Playframework, Realtime Web

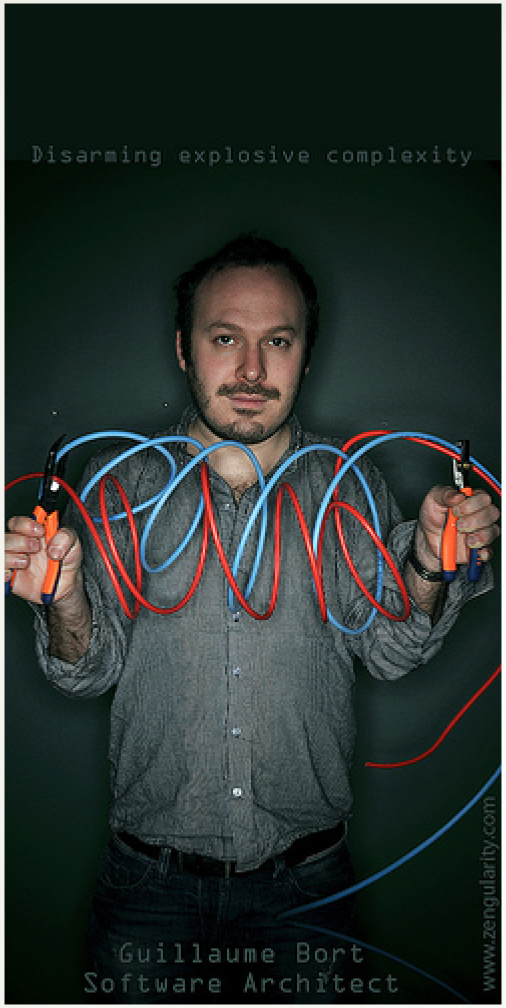
Sadek Drobi

ZENEXITY

Sadek Drobi

Play2 co-creator
CTO @ zenexity

@sadache



@guillaumebort

@hguergachi

Web Oriented Architectures

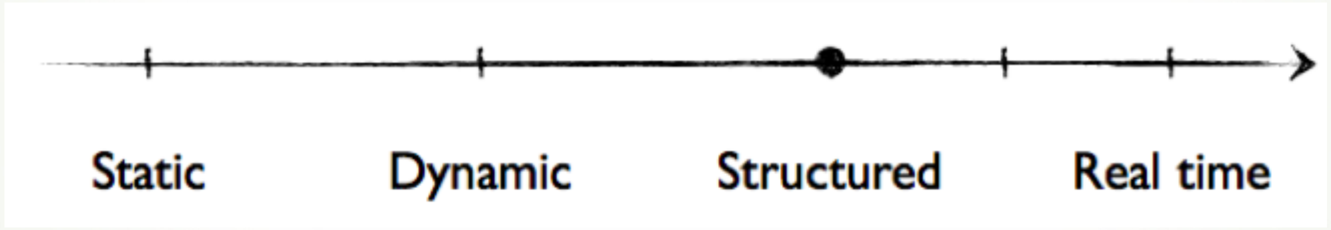
What is Playframework?

The Play framework makes it easier to build web applications with Java & Scala.

Play is based on a lightweight, stateless, web-friendly architecture for highly-scalable and realtime web applications

- thanks to its reactive model, based on Futures and Iteratee IO.

The Web Evolved



What is Realtime Web?

Wikipedia: The real-time web is a set of technologies and practices that enable users to receive information as soon as it is published by its authors, rather than requiring that they or their software check a source periodically for updates.

What is Realtime Web?

- Long polling
- Comet, Http 1.1 chunked and hacks
- Websockets
- Server Sent Events

Inspiration, examples of RTW with Play!

Connected peers exchanging data (a chat room, peer games, ...)

Streaming information (systems monitoring, datastore updates, ...)

Collecting Map/Reduce

Web Streams API mashups

A mix of these

Inspiration, examples of RTW with Play!

http://en.lichess.org/

http://console-demo.typesafe.com/

http://live.gilt.com

Why another web framework?

