

# Some(Scala) @ GrowingIO

— Hawstein(邱嘉和)



# About Me

Scala Programmer, Akka Contributor

Backend Manager @ GrowingIO





# About GrowingIO

To B data analytics company

Backend Tech Stack: Scala & its friends

<https://www.growingio.com>

**GrowingIO**



# Agenda

- ◆ Backend Evolution
- ◆ Service Architecture
- ◆ Play / Akka
- ◆ Dive into dpx online service

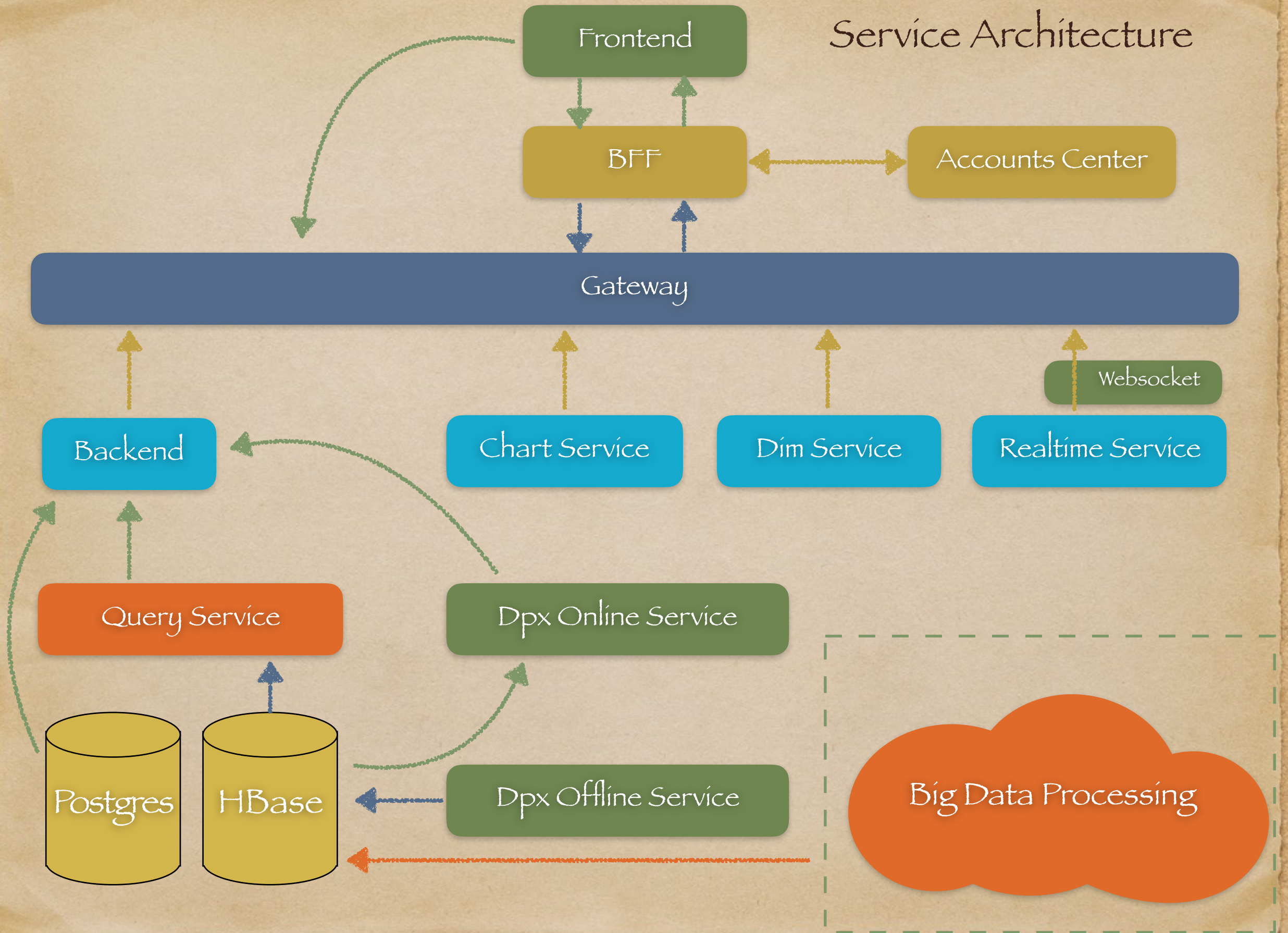


# Backend Evolution

- ◆ One monolithic app built on Play
- ◆ Frontend/Backend separation
- ◆ Separate Accounts from Backend
- ◆ Add Realtime Service
- ◆ Add Dim Service
- ◆ Separate Chart Service from Backend



# Service Architecture





- ◆ Frontend: Javascript / React
- ◆ BFF: Scala / Play
- ◆ Accounts Center: Scala / Play
- ◆ Gateway: Lua / Openresty
- ◆ Backend: Scala / Play / Akka
- ◆ Chart Service: Scala / Akka / Akka Http
- ◆ Dim Service: Scala / Play
- ◆ Realtime Service: Scala / Disruptor
- ◆ Query Service: Scala / Scalatra
- ◆ Dpx Online Service: Scala / Akka / Akka Http
- ◆ Dpx Offline Service: Scala / Akka Stream



# Play

- ◆ Full Stack Web Framework
- ◆ Template engine
- ◆ Non-blocking
- ◆ Embrace DI
- ◆ Test coverage



# Play - Full Stack Web Framework

- ◆ You get everything to build a web service
- ◆ Rich 3th party plugins
- ◆ Scaffold
- ◆ Community



**Play Framework** Shared publicly  
34 of 23902 topics (99+ unread)

Welcome to the Play Framework mailing list!

