Ottawa

ERLANG

Welcome!

June 12th 2013

Agenda

- Quick overview of web servers/ frameworks available in Erlang
    - Example using ChicagoBoss
- Quick overview of the Elixir language (built atop Erlang)
- From OOP to FP with Erlang (open discussion)

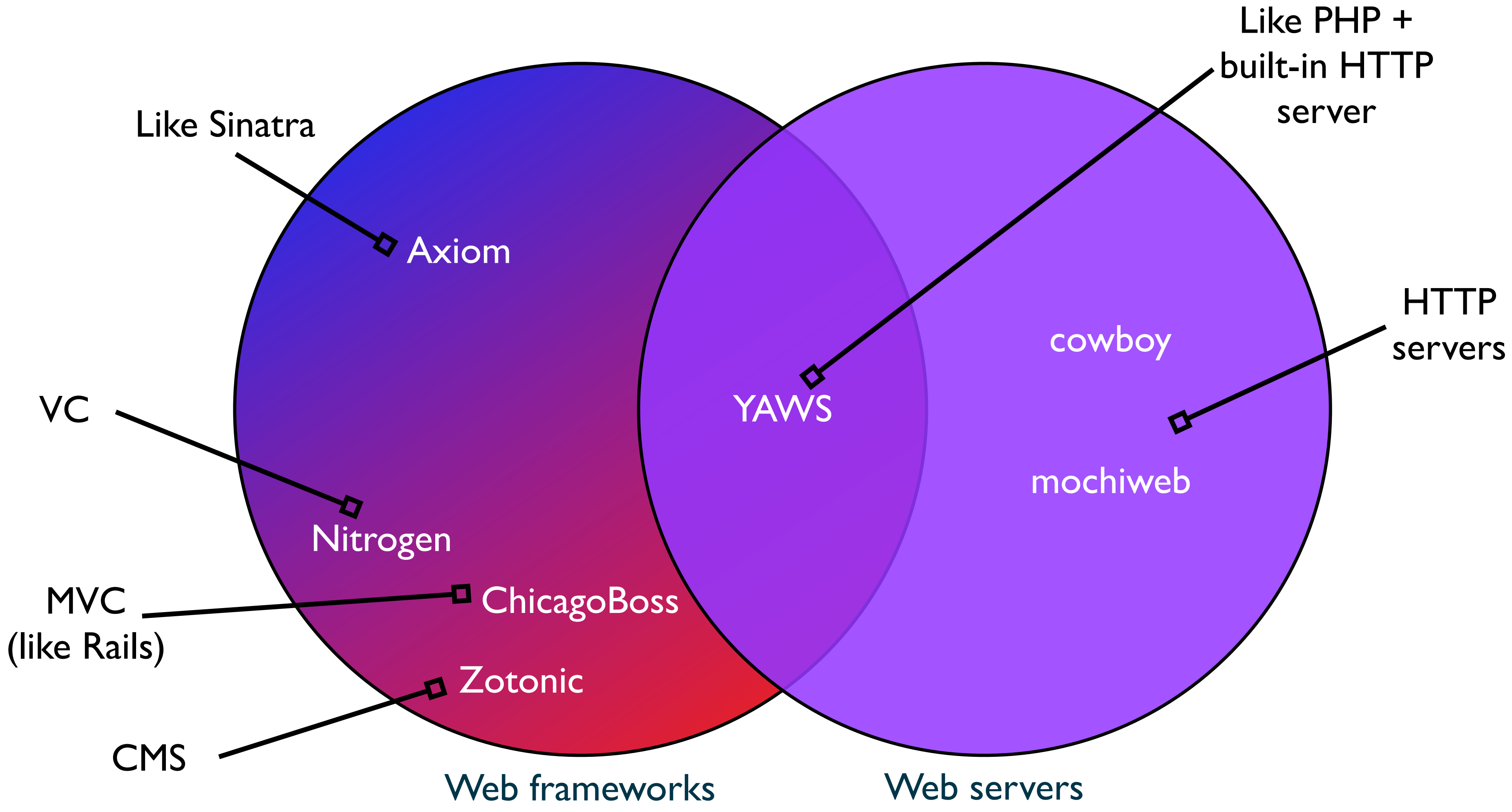
Erlang Web Frameworks/Servers

Quick Overview by Francis Joanis

Web Servers + Erlang

- Winning combination
  - Fully non blocking I/O easily exposed to programmers = scalable
  - Model requests as processes
    - Code like the real world
  - Data immutability = fast template performance (no string concatenation)