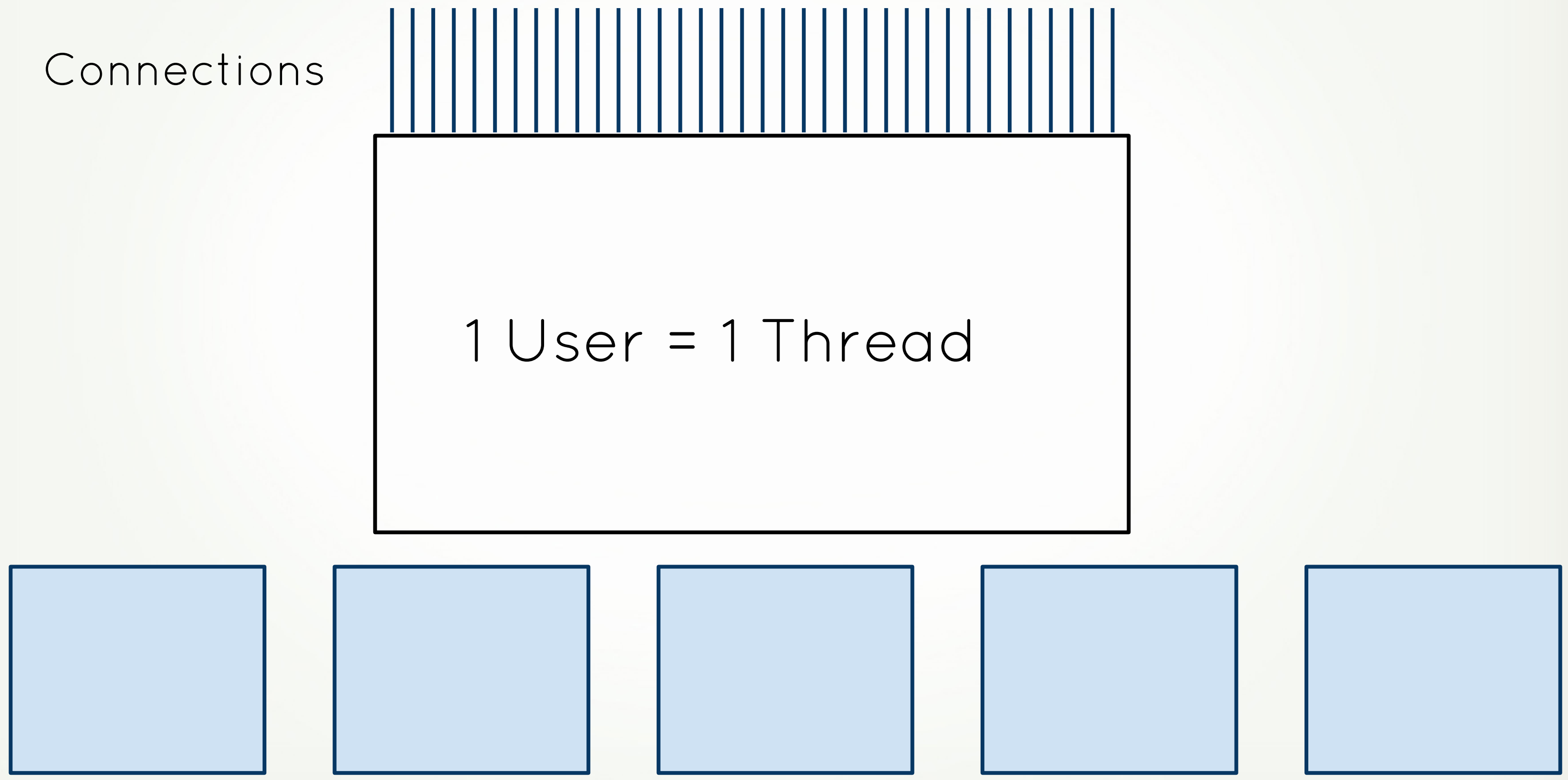
What's wrong with traditional WAR servers?

Some context...

Why another web framework?

1 User = 1 Thread

Why another web framework?



+ More IO, longer connections =?

Why another web framework?

Non-blocking

Reactive

Why didn't I say Asynchronous?

Streams on top of non-blocking, reactive architecture

```
public static WebSocket<String> index() {
 return new WebSocket<String>() {

   // Called when the Websocket Handshake is done.
   public void onReady(WebSocket.In<String> in, WebSocket.Out<String> out) {

     // For each event received on the socket,
     in.onMessage(new Callback<String>() {
        public void invoke(String event) {

          // Log events to the console
          println(event);
        }
     });

     // When the socket is closed.
     in.onClose(new Callback0() {
        public void invoke() {

          println("Disconnected");
        }
     });

     // Send a single 'Hello!' message
     out.write("Hello!");

   }
```

And that's about it?

You take the blue pill - the story ends, I walk you through a couple of examples and you do your best with what you got.

You take the red pill - you stay in Wonderland and I show you how deep the rabbit-hole goes.

I am only offering you the truth

You take the blue pill - the story ends, I walk you through a couple of examples and you do your best with what you got.

You take the red pill - you stay in Wonderland and I show you how deep the rabbit-hole goes.

What are Streams?

