Newman

A Functional REST Client for Scala

Aaron Schlesinger
Sr. Member of the Technical Staff, PayPal

PayPal™

Background

- Internal HTTP Services
- External HTTP Services
- Many HTTP Clients

PayPal™

Existing Wheels

PayPal™

Apache HttpClient

- Synchronous
- Thread per request
- Apache Backed

PayPal™

Apache HttpClient Setup

```
val connManager: ClientConnectionManager = {
  val cm = new PoolingClientConnectionManager()
  cm.setDefaultMaxPerRoute(maxConnectionsPerRoute)
  cm.setMaxTotal(maxTotalConnections)
  cm
}
val httpClient: AbstractHttpClient = {
  val client = new DefaultHttpClient(connManager)
  val httpParams = client.getParams
  HttpConnectionParams.setConnectionTimeout(httpParams,
connectionTimeout)
  HttpConnectionParams.setSoTimeout(httpParams,
socketTimeout)
  client
}
```

PayPal™

Apache HttpClient Execution

```
val req = new HttpGet
val url = new URL("http://paypal.com”)
req.setURI(url.toURI)
val headers: List[(String, String)] = ???
headers.foreach { tup: (String, String) =>
  val (headerName, headerValue) = tup
  if(!headerName.equalsIgnoreCase("Content-Type")) {
    req.addHeader(headerName, headerValue)
  }
}
val body: Array[Byte] = Array(‘a’.toByte, ‘b’.toByte,
‘c’.toByte)
req.setEntity(new ByteArrayEntity(body))
val resp = httpClient.execute(req)
```

PayPal™

Finagle

- Netty event loop(s)
- Twitter's core
- Big Community

PayPal™

Finagle Code

```
val host = "http://www.siliconvalley-codecamp.com/"
val url = new URL(host)
val client = ClientBuilder()
  .codec(Http())
  .hosts(host) //there are more params you can set here
  .build()
val headers: Map[String, String] = ???
val method: Method = new HttpGet()
//this is an org.jboss.netty.buffer.ChannelBuffer
val channelBuf: ChannelBuffer = ???
val req = RequestBuilder()
  .url(url)
  .addHeaders(headers)
  .build(method, Some(channelBuf))
val respFuture: Future[HttpResponse] = client.apply(req)

respFuture.ensure {
  client.close() //don't forget!
}
```

PayPal™

The Result

- Battle tested Core(s)
- Must Write Setup, Cleanup Code
- 4-8 LOC per request
- Non-Idiomatic
- Lots to Remember
- Must Commit to One Implementation

HTTP Clients are Solved.

PayPal™

Make Them Consistent.

PayPal™

Why Newman

- Just hold Newman in our Head
- Make it Safe and Fast
- Build Higher level features

PayPal™

The Standard Interface

- Fewer LOC, fewer bugs
- Standard Practices
- Common Patterns

PayPal™

Safety

PayPal™

Referential Transparency

- Replace a `def` with a `val`
- No exceptions, I/O, random numbers
- Use for “isolation”

