TDC2012

the developer's conference

Trilha – JavaEE

Rest in Java 2.0

Eder Ignatowicz

Eder Ignatowicz...

@ederign

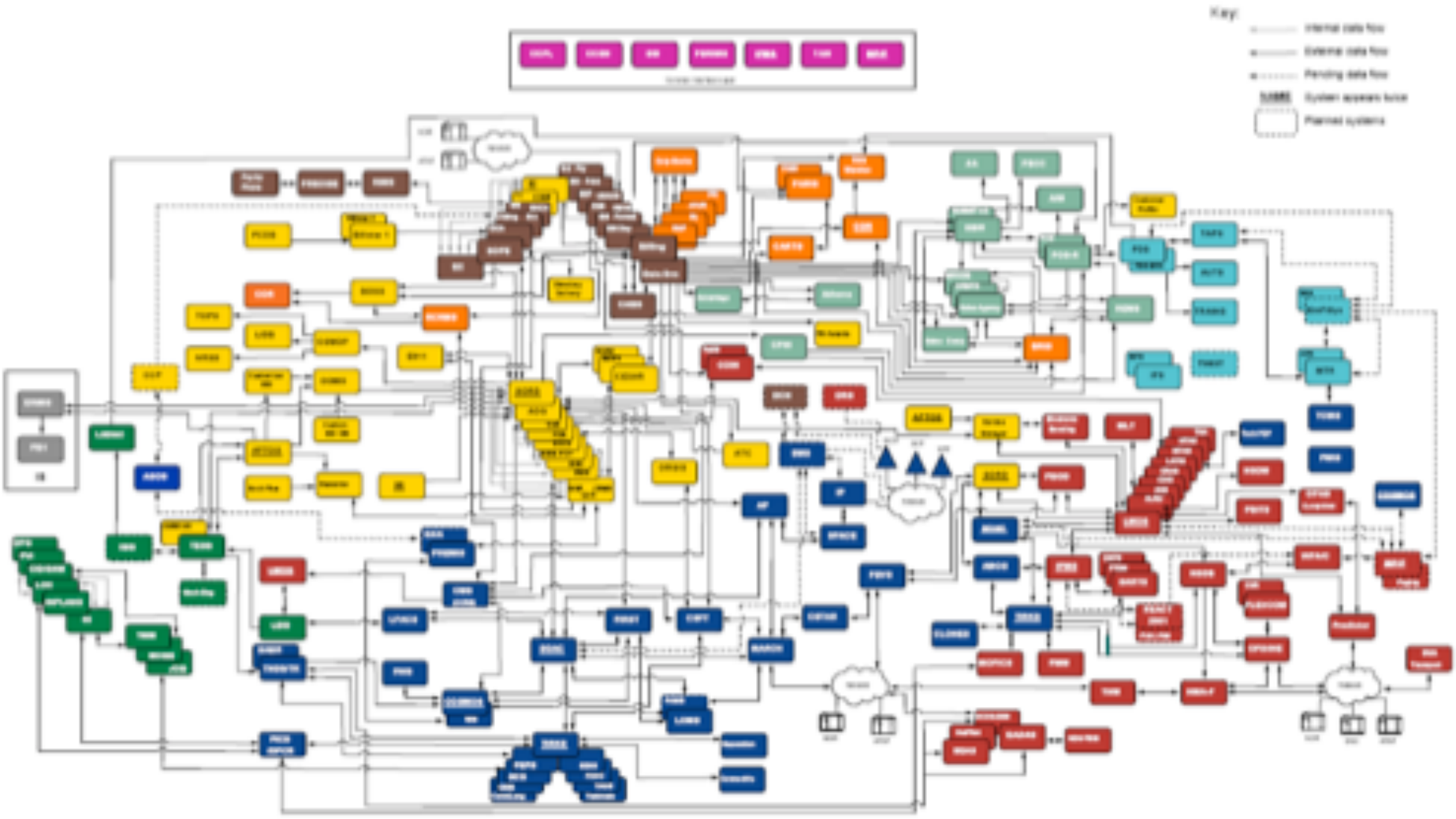
Generalista
(Arquitetura, NoSQL, Devops, QA)

Doutorando na Unicamp
(Polyglot Persistence em Cidades Digitais)

Professor na Faccamp e Unisal

Editor líder no InfoQ Brasil

O mundo, antes de REST


Key:
Internal data flow
External data flow
Pending data flow
System appears twice
Planned systems

Tempos difíceis...

