Chicago Boss

A web framework built for comfort (and speed)

Evan Miller

March 22, 2013

Erlang Factory San Francisco

Personal History

Personal History

- 2006-2007: Amazon Search Operations (Operational Excellence Engineer)

Personal History

- 2006-2007: Amazon Search Operations (Operational Excellence Engineer)
- 2007: Software Engineer @ IMVU

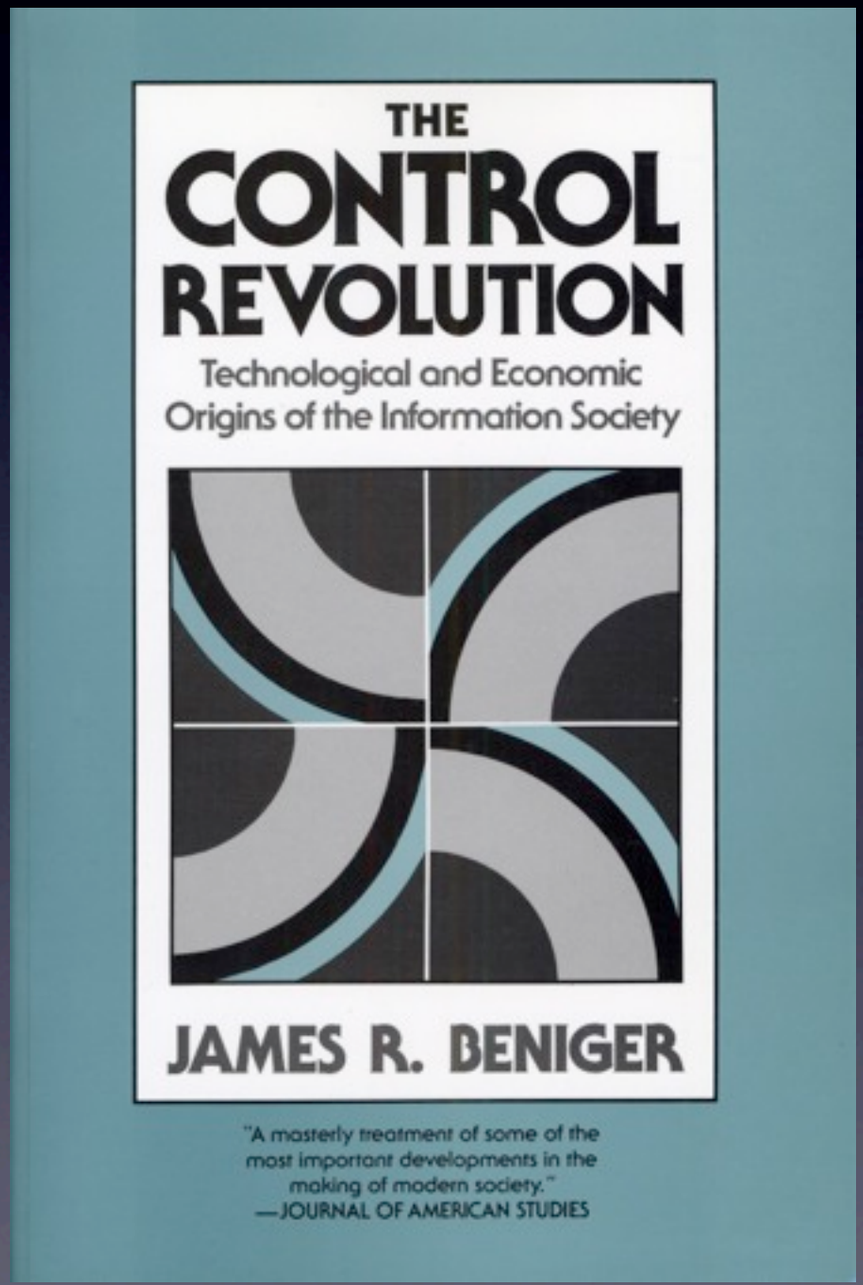
Personal History

- 2006-2007: Amazon Search Operations (Operational Excellence Engineer)
- 2007: Software Engineer @ IMVU
- 2008 - ???: Grad school

Why Chicago Boss?

Why Chicago Boss?

Why Chicago Boss?



Why Chicago Boss?

Why Chicago Boss?

Why Chicago Boss?

- Goal: Reduce a typical website's operational costs by 90%

Why Chicago Boss?

- Goal: Reduce a typical website's operational costs by 90%
- Foster non-profit, low-profit, and marginal websites. Enable more self-hosted stuff. Power and independence!

Why Chicago Boss?

- Goal: Reduce a typical website's operational costs by 90%
- Foster non-profit, low-profit, and marginal websites. Enable more self-hosted stuff. Power and independence!
- Enable a new generation of deployable open-source applications (blogs, forums, mailing lists, knowledge bases, wikis...)

What is Chicago Boss?

What is Chicago Boss?

- Chicago Boss : Ruby on Rails :: Nginx : Apache

What is Chicago Boss?

- Chicago Boss : Ruby on Rails :: Nginx : Apache
- Same overall goals, but:
  - Non-blocking network I/O
  - No-copy memory architecture

What is Chicago Boss?