This list is the main place for general discussion about Play

New members are moderated to prevent spam. Please see

If you think you've found a bug or have a feature request, it

If you're trying to get help, please be polite and provide as much information as possible to help us understand the problem. Bear in mind that there are no guarantees that anyone here will be able and willing to help you, especially if it is specific to your application. If you want production support with guaranteed response times, you can purchase support from [Lightbend](#).

We encourage you to search the list for relevant posts, but please do not simply reply to every post that is at all related to your issue, as that just creates noise. It's usually best to create a new post explaining your particular problem in detail, and reference the old posts if you think they are relevant.

All members of the community are expected to abide by our [Code of Conduct](#). We will not tolerate attacks on group members or destructive negative commentary. Anyone engaging in such behavior will be banned at the sole discretion of the group owners. Anyone engaging in antisocial or illegal behavior including (but not limited to) spamming, posting binaries or images, or impersonating other users may also be banned at the sole discretion of the group owners.

- Play 2.6.0-M3 released!** (1)  
By Greg Methvin - 1 post - 24 views Mar 25
- Play 2.6.0-M2, 2.5.13 and 2.4.11 released!** (3)  
By Marcos Pereira - 5 posts - 549 views Mar 17
- New members are moderated**  
By Will Sargent - 13 posts - 1116 views Feb 7
- Guide to making a REST API with Play** (5)  
By Will Sargent - 5 posts - 893 views 9/20/16
- ApplicationLifecycle in 2.6.0-M1 & -M2** (3)  
By David - 3 posts - 5 views

**playframework / playframework** Watch 795 Unstar 9,101 Fork 3,153

Code Issues 242 Pull requests 27 Projects 0 Pulse Graphs

Play Framework <http://www.playframework.com/>

scala java reactive web-framework restful play playframework

8,107 commits 7 branches 120 releases 593 contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

```

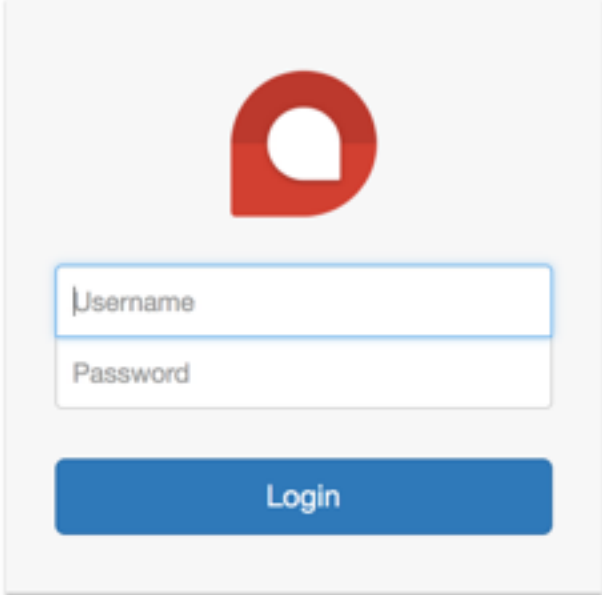
Browse the list of templates: http://typesafe.com/activator/templates
Choose from these featured templates or enter a template name:
1) minimal-akka-java-seed
2) minimal-akka-scala-seed
3) minimal-java
4) minimal-scala
5) play-java
6) play-scala
(hit tab to see a list of all templates)
>
Display all 517 possibilities? (y or n)

```



# Play - Template engine

```
1 > @error: Option[String] = None)
2
3 @layout.fixed("Login") {
4
5   <div class="row">
6     <div class="col-sm-6 col-md-4 col-md-offset-4">
7       <div class="account-wall">
8         
9         <form class="form-signin" action="@routes.UserController.login()" method="post">
10          <input type="text" class="form-control" name="username" placeholder="Username" required autofocus>
11          <input type="password" class="form-control" name="password" placeholder="Password" required>
12          <span style="display: block; text-align: center; margin-top: 10px;">
13            @if(error.isDefined) {
14              @Html(error.get)
15            }
16          </span>
17          <button class="btn btn-lg btn-primary btn-block" type="submit">Login</button>
18        </form>
19      </div>
20    </div>
21  </div>
22
23 }
24
```



A login form with a red logo at the top. Below the logo are two input fields: "Username" and "Password". At the bottom is a blue button labeled "Login".