Muitos “padrões”

RMI, Corba, DCE, DCOM

Muitos fornecedores

Sun, Microsoft, IBM, OASIS, OMG

Muitas lágrimas

Não existia interoperabilidade
Reinvenção da roda
Vendor “lock-in”

Web Services SOAP
A promessa

Web Services SOAP

Solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes.

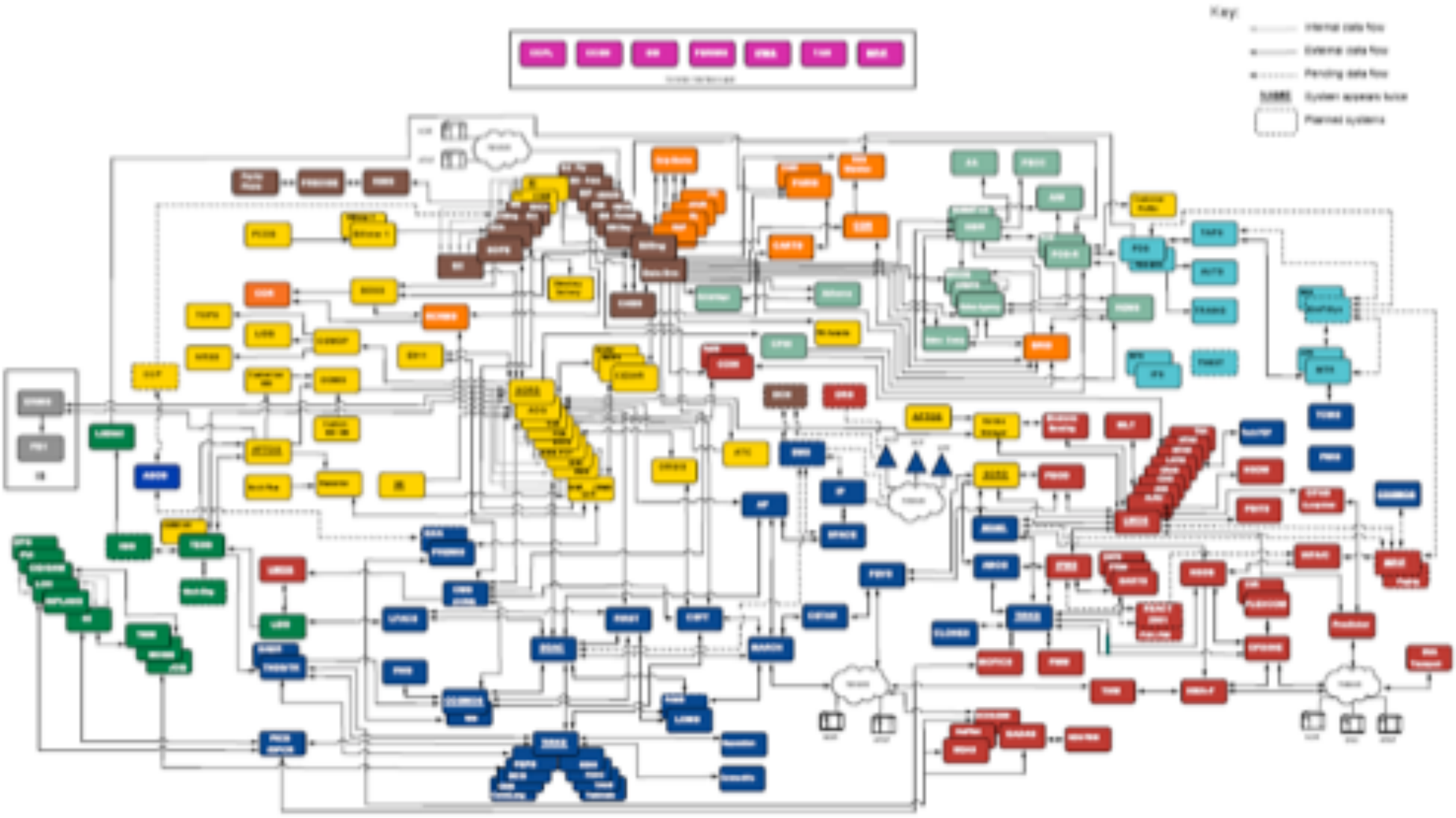
Padrões Abertos

Independência

Sistema Operacional

Linguagem de Programação

Como SOAP é...


