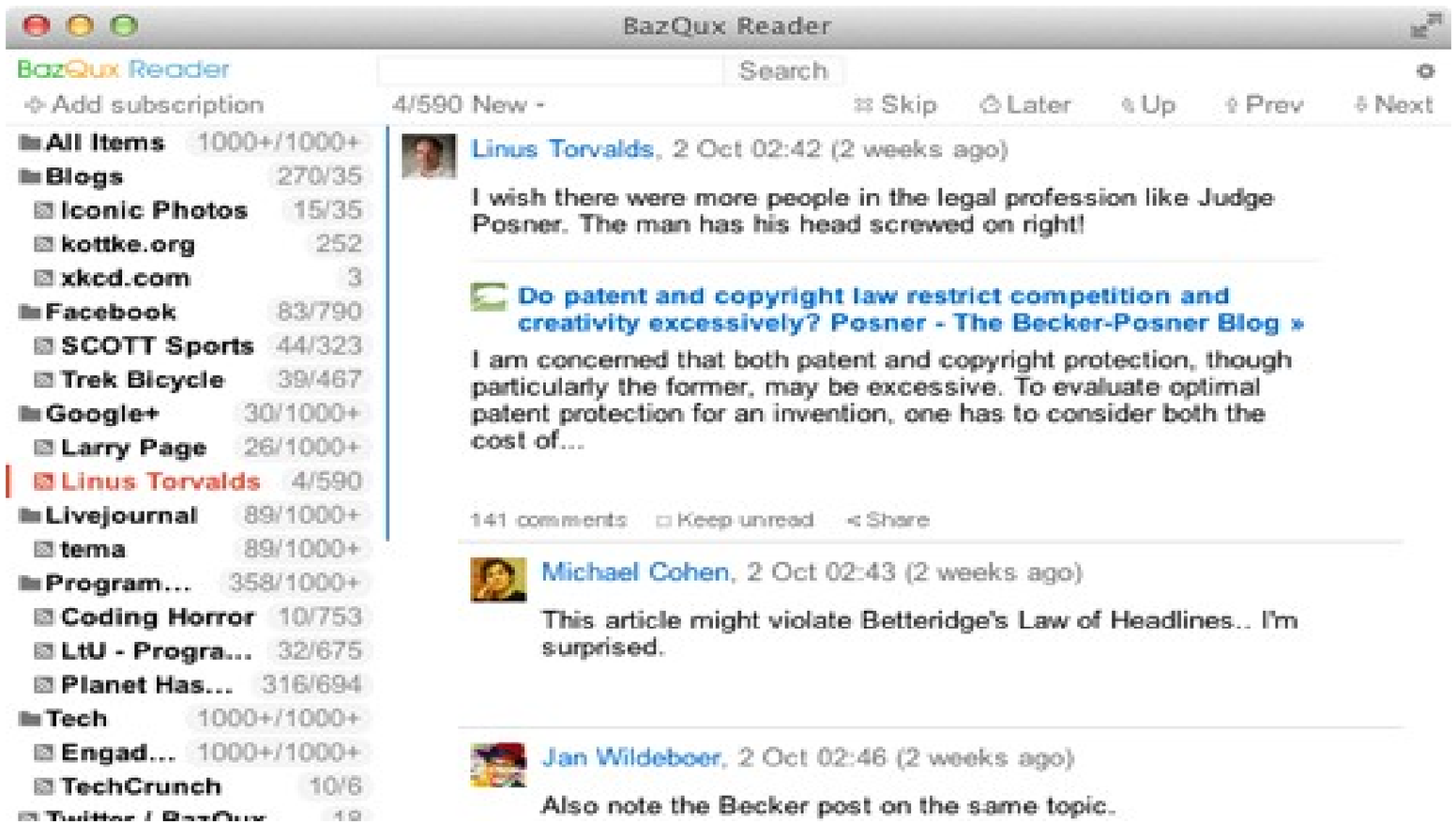
Опубликованные приложения

Первое коммерческое приложение Ur/Web:
BazQux Reader, автор – Владимир Шабанов

Новостной аггрегатор

http://www.bazqux.com/

1000 пользователе в день



Опубликованные приложения (продолжение)

- EcoSrv (http://ecosrv.hit.msk.ru)
- Bitcoin merge mining pool (http://mmpool.org/pool)
- Logitext (http://logitext.mit.edu)
- Big map of Latin America (http://map.historyisaweapon.com/)