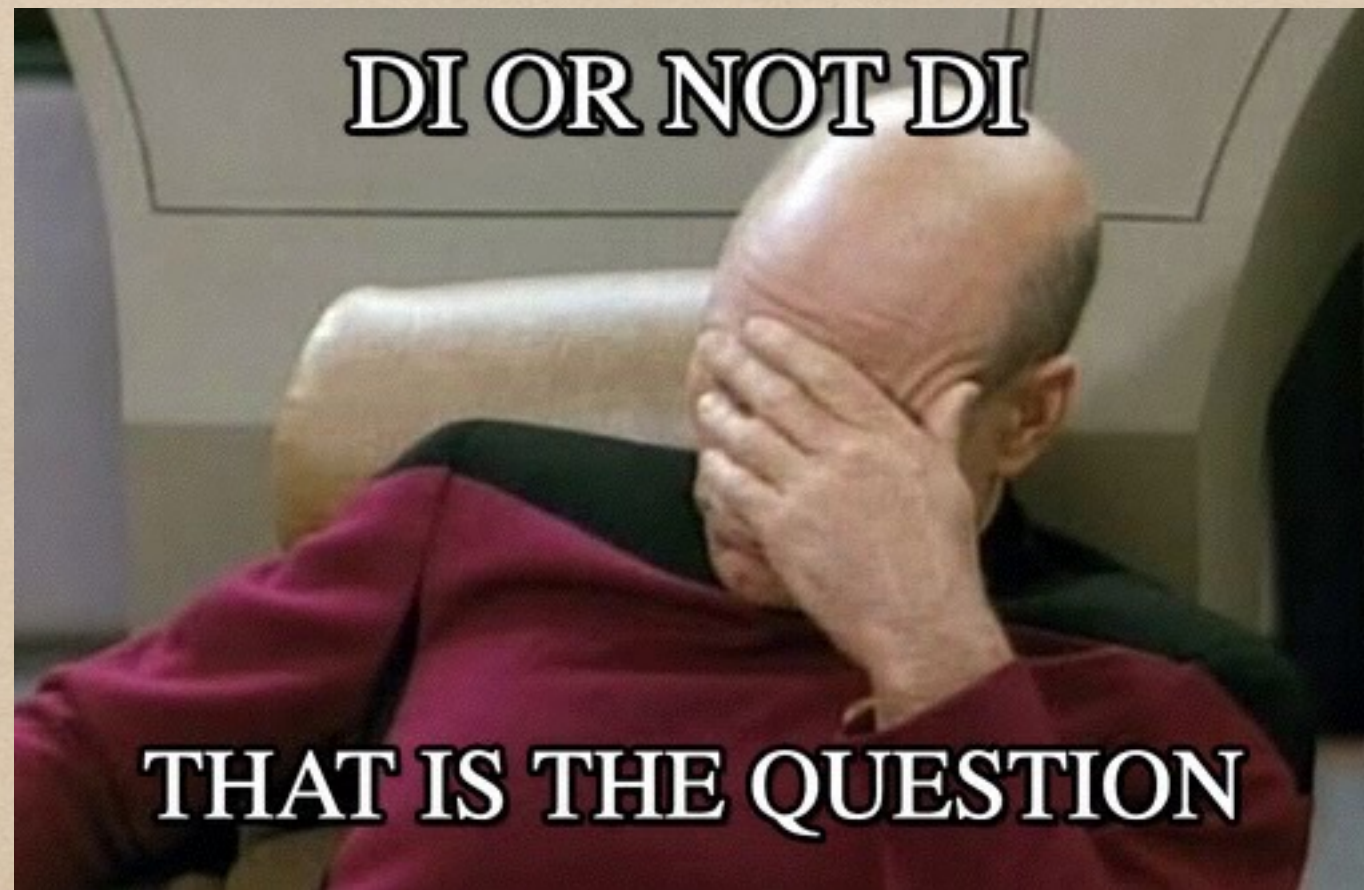


# Play - Non-blocking

- ◆ Built-in WS lib to make non-blocking http calls
- ◆ Async Result in Controller
- ◆ Build on Netty (now on Akka Http)