Key:
Internal data flow
External data flow
Persisting data flow
System appears twice
Planned systems


Então surgiu o
REST!!!

Representational **S**tate **T**ransfer (**REST**) é um estilo de arquitetura de software para sistemas distribuídos hypermedia semelhantes a World Wide Web”

Roy Thomas Fielding

Características REST

Cliente-servidor

Stateless

Cacheable

Interface Uniforme

Baseado em camadas

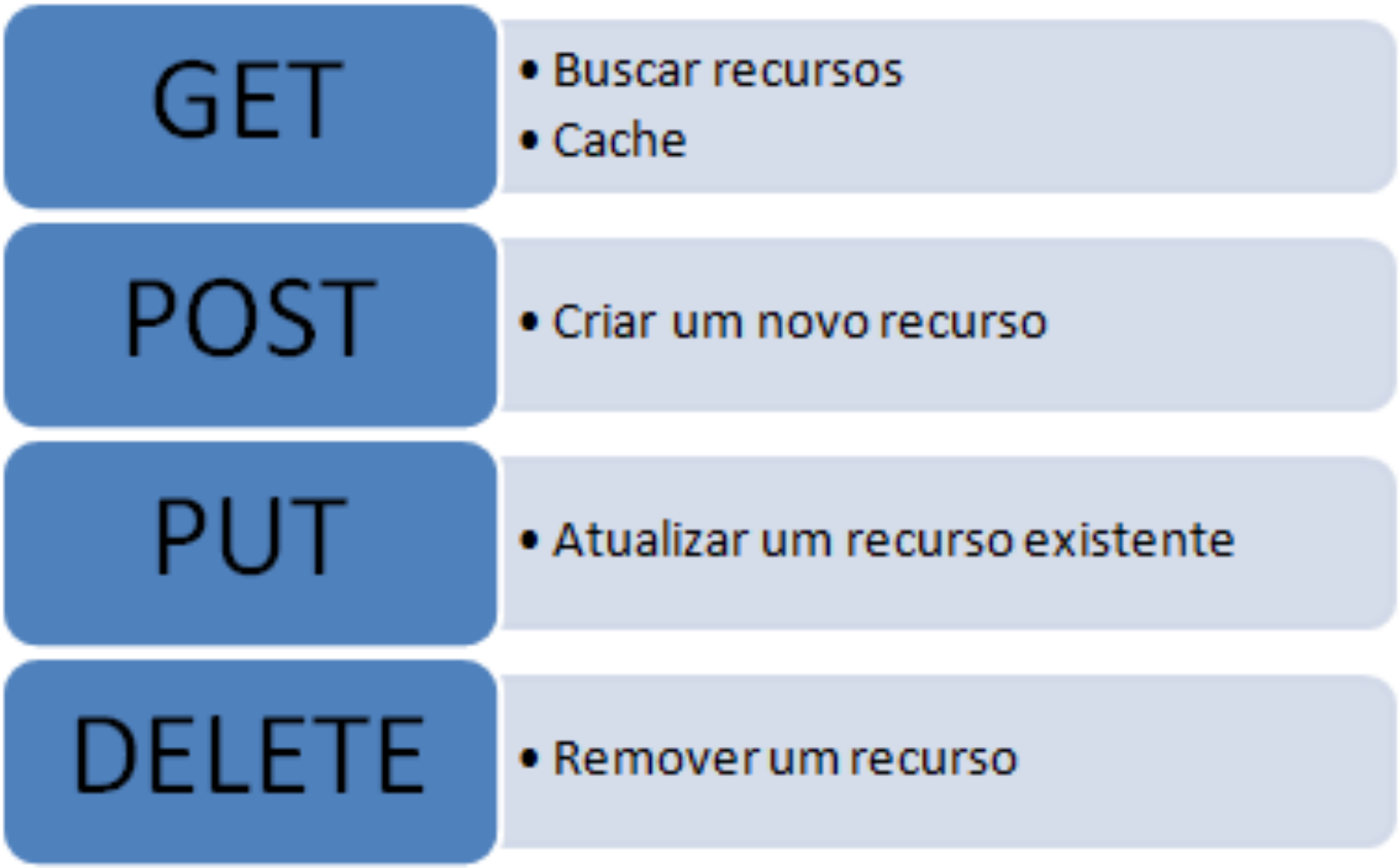
Princípios REST

Identificação de recursos

Manipulação destes recursos através de representações

Mensagens auto-descritivas

Hypermedia como engine do estado da aplicação


GET
• Buscar recursos
• Cache
POST
• Criar um novo recurso
PUT
• Atualizar um recurso existente
DELETE
• Remover um recurso

O que sempre sonhamos na integração de sistemas...

Mas extremamente mal compreendida

O que precisa ser feito para que entendam que no estilo **arquitetural REST** o ***hypertext*** é um **pré-requisito**? Em outras palavras, se a ***engine*** do estado da aplicação (e consequentemente sua API) não é guiada por ***hypertext***, então sua aplicação não pode ser **RESTful** e nem ter uma **API REST**. PONTO . Existe por ai algum manual que necessite ser