- Chicago Boss : Ruby on Rails :: Nginx : Apache
- Same overall goals, but:
  - Non-blocking network I/O
  - No-copy memory architecture
- "Faster, Cheaper, Better"

ErlyDTL

Template Memory Architecture

Template Memory Architecture

	Ruby String	Erlang I/O List

Template Memory Architecture

	Ruby String	Erlang I/O List
Append Function	String.concat	[H|T]

Template Memory Architecture

	Ruby String	Erlang I/O List
Append Function	String.concat	[H|T]
Append Complexity	O(N)	O(1)

Template Memory Architecture

	Ruby String	Erlang I/O List
Underlying data structure	char *	char **

Template Memory Architecture

	Ruby String	Erlang I/O List
Underlying data structure	char *	char **
OS network write function	write (single write)	writev (scattered write)

Template Memory Architecture

	Ruby String	Erlang I/O List
Data Mutability	Mutable	Immutable

Template Memory Architecture

	Ruby String	Erlang I/O List
Data Mutability	Mutable	Immutable
Memory footprint	O(clients)	O(templates)

Template Memory Architecture

In Erlang, shared template snippets occupy same hunk of memory across multiple requests!

Template Memory Architecture

- Practical Benefits:
  - Tests run quickly. *Fast* development cycle.
  - Drastically cuts hardware requirements and operational headaches
  - No complex view caching. Feature set isn't constrained by ability to cache.

Erlang VM

100% Non-Blocking I/O

100% Non-Blocking I/O

	Blocking I/O (Apache / RoR)	Non-Blocking I/O (Nginx / Erlang)
# processes	many	one

100% Non-Blocking I/O

	Blocking I/O (Apache / RoR)	Non-Blocking I/O (Nginx / Erlang)
# processes	many	one
Memory per concurrent request	MB	KB

100% Non-Blocking I/O

Node.js

```
mysql_read(
 function(retval1) {
  mysql_read(
   function(retval2) {
     ...
   });
});
```

Erlang

```
Result1 = mysql:read(),
Result2 = mysql:read().
```

100% Non-Blocking I/O

- Erlang's callback-free design is pure genius
- Engineering benefits:
  - Eliminates context switches
  - Lower RAM requirements
  - Better CPU cache-hit rate

100% Non-Blocking I/O

- Practical benefits:
  - Makes it hard *not* to write a high-performance server
  - WebSockets and real-time notifications "for free"
  - Build lots of services into one server (web + email + Jabber + ...)

Chicago Boss: Features

Chicago Boss: Features

- High-performance web server
    - Django templates (99% compatible)
    - Rails-like controller and model API
    - WebSockets and long polling
- Built-in email server (send and receive)
- Built-in cluster-wide message queue

Chicago Boss: Features

- i18n support
    - UTF-8 source code
    - PO file editor + {% trans %} tag
- Admin interface for editing database
- Rich data modeling (validations + associations)
- Hot code loading and all that

BossDB

BossDB

- Only (?) database abstraction layer for Erlang
- Many databases supported, easy to add others
  - MySQL
  - PostgreSQL
  - Mnesia
  - Tyrant
  - DyanmoDB
  - Riak
  - MongoDB

BossDB

- Models are parameterized modules (boo-hoo)
- Accessor functions are automatically generated

```
Author:first_name().
```

- Associations

```
-belongs_to(author).
```

- Compatible with ErlyDTL

```
{{ message.author.first_name }}
```

BossDB

- Rails conventions
  - Plural table names, column named `id`
- Rich, language-integrated querying

```
boss_db:find(message,
             [contents = “Hello!”])
```

- Built-in caching and connection pooling

BossDB

- Built-in event system (BossNews)

```
boss_news:watch(
        “message-42.contents”,
        CallBack)
```

- Events expire out-of-date cache entries
- BossDB is great. More projects should use it.

Example Code

Hello, World

```
-module(my_test_controller, [Req]).
-compile(export_all).

index('GET', []) ->
    {output, "Hello, world!"}.
```

Hello, Template

- index.html:

```
A message for you: {{ msg }}
```

- Controller code:

```
index('GET', []) ->
    {ok, [{msg, "Hello, world!"}]}.
```

Hello, Model

- message.erl:

```
-module(message, [Id, Contents]).
```

- Controller code:

```
index('GET', []) ->
    Msg = message:new(id, "Hello!"),
    {ok, [{msg, Msg:contents()}]}.
```

Routing

Routing

- `GET /message/show/message-42`
  - Controller: `message`
  - Action: `show`
  - Token list: `["message-42"]`

Routing

- Custom routes with regular expressions

```
{“/show/(message-\d+)”, [
    {controller, “message”},
    {action, “show”},
    {id, ‘$1’}]}.
```

- Support for named capture groups

```
{“/show/(?<msg_id>message-\d+)”, [
    {controller, “message”},
    {action, “show”},
    {id, ‘$msg_id’}]}.
```

Routing

Parameter names are inferred from the parse tree!

Routing

```
[
      {controller, “message”},
      {action, “show”},
      {message_id, ‘$1’}
]
```

Matches

```
show(‘GET’, [MessageId]) ->
    ...
```

Thanks

Libraries Used

- Aleppo *
- BossDB *
- BSON
- Cowboy
- DDB
- dynamic_compile
- Elixir
- epgsql
- erlmc
- ErlyDTL *
- gen_server2
- gen_smtp
- ibrowse
- Jaderl *
- JSX
- Lager
- medici
- mimetypes
- misultin
- mochicow
- mochiweb
- mongodb
- mysql
- pmod_transform
- poolboy
- proper
- protobuffs
- ranch
- riakc
- SimpleBridge
- TinyPQ *
- TinyMQ *
- uuid

* Originally developed for Chicago Boss

Find Us

@chicagoboss on Twitter

chicagoboss@googlegroups.com

#chicagoboss on irc.freenode.net

http://www.chicagoboss.org/