PayPal™

Type Safety

- Rich Type System
- Use Wisely
- Happy/Furious Spectrum

PayPal™

Building URLs

PayPal™

Demo

UrlBuilderExample.scala

PayPal™

Immutability

- Copy, don’t change
- Follow data through code
- Cache forever. It won’t change

PayPal™

Performance

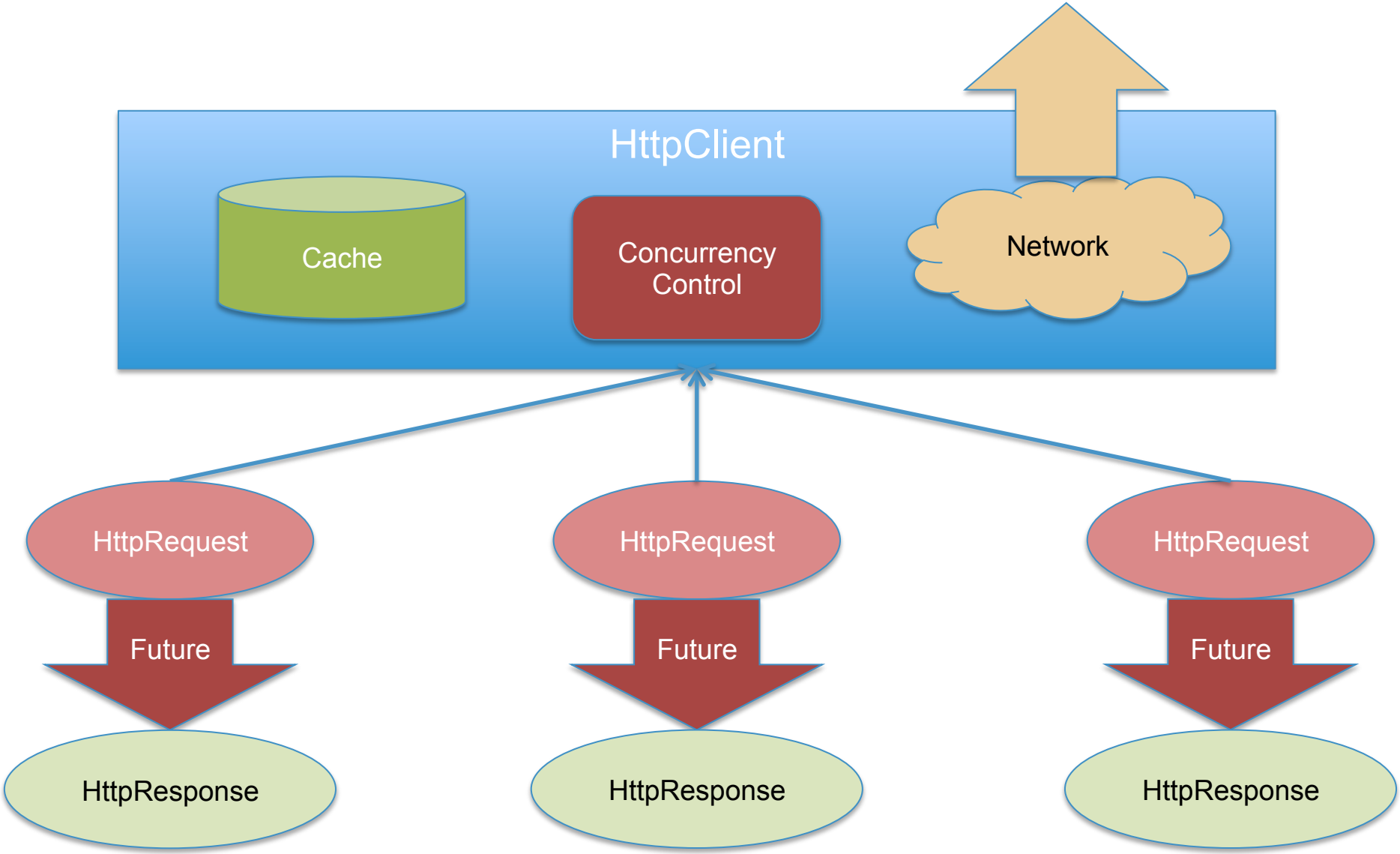
PayPal™

Make the common case fast and the hard case possible.

PayPal™

Demo
CommonRequestPattern.scala

PayPal™


HttpClient
Cache
Concurrency Control
Network
HttpRequest
HttpRequest
HttpRequest
Future
Future
Future
HttpResponse
HttpResponse
HttpResponse

PayPal™

Building on Top of Core Newman

PayPal™

DSLs

- Non-Evil
- Canonical usage
- Easy to read, write, review, teach

PayPal™

Demo

RequestBuilding.scala

PayPal™

Demo

ResponseHandling.scala

PayPal™

Client Diversity

- Plug and Play
- Performance tradeoffs
- Match with your environment

PayPal™

Creating New Clients

- Subclass HttpClient
- Add a new HTTP Verb
- Maybe Add a Cache, Serializer

PayPal™

Demo

StubClient.scala

PayPal™

Futures

- Separate response handling from request
- Part of Scala Standard Lib
- Lots of prior art

PayPal™

Time Boxing

PayPal™

Demo

TimeBoxing.scala

PayPal™

First One Wins

PayPal™

Demo

FirstOneWins.scala

PayPal™

Fan Out

PayPal™

Demo

FanOut.scala

PayPal™

Caching

- Plug and Play
- Strategies
- Backends

PayPal™

JSON

- Encoding In HttpRequest
- Decoding in HttpResponse
- Cache saves CPU here

PayPal™

More

- Unified Encoding/Decoding
- Unified configuration
- Instrumentation
- Client stubs
- https://github.com/stackmob/newman/issues

PayPal™

Aaron Schlesinger

https://github.com/arschles/newman-example

http://github.com/stackmob/newman

http://github.com/arschles

PayPal™

Extra: Best Tool For The Job

- Thread-per-request
- Event loop
- Actor based

PayPal™

Performance Overhead

- Immutability = copying stuff
- Copies are small and “young”
- Small & Short Lived Memory = Good GC Throughput

PayPal™