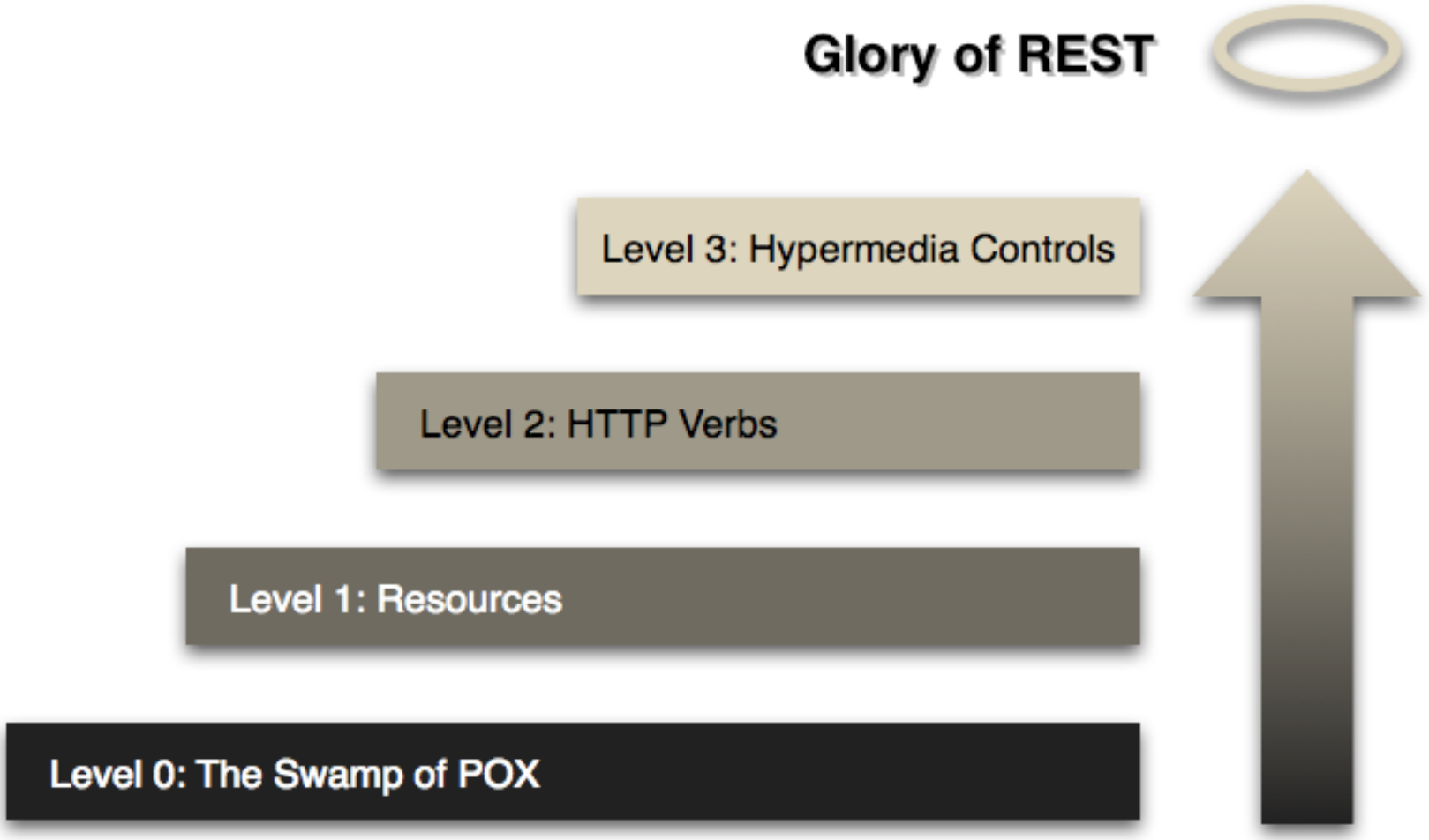
Roy Thomas Fielding

Primeira linha do capítulo 5 da tese do Roy...

"REST [is an] architectural style for distributed **hypermedia** systems"

Roy Thomas Fielding

Richardson's Maturity Model



Level 0: The Swamp of Pox

Uma **URI**, um método **HTTP**
XML-RPC / **SOAP** / POX
HTTP usado como **transporte**

```
POST /appointmentService HTTP/1.1
[various other headers]

<appointmentRequest>
  <slot doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointmentRequest>
```

Level 1: Resources

Cada recurso tem uma **única URI**
URI tunneling
Um único verbo HTTP (POST ou GET)
HTTP usado como **transporte**

```
POST /slots/1234 HTTP/1.1
[various other headers]

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>
```

Level 2: HTTP Verbs

Muitas URIs, utilizando corretamente os verbos HTTP

Uso correto dos códigos de resposta

Expõe estado e não comportamento

CRUD

```
GET /doctors/mjones/slots?date=20100104&status=open HTTP/1.1
Host: royalhope.nhs.uk

HTTP/1.1 200 OK
<openSlotList>
  <slot id = "1234" start = "1400" end = "1450"/>
  <slot id = "5678" start = "1600" end = "1650"/>
</openSlotList>
```

Roy, os níveis 0, 1 e 2 são RESTful?

NÃO!

Roy Thomas Fielding

Level 3: Hypermedia controls

Recursos auto descritivos

Hypermedia **A**s **T**he **E**ngine **o**f **A**pplication **S**tate (**HATEOAS**)

Clientes só precisam saber a URI root (home page) de uma API e os media types utilizados

O resto é HTTP e links

```
<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400"
        end = "1450"/>
  <patient id = "jsmith"/>
  <link rel = "/linkrels/appointment/addTest"
        uri = "/slots/1234/appointment/tests"/>
  <link rel = "/linkrels/appointment/updateContactInfo"
        uri = "/patients/jsmith/contactInfo"/>
</appointment>
```

HATEOAS

Hypermedia / Mime-types / Media-types
Descrevem o estado atual da aplicação
Analogia da página web

Links
Descrevem como navegar ao próximo estado
Analogia aos links tradicionais

HATEOAS é o que nos faz navegar na web

“Em cada mensagem de resposta, inclua os links para a próxima mensagem”

HATEOAS

Descreva contratos com links

Links das páginas são contratos de navegação
Links nos levam a outros recursos que também possuem links
*O mesmo se aplica aos nossos sistemas: descrevendo **protocolos***
Use links como uma máquina de transição de estados
Formatos Hypermedia (ATOM e XHTML)

