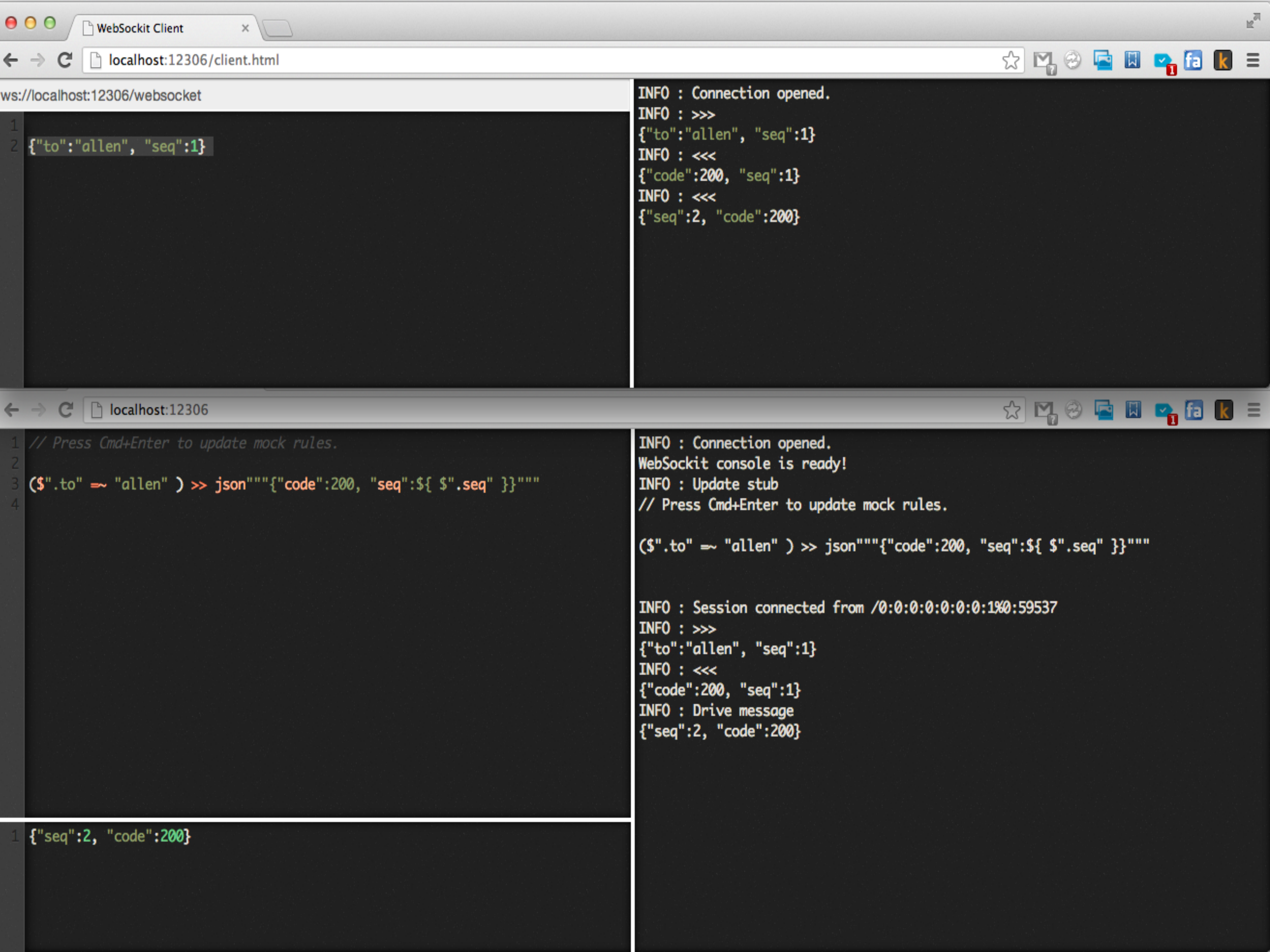
我的 Scala 实践总结

zhongl

Scala DSL

https://github.com/zhongl/websockit



ws://localhost:12306/websocket

```
{"to":"allen", "seq":1}
```

```
INFO : Connection opened.
INFO : >>>
{"to":"allen", "seq":1}
INFO : <<<
{"code":200, "seq":1}
INFO : <<<
{"seq":2, "code":200}
```

```
// Press Cmd+Enter to update mock rules.

($".to" =~ "allen" ) >> json"""{"code":200, "seq":${ $".seq" }}"""