How do we talk about them?

Can I assign them to variables?

Can I adapt them?

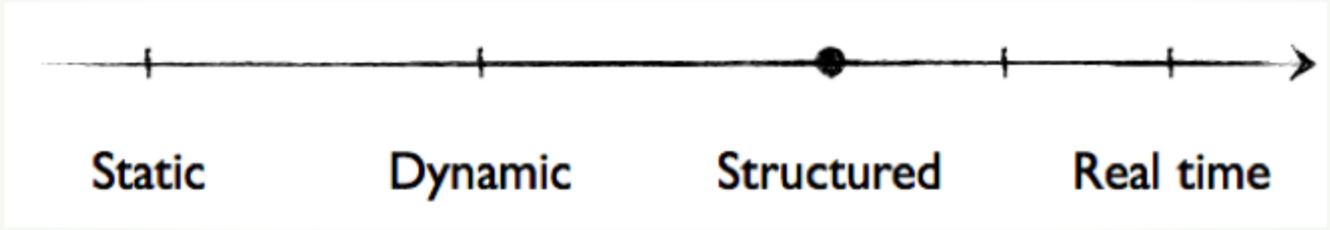
Is this what we really need to deal with Streams?

```
  });

  // Send a single 'Hello!' message
  out.write("Hello!");

}
```

Deja Vu?

```
  });

  // Send a single 'Hello!' message
  out.write("Hello!");

}
```

Deja Vu?



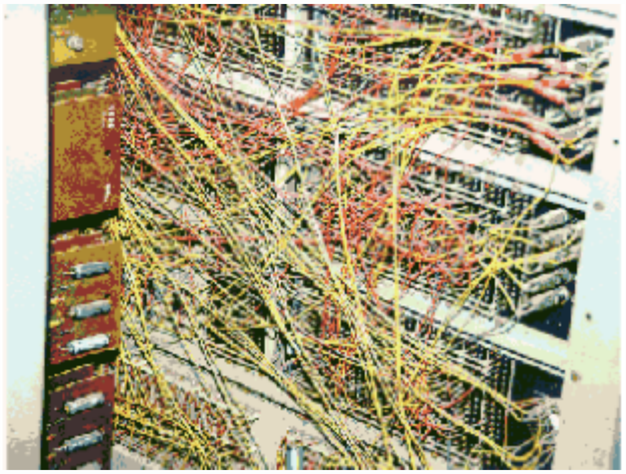
Deja Vu?

```
public class HelloWorld extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse res)
                              throws ServletException, IOException {

    res.setContentType("text/html");
    PrintWriter out = res.getWriter();

    out.println("<HTML>");
    out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<BIG>Hello World</BIG>");
    out.println("</BODY></HTML>");
  }
}
```

What about?

```
// For each event received on the socket,
in.onMessage(new Callback<String>() {
  public void invoke(String event) {

    // Log events to the console
    println(event);
  }
});
```

Callback Hell!



What are Streams?

How do we talk about them?

Can I assign them to variables?

Can I adapt them?

InputStream / OutputStream

But they are blocking and lack composition facilities.

Did industry move backwards?

What are Streams?

Streams as first class citizen, not represented as events and methods.

Streams, Adapters, Sinks

Reactive Streams

```
// A Stream for tweets
Enumerator<String> tweets = ...

// An Adapter
Enumeratee<String,JsValue> toJson = Enumeratee.map { s ->
Json.parse(s) }

// Adapt the stream
Enumerator<JsValue> tweetsAsJson = tweets.through(toJson)

// Another stream
Enumerator<JsValue> someAds = ...

Enumerator<JsValue> mainStream = tweets.interleave
(someAds).through(EventSource)
```

It does already exist for Scala!

```
// A Stream for tweets
val tweets:Enumerator[String] = ...

// An Adapter
val toJson: Enumeratee[String,JsValue] =
  Enumeratee.map { s -> Json.parse(s) }

// Adapt the stream
val tweetsAsJson: Enumerator[JsValue] =
  tweets.through(toJson)

// Another stream
val someAds: Enumerator[JsValue]= ...

val mainStream: Enumerator[JsValue] = tweets.interleave
(someAds).through(EventSource)
```

It does already exist for Scala!

A powerful reactive Stream manipulation API for doing serious Realtime Web applications, with sophisticated logic and rules!

We don't just give you the possibility, we give you the tools!

And it is coming to Java 8!

A powerful reactive Stream manipulation API for doing serious Realtime Web applications, with sophisticated logic and rules!

We don't just give you the possibility, we give you the tools!

Questions?

- www.playframework.com
- Check and play with the provided samples
- Ask