HATEOAS

```
<feed xmlns="http://www.w3.org/2005/Atom">
        <title>Example Feed</title>
        <subtitle>A subtitle.</subtitle>
        <link href="http://example.org/feed/"
              rel="self" />
        <link href="http://example.org/" />
        <id>urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6</id>
        <updated>2003-12-13T18:30:02Z</updated>
        <author>
                <name>John Doe</name>
                <email>johndoe@example.com</email>
        </author>
        <entry>
                <title>Atom-Powered Robots Run Amok</title>
                <link href="http://example.org/2003/12/13/atom03" />
                <link rel="alternate" type="text/html"
                     href="http://example.org/2003/12/13/atom03.html"/>
```

HATEOAS

Web é o nosso framework

A web nos dá um modelo de processamento e metadados
Verbos e códigos de status
Headers

Nos dá contratos ou Web “APIs”
URI
Links

Workflows de uma maneira “web-friendly”

Isto é REST !

Roy Thomas Fielding

Eder, esta é a trilha de JavaEE.

JAX-RS API

JAX-RS 1.0 é a API Java para RESTful WS

POJO-Based

(sem complicações ou contratos, só annotations)

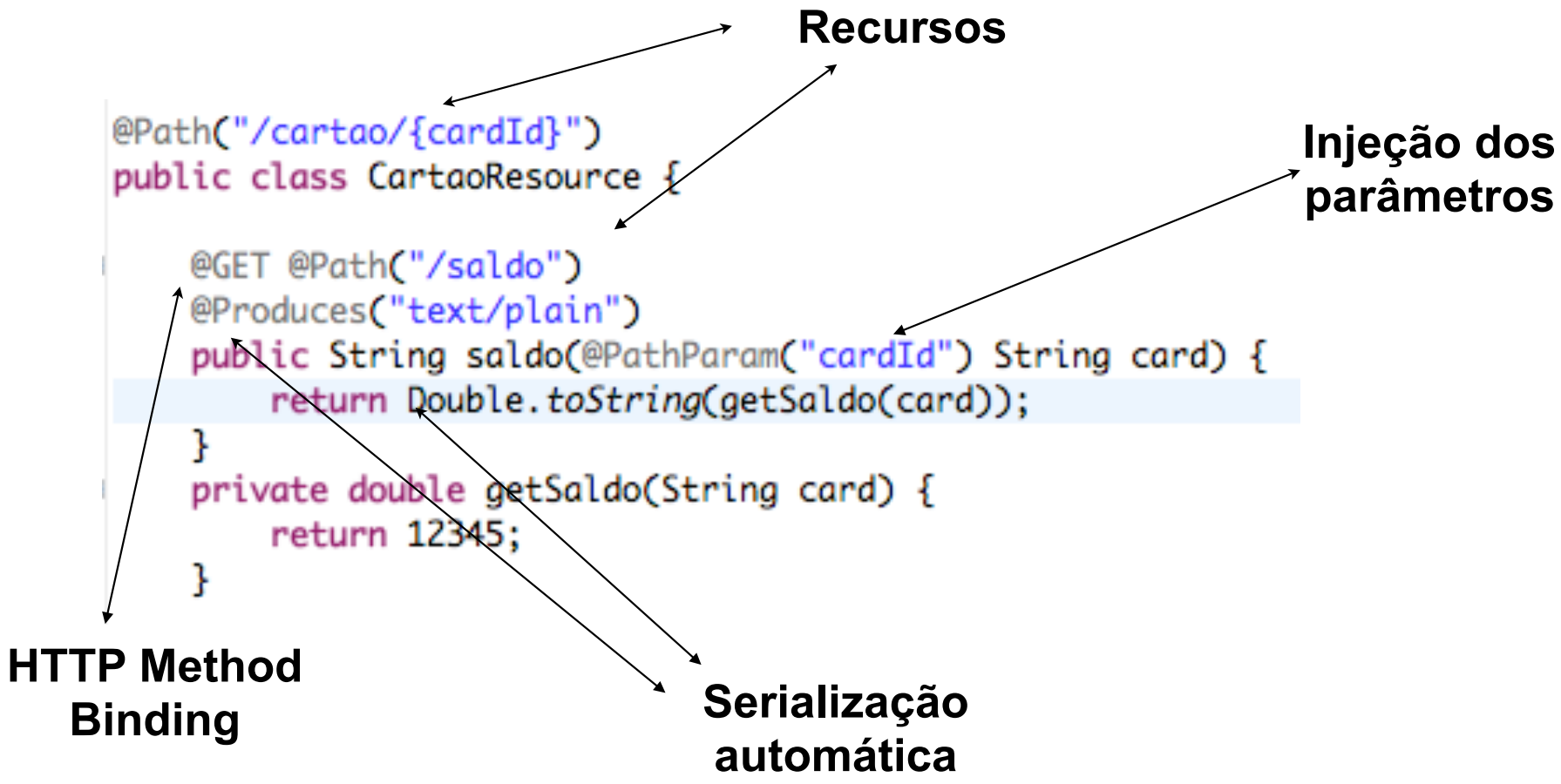
HTTP-Based

JAXB Based

Independente de container, formato

Parte do JavaEE

JAX-RS API


Recursos
@Path("/cartao/{cardId}")
public class CartaoResource {
@GET @Path("/saldo")
@Produces("text/plain")
public String saldo(@PathParam("cardId") String card) {
return Double.toString(getSaldo(card));
}
private double getSaldo(String card) {
return 12345;
}
Injeção dos parâmetros