# Play - Embrace DI



DI: Dependency injection



Dependency chain is toooooo long

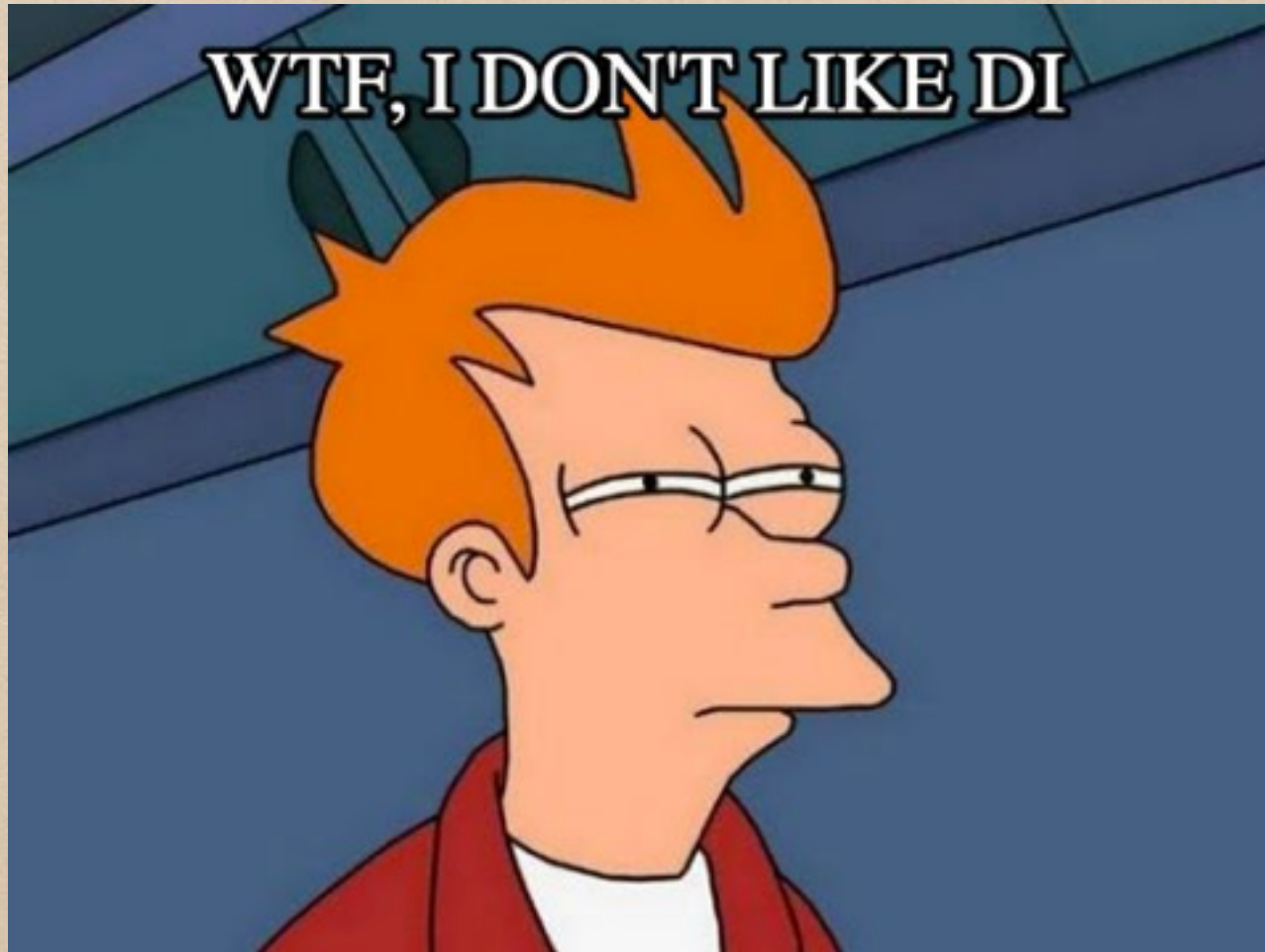
What I want is a simple Object

```
class CacheSystem @Inject() (system: ActorSystem) {  
  def newCache[T](namespace: String): Cache[T] = {  
    // blabla, you needs ActorSystem here  
    ???  
  }  
}  
  
class B1 @Inject() (cache: CacheSystem) {  
  private val b1Cache = cache.newCache("b1")  
}  
  
class B2 @Inject() (cache: CacheSystem) {  
  private val b2Cache = cache.newCache("b2")  
}  
  
class C1 @Inject() (b1: B1, b2: B2)
```

```
object CacheSystem {  
  def newCache[T](namespace: String): Cache[T] = {  
    // blabla, get a global ActorSystem here  
    ???  
  }  
}  
  
class B1 {  
  private val b1Cache = CacheSystem.newCache("b1")  
}  
  
class B2 {  
  private val b2Cache = CacheSystem.newCache("b2")  
}  
  
class C1(b1: B1, b2: B2)
```



WTF, I DON'T LIKE DI





# However, when it comes to tests

- ◆ Objects life cycle management
- ◆ Binding different implementations for the same component



OK, LET'S EMBRACE IT

IN  play





# Test coverage

Scalatest and Scalacheck are your friends

```
lazy val clientGen: Gen[Client] = for {  
  clientId ← Gen.nonEmptyAlphaStr(16)  
  clientSecret ← Gen.nonEmptyAlphaNumStr(32)  
} yield Client(  
  clientId = clientId,  
  clientSecret = clientSecret,  
  redirectUri = RedirectUrl)
```

```
private def WithClientAuthRequest[A](genA: Gen[A])(testBlock: (FakeRequest[AnyContentAsEmpty.type], A) ⇒ Any): Unit = {  
  forAll(clientGen, genA) {  
    case (client, dataA) ⇒  
      val request = clientAuthRequest(client)  
      testBlock(request, dataA)  
  }  
}
```

```
"getByAccessToken" = {  
  "should get token info if accessToken exists" in WithClientAuthRequest(accessTokenGen) {  
    case (request, token) ⇒  
      await(accessTokenDao.insertAndReturn(token))  
      val result = accountController.getByAccessToken(token.accessToken)(request)  
      status(result) shouldBe OK  
      val opt = contentAsString(result).toCaseClass[NormalTokenResponse]  
      opt shouldBe defined  
      opt.get.refreshToken shouldBe token.refreshToken  
  }  
}
```

- ▼ test
  - ▶ cache
  - ▶ controllers
  - ▶ core
  - ▶ dao
  - ▶ forms
  - ▶ helpers
  - ▶ models
  - ▶ resources
  - ▶ services
  - ▶ utils



# Play vs Akka-Http

- ◆ Play: Build a complex app with many 3th party plugins (Authorization/Session/complex page and so on)
- ◆ Akka-Http: Rest API



# Akka

“Akka is a toolkit and runtime  
for building highly concurrent,  
distributed, and resilient  
message-driven applications  
on the JVM.”