Опубликованные библиотеки

- urweb-callback
- urweb-monad-pack
- uru3
- personna

- Openid
- E-mail
- Bootstrap
- uw-process

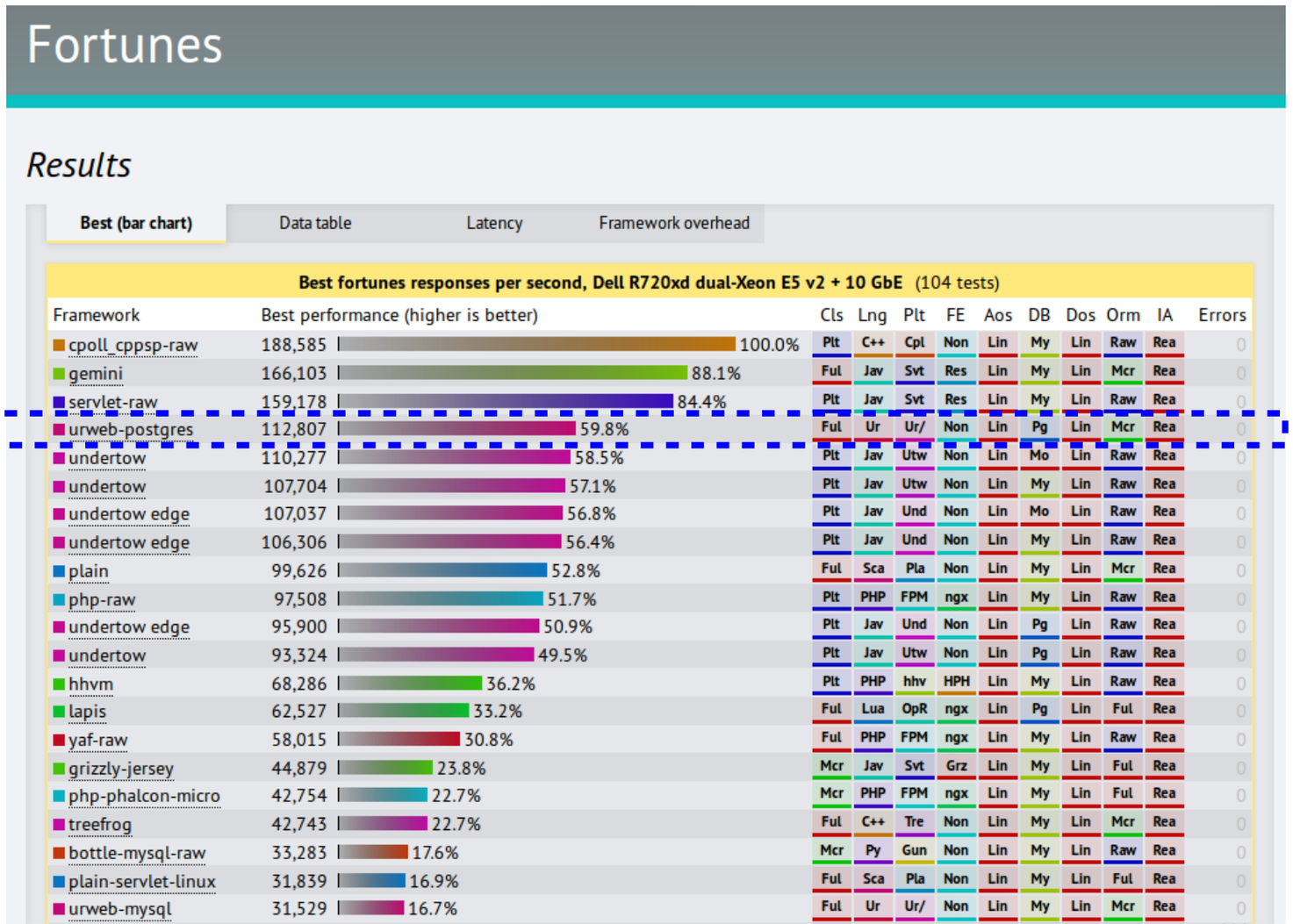
TechEmpower Web Framework Benchmarks

<http://www.techempower.com/benchmarks/>

- Независимое тестирование
- 93 участника конкурса
- Производительность Ur/Web:
  - 4-е место (~100 krps) в тесте на случайный записи
  - Минимальные задержки (2.1 ms)
- Есть простор для оптимизации :-)

Где же Haskell?