Some of the most talked about



ChicagoBoss

An Example by Francis Joanis

ChicagoBoss

- Project's goal
  - "Reduce a typical website's operational costs by 90%" - Evan Miller @ Erlang Factory SF 2013
    - http://www.youtube.com/watch?v=LGGo6bIuj8w

ChicagoBoss

- Like RoR (MVC) and:
  - Fast templating (due to immutability)
  - Lower memory usage
  - Highly scalable web server
  - Easy WebSockets / long polling

ChicagoBoss

- Django templates
- ORM
- Event notification
- Built-in email client/server
- Built-in message queue (Erlang cluster)
- Hot code loading / updating

The Example App

- Super Simple Personal Pet Registry
  - Add new pet
  - Edit existing pet
  - Remove existing pet
  - Browse existing pets

The Model

Name	Fluffy	Rex
Kind	Cat	Dog

The Model

- Animal names must be unique
- Animal names and kinds cannot be empty

Pet Registry + CB

- Prerequisite
  - ChicagoBoss installed + compiled
- Steps
  - Create new ChicagoBoss project
  - Implement MVC
  - Run dev environment

Creating a new CB project

- From under the CB directory
  - `make app PROJECT=pet_registry`
- Will create `../pet_registry` where the project lives
- `cd ../pet_registry`

Creating the Model

- cd src/model
- touch pet.erl

pet.erl

```
% Note the usage of parameterized modules
-module(pet, [Id, Name, Kind]).
-compile(export_all).
% Pet = pet:new(id, “Fluffy”, “Cat”).
% MyPet = boss_db:find(“pet-1”).
% MyPet:name(). % “Fluffy”
```

Creating the Views

- 3 views
  - Browse the list of pets
  - Edit existing pet
  - Add new pet

Creating the Views

- Views use Django templates
  - `src/view/registry/browse.html`
  - `src/view/registry/edit.html`
  - `src/view/registry/add.html`

Creating the Controllers

- Controllers are Erlang modules
  - cd src/controller/
    - pet_registry_registry_controller.erl

VC Source File Relationships

- Controller (src/controller)
  - pet_registry_registry_controller.erl
- View (src/view)
  - Placed under actual controller name
    - registry/[browse, add, edit].html

VC Source Code Relationships

```
-module(pet_registry_registry_controller, [Req]).

-compile(export_all).


browse('GET', []) ->

  {ok, []}. % Django variables would get set here for the browse view


edit('GET', []) -> % Method; 2nd argument is URI tokens (/edit/FIRST/SECOND)

  {ok, []}.


add('GET', []) ->

  {ok, []}.
```

Show The Code!

- MVC
- Validators
- Routes

Launching the dev environment

- ./init-dev.sh
  - You can change MVC source files and just refresh the pages - CB will recompile them on demand
  - Dev environment doesn't persist data
  - Built-in model documentation
    - http://localhost:8001/doc/pet

Launching the dev environment

- http://localhost:8001

To-dos

- Automated Tests using CB
- Pagination
- Add 4xx/5xx handlers
- cb_admin
- ...

Going to production

- Tweak boss.config
  - Connect to chosen production data store
  - ...
- Run the production server
  - `./init.sh start`

More info

- http://www.chicagoboss.org/tutorial.pdf
- https://github.com/evanmiller/ChicagoBoss/wiki/Quickstart
- http://www.chicagoboss.org/api.html