–akka.io



# Akka Components

- ◆ Actor
- ◆ IO
- ◆ Cluster
- ◆ Cluster Tools (PubSub, Sharding, ...)
- ◆ Persistence
- ◆ Streams
- ◆ Http
- ◆ Typed



“Akka is too much, let’s just talk about actor”



- ◆ Everything is actor
- ◆ One actor is no actor
- ◆ Structure your actor
- ◆ Blocking needs careful management
- ◆ Let it crash



# Everything is actor

- ◆ Actors are cheap
- ◆ Actors model real-world entity
- ◆ user, dog, neuron, ATM, bank, etc

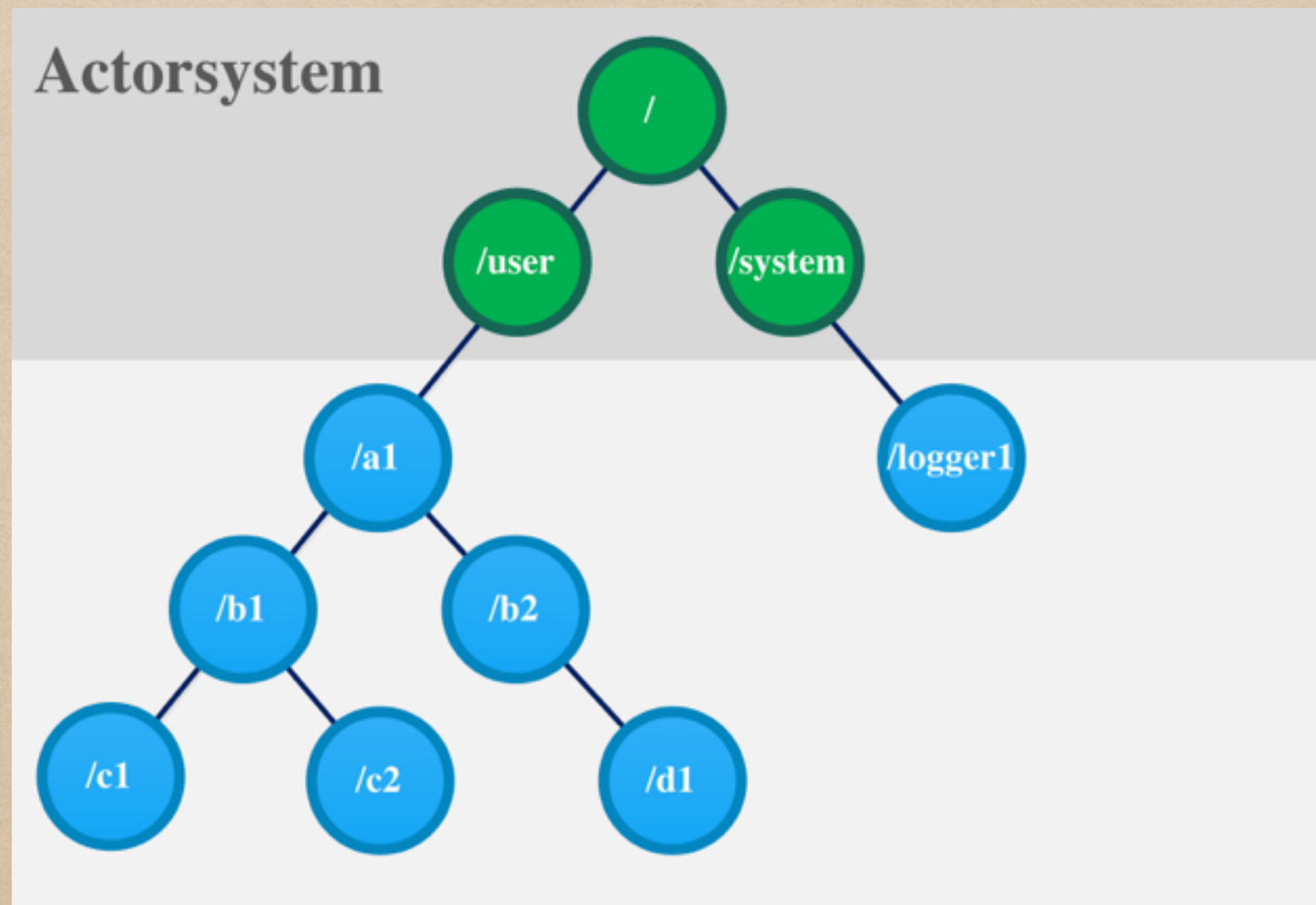


# One actor is no actor

- ◆ Actors are meant to work together
- ◆ An actor should do one thing and do it well
- ◆ Philosophy: delegate, delegate and delegate