Почему-то никто не реализовал этот тест на Haskell!
~50% производительности Ur/Web's на других тестах (Snap, Wai, Yesod)



Fortunes

Results

Best (bar chart) | Data table | Latency | Framework overhead

Best fortunes responses per second, Dell R720xd dual-Xeon E5 v2 + 10 GbE (104 tests)

Framework	Best performance (higher is better)	Cls	Lng	Plt	FE	Aos	DB	Dos	Orm	IA	Errors
cpoll_cppsp-raw	188,585 100.0%	Plt	C++	Cpl	Non	Lin	My	Lin	Raw	Rea	0
gemini	166,103 88.1%	Ful	Jav	Svt	Res	Lin	My	Lin	Mcr	Rea	0
servlet-raw	159,178 84.4%	Plt	Jav	Svt	Res	Lin	My	Lin	Raw	Rea	0
urweb-postgres	112,807 59.8%	Ful	Ur	Ur/	Non	Lin	Pg	Lin	Mcr	Rea	0
undertow	110,277 58.5%	Plt	Jav	Utw	Non	Lin	Mo	Lin	Raw	Rea	0
undertow	107,704 57.1%	Plt	Jav	Utw	Non	Lin	My	Lin	Raw	Rea	0
undertow edge	107,037 56.8%	Plt	Jav	Und	Non	Lin	Mo	Lin	Raw	Rea	0
undertow edge	106,306 56.4%	Plt	Jav	Und	Non	Lin	My	Lin	Raw	Rea	0
plain	99,626 52.8%	Ful	Sca	Pla	Non	Lin	My	Lin	Mcr	Rea	0
php-raw	97,508 51.7%	Plt	PHP	FPM	ngx	Lin	My	Lin	Raw	Rea	0
undertow edge	95,900 50.9%	Plt	Jav	Und	Non	Lin	Pg	Lin	Raw	Rea	0
undertow	93,324 49.5%	Plt	Jav	Utw	Non	Lin	Pg	Lin	Raw	Rea	0
hhvm	68,286 36.2%	Plt	PHP	hhv	HPH	Lin	My	Lin	Raw	Rea	0
lapis	62,527 33.2%	Ful	Lua	OpR	ngx	Lin	Pg	Lin	Ful	Rea	0
yaf-raw	58,015 30.8%	Ful	PHP	FPM	ngx	Lin	My	Lin	Raw	Rea	0
grizzly-jersey	44,879 23.8%	Mcr	Jav	Svt	Grz	Lin	My	Lin	Ful	Rea	0
php-phalcon-micro	42,754 22.7%	Mcr	PHP	FPM	ngx	Lin	My	Lin	Ful	Rea	0
treefrog	42,743 22.7%	Ful	C++	Tre	Non	Lin	My	Lin	Mcr	Rea	0
bottle-mysql-raw	33,283 17.6%	Mcr	Py	Gun	Non	Lin	My	Lin	Raw	Rea	0
plain-servlet-linux	31,839 16.9%	Ful	Sca	Pla	Non	Lin	My	Lin	Ful	Rea	0
urweb-mysql	31,529 16.7%	Ful	Ur	Ur/	Non	Lin	My	Lin	Mcr	Rea	0

“Узкие места” и критика

- Отсутствие тэгирования серверного и клиентского кода на уровне типов (transaction server a/transaction client a)
- Ограничения при создания семейств классов, препятствующие созданию полноценных монад-трансформеров (можно выкрутиться с помощью выразительной системы модулей ML)
- Отсутствие инструментов для работы со статическими файлами (by design, автор пошел на небольшие уступки, в остальном отдано на откуп C-FFI)
- Необычная политика в области определения пользовательских HTML тэгов (всё ещё неполная поддержка HTML5)
- Требуется дополнительная работа для интеграции стороннего JavaScript(JavaScript-FFI)
- Слабый C/C++ FFI, отсутствие маршалинга сложных типов данных (автор считает, что степень поддержки достаточна)

Неосвященные пункты

- Работа с формами (POST-запросы)
- Удаленный вызов процедур
- Программирование пользовательского интерфейса (“реактивный” подход, сигналы)
- Взаимодействие клиентов между собой (каналы)
- Выполнение кода по таймеру
- Работа с CSS-стилями (элементарные проверки, отказ от ответственности)
- FFI, подключение стороннего кода на C/C++ и JavaScript

Спасибо за внимание

Веб-сайт проекта: http://www.impredicative.com/ur/
Вики: http://www.impredicative.com/wiki

[1] – A. Chlipala. Ur: Statically-typed metaprogramming with type-level record computation. In Proc. PLDI, pages 122–133. ACM, 2010.
[2] – A.Chlipala. Ur/Web: A simple model for programming the Web, Draft, 2014

Сергей Миронов
grrwlf@gmail.com
25.10.2014