HTTP Method Binding
Serialização automática

Implementações JAX-RS

Apache CXF

wink

Triaxrs

Jersey

Restlet

RESTEasy

Restfulie

Restful made easy.

JSR 339: JAX-RS 2.0

Early Draft Review 3 07/06/2012

Adopt a JSR

JSR 339: JAX-RS 2.0

Como experimentar?

Jersey

+

Grizzly

```
mvn archetype:generate -DarchetypeArtifactId=jersey-
quickstart-grizzly2
-DarchetypeGroupId=org.glassfish.jersey.archetypes
-DinteractiveMode=false -DgroupId=com.example
-DartifactId=simple-service -Dpackage=com.example
-DarchetypeVersion=2.0-m04
```

Client API

```
@Before
public void setUp() throws Exception {
    // start the server
    server = Main.startServer();
    // create the client
    Client c = ClientFactory.newClient();
    target = c.target(Main.BASE_URI);
}

@After
public void tearDown() throws Exception {
    server.stop();
}

@Test
public void testExtrato() {
    String responseMsg = target.path("cartao/1/saldo").request()
            .get(String.class);
    assertEquals(CartaoResource.CONSTANT_DEMO, responseMsg);
}
```

Interceptors/Handlers

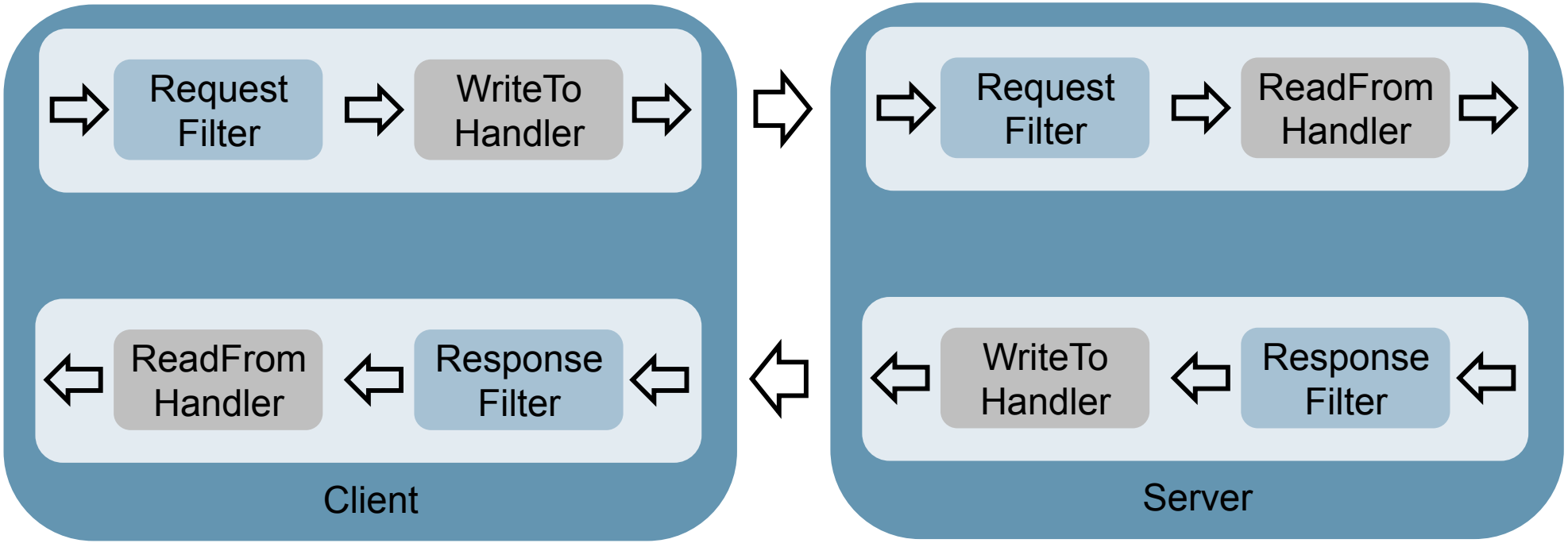
Pontos de extensão: Logging, Compression, Security, etc.

```
@Provider
class LoggingFilter
    implements RequestFilter, ResponseFilter {

    @Override
    public FilterAction preFilter(FilterContext ctx)
      throws IOException {
        logRequest(ctx.getRequest());
        return FilterAction.NEXT;
    }
    @Override
    public FilterAction postFilter(FilterContext ctx)
      throws IOException {
        logResponse(ctx.getResponse());
        return FilterAction.NEXT;
    }
```