# Structure your actor



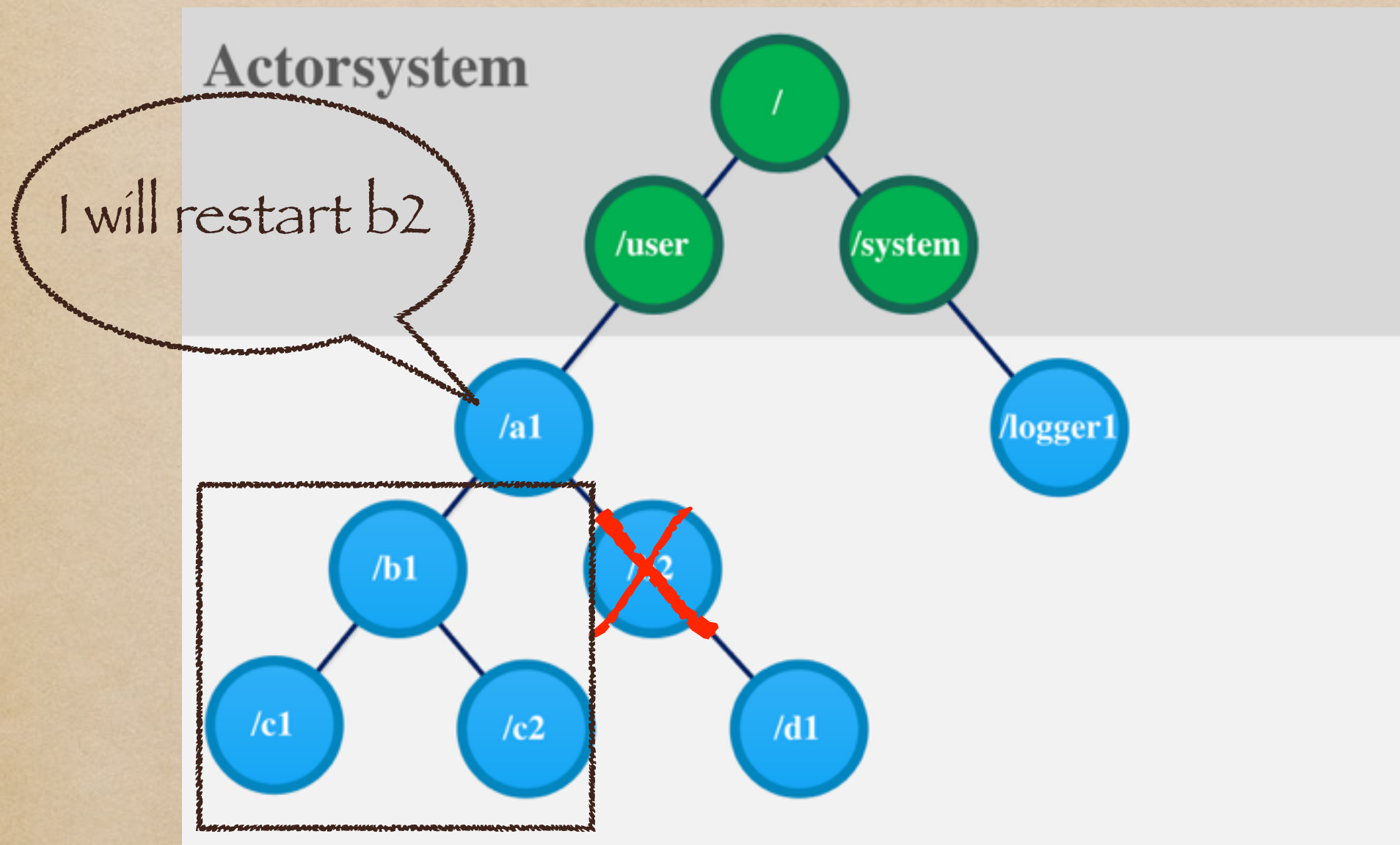


# Blocking needs careful management

- ◆ Don't block in actor
- ◆ Use dedicated Execution Context



