REST in Java 2.0

Eder Ignatowicz

Eder Ignatowicz...

@ederign

Generalista

(Arquitetura, NoSQL, Devops, QA)

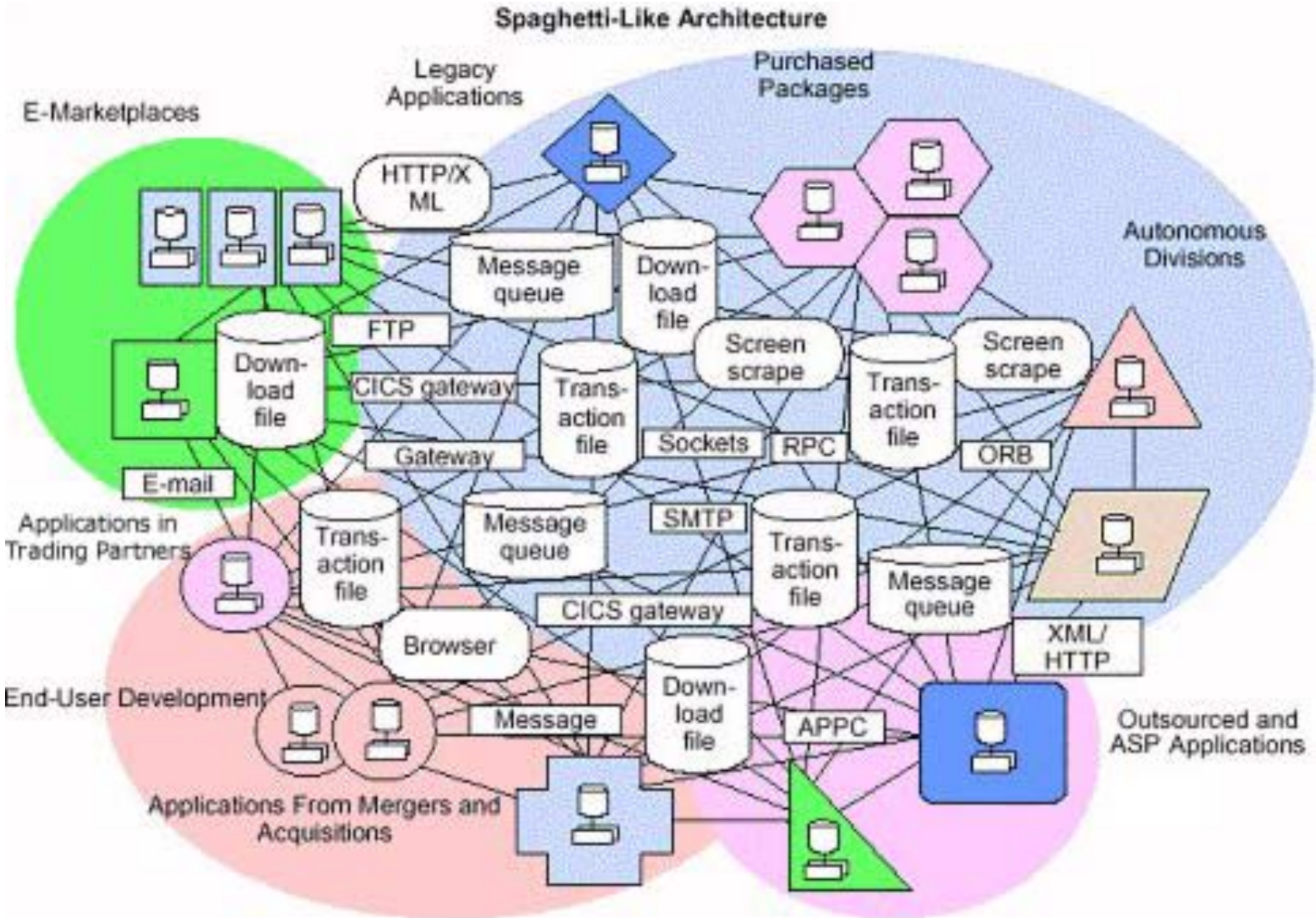
Doutorando na Unicamp

(RESTful e Polyglot Persistence aplicado em Cidades Digitais)

Professor na Faccamp e Unisal

Editor líder no InfoQ Brasil


O mundo, antes de REST



http://geekswithblogs.net/images/geekswithblogs_net/ugandadotnet/eai-spaghetti.jpg

Tempos difíceis...

Muitos "padrões"

RMI, Corba, DCOM

Muitos fornecedores

Sun, Microsoft, IBM, OASIS, OMG

Muitas lágrimas

Não existia interoperabilidade

Reinvenção da roda

Vendor "lock-in"

http://thetowerofbabel.net/yahoo_site_admin/assets/images/tower_of_babel.170113154.jpg

Web

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

Padrões Abertos