Interceptors/Handlers

```
@Provider
public class GzipInterceptor implements ReaderInterceptor,
WriterInterceptor {

    @Override
    public Object aroundreadFrom(ReadInterceptorContext ctx)
        throws IOException {
        if (gzipEncoded(ctx)) {
            InputStream old = ctx.getInputStream();
            ctx.setInputStream(new GZIPInputStream(old));
        }
        return ctx.proceed();
    }
}
```

Interceptors/Handlers



Async

Suspende e resume conexões
Suporte async doServlet 3.x
Suporte na API Client

```
@Path("/async/long")
public class AsyncResource {
    @Context
    private ExecutionContext ctx;

    @GET
    @Produces("text/plain")
    public void longRunningOp() {
        Executors.newSingleThreadExecutor().submit(new Runnable() {
            public void run() {
                try {
                    Thread.sleep(10000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                ctx.resume("Hello async world!");
            }
        });
        ctx.suspend(); // Suspend connection and return
    }
}
```

Async

Exemplo CLIENT API Async

```
// Acesso URI
Target target = client.target("http://.../atm/balance")…

// Chamada async e callback
Future<?> handle = target.request().async().get(
    new InvocationCallback<String>() {
        public void complete(String balance) { … }
        public void failed(InvocationException e) { … }
      });
```

Hypermedia

Suporte a HATEOAS

```
// Server API
Response res = Response.ok(order)
    .link("http://.../orders/1/ship", "ship")
    .build();

// Client API
Response order = client.target(…)
    .request("application/xml").get();

if (order.getLink(“ship”) != null) {
    Response shippedOrder = client
        .target(order.getLink("ship"))
        .request("application/xml").post(null);
 }
```

Melhora na negociação de conexão

```
GET http://.../resource
Accept: text/*; q=1
…

Path("resource")
public class Resource {
  @GET
  @Produces("text/plain;qs=0.5",
            "text/html;qs=0.75")
  public String getResource() {...}
}
```

Mais informações?

JSR: http://jcp.org/en/jsr/detail?id=339
Java.net: http://java.net/projects/jax-rs-spec
User Alias: users@jax-rs-spec.java.net
Adopt a JSR: https://blogs.oracle.com/java/entry/adopt_a_jsr

Dúvidas?

@ederign

Bibliografia

https://blogs.oracle.com/arungupta/entry/jax_rs_2_0_early

REST: From GET to HATEOAS - Jos Dirksen

Architectural Styles and
the Design of Network-based Software Architectures: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

JSR 339: http://jcp.org/en/jsr/detail?id=339