# Let it crash

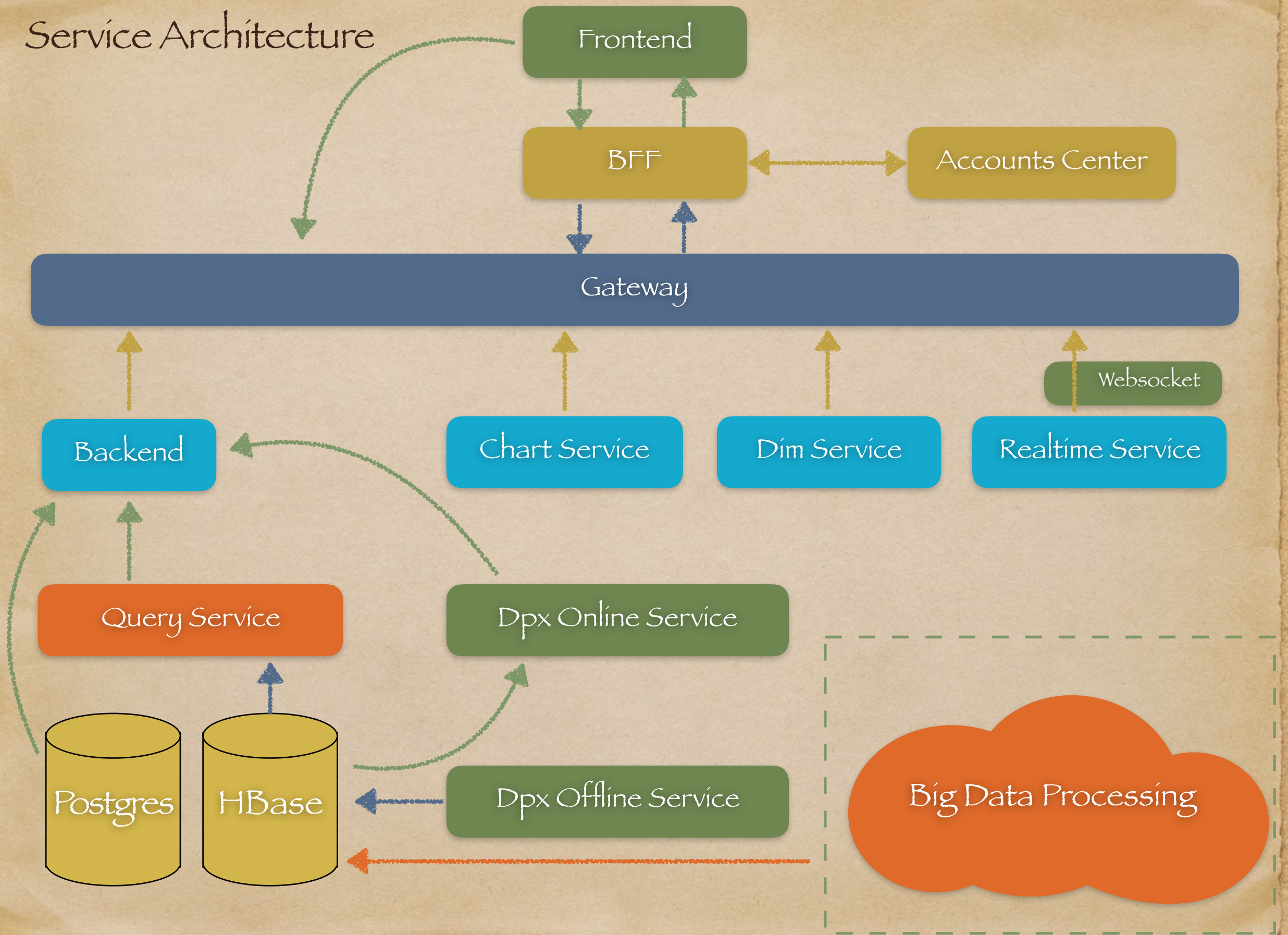




“Look at Service Architecture again”



# Service Architecture

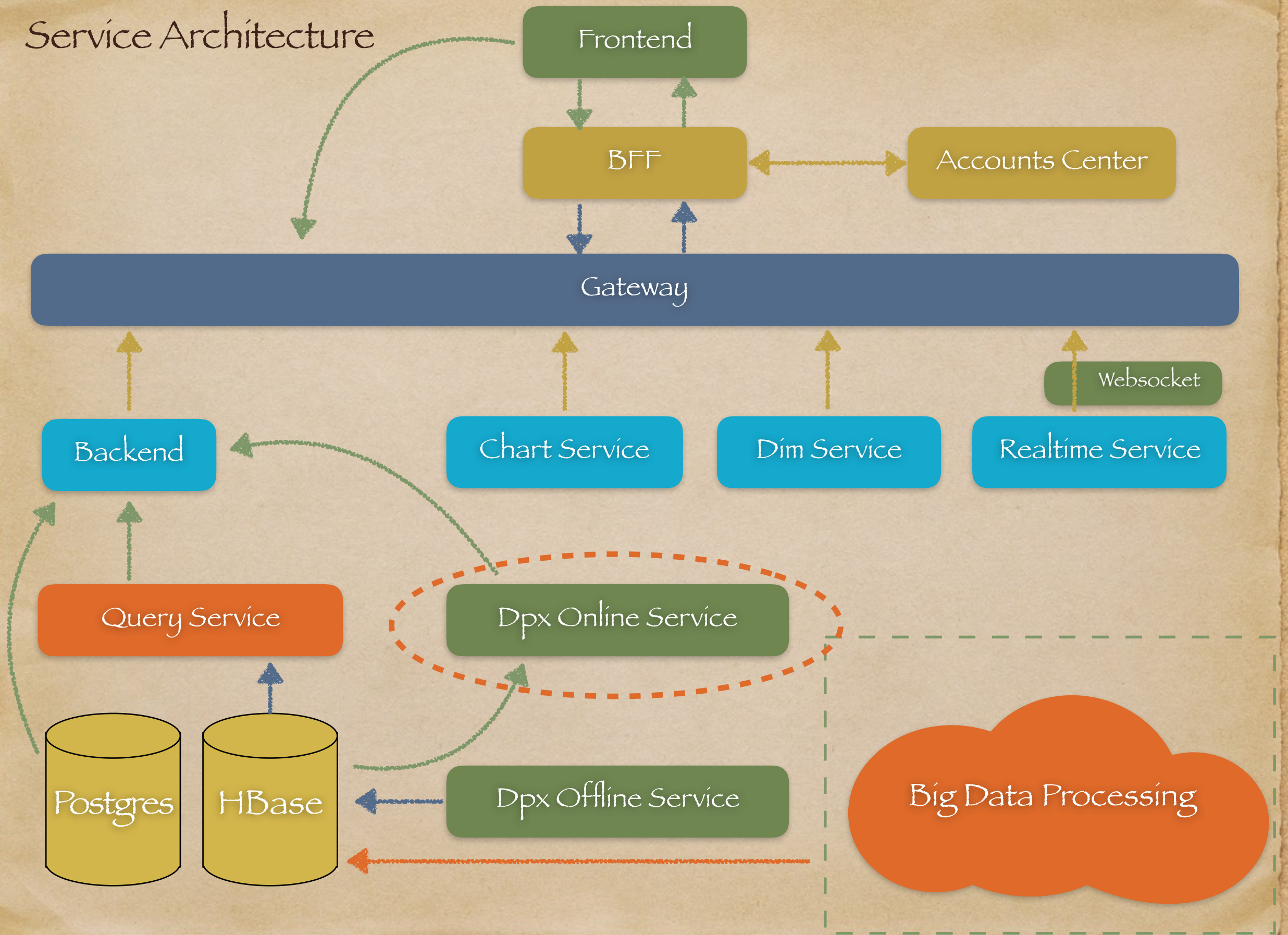




“That’s too much, let’s just talk about one service”