```

```
{"seq":2, "code":200}
```

```
INFO : Connection opened.
WebSockit console is ready!
INFO : Update stub
// Press Cmd+Enter to update mock rules.

($".to" =~ "allen" ) >> json"""{"code":200, "seq":${ $".seq" }}"""


INFO : Session connected from /0:0:0:0:0:0:0:1%0:59537
INFO : >>>
{"to":"allen", "seq":1}
INFO : <<<
{"code":200, "seq":1}
INFO : Drive message
{"seq":2, "code":200}
```

```
($".to" =~ "allen" && $".seq" > 25) >> json"""
   {"code":200}
"""
```

twitter util eval is cool

@fujohnwang

JSONPath

http://goessner.net/articles/JsonPath/index.html#e2
https://code.google.com/p/json-path/

$”.Magic”

String Interpolation

Implicit Classes

Regular Expressions Interpolation in Pattern Matching

Demo

Trait

案例两则

用户的职责拆装

```
class User extends Actor {
  private var authorized = false
  def receive = {
    case Register(profile)    =>
    case Update(profile)      =>
    case Login(id, passwd)    =>
    case Logout               =>
    case Friends              =>
    case Message(to, payload) =>
  }
}
```

随着场景复杂度的变化,case会极速膨胀,如何破?

```
object StateMachine {
  trait State
  case object Connected extends State
  case object Authorized extends State

  trait Data
  case object Anonymous extends Data
  case class Identity(v: Int) extends
Data
}
```

```
import StateMachine._

trait Security extends FSM[State,Data] {
  startWith(Connected, Anonymous)

  when(Connected) {
    case Event(Login(id, passwd), Anonymous) =>
          goto(Authorized) using Identity(id)
  }
  when(Authorized) {
    case Event(Logout, Identity(v)           =>
          goto(Connected)  using Anoymous
  }
  case class Login(id: String, passwd: String)
  case object Logout
}
```

```
import StateMachine._

trait Profile extends FSM[State,Data] {

  when(Connected) {
    case Event(Register(id, passwd),Anonymous) =>
          stay
  }
  when(Authorized) {
    case Event(Update(passwd),Identity(v))     =>
          stay
  }

  case class Register(id: String, passwd: String)
  case class Update(passwd: String)
}
```

```
import StateMachine._

trait Network extends FSM[State,Data] {

  when(Authorized) {
    case Event(Friends, Identity(v))         =>
          stay
    case Event(Message(to, data),Identity(v) =>
          stay
  }

  case object Friends
  case class Message(to: String, payload: String)
}
```

```
import StateMachine._

class User extends FSM[State,Data] {

  whenUnhandled {
    case Event(s, d) => log.warn(...); stay
  }
  initialize()
}

object User {
  def props = Props(new User with Security
                             with Profile
                             with Network)
}
```

集中化依赖

```
import org.json4s._
import org.json4s.jackson.JsonMethods._
import org.json4s.JsonDSL._

class Command(name: String) {
  def unapply(s: String): Option[JValue] = {
    try {
      val j = parse(s)
      if (j \ "cmd" == name) Some(j) else None
    } catch { case _: Throwable => None }
  }
  implicit val formats = DefaultFormats
  def ack(code: Int) = ("code" -> code)
}
```

```
import org.json4s._
import org.json4s.jackson.JsonMethods

trait Json extends JsonDSL
           with JsonMethod
           with DoubleMode {
  type JValue = org.json4s.JValue
  implicit val formats = DefaultFormats
}
```

```
class Command(name: String) extends Json {
  def unapply(s: String): Option[JValue] = {
    try {
      val j = parse(s)
      if (j \ “cmd” == name) Some(j) else None
    } catch { case _: Throwable => None }
  }

  def ack(code: Int) = (“code” -> code)
}
```

Akka Actor

理解 Actor 的生命周期

小心 Restart 给你来个 NPE

```
class Dummy(r: Resource) extends Actor
```

理解 Option 与 Either

Effective SBT

Thanks