# Service Architecture





domain, page(path), xpath



Dpx Online Service



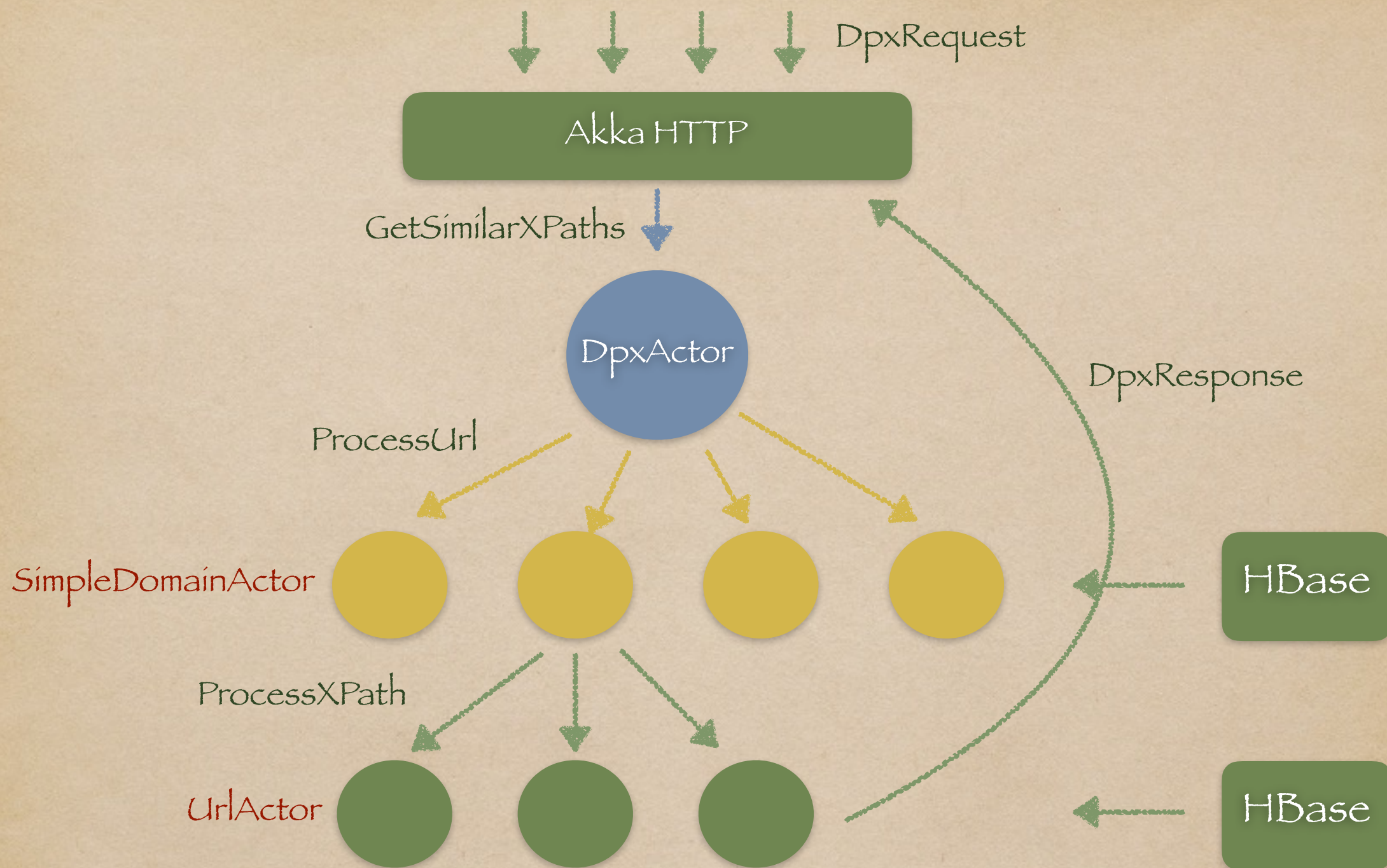
recommanded xpaths





HERE ARE SOME MAGIC ALGORITHMS







Use domain model and serialize it

```
sealed trait DpxRequest

case class GetSimilarXPaths(
  domain: String,
  page: String,
  xpath: String,
  levelLimit: Int,
  valueLimit: Int) extends DpxRequest
```

```
sealed trait DpxResponse

case class SimilarXPaths(xpath: List[String]) extends DpxResponse
case class ErrorResponse(error: String) extends DpxResponse
```



# Serialization: Json4s Support

```
trait Json4sSupport {  
  import Json4sSupport._  
  
  implicit def json4sUnmarshaller[A: Manifest](implicit serialization: Serialization, formats: Formats): FromEntityUnmarshaller[A] =  
    Unmarshaller  
      .byteStringUnmarshaller  
      .forContentTypes(`application/json`)  
      .mapWithCharset { (data, charset) =>  
        try serialization.read(data.decodeString(charset.nioCharset.name))  
        catch {  
          case MappingException("unknown error", ite: InvocationTargetException) => throw ite.getCause  
        }  
      }  
  
  implicit def json4sMarshaller[A <: AnyRef](implicit serialization: Serialization, formats: Formats, shouldWritePretty: ShouldWritePretty = ShouldWritePretty.False): ToEntityMarshaller[A] =  
    shouldWritePretty match {  
      case ShouldWritePretty.False => Marshaller.StringMarshaller.wrap(`application/json`)(serialization.write[A])  
      case _ => Marshaller.StringMarshaller.wrap(`application/json`)(serialization.writePretty[A])  
    }  
}
```

```
trait BaseRoute extends Json4sSupport with LazyLogging {  
  
  implicit val serialization: Serialization = jackson.Serialization  
  implicit val formats = DefaultFormats  
}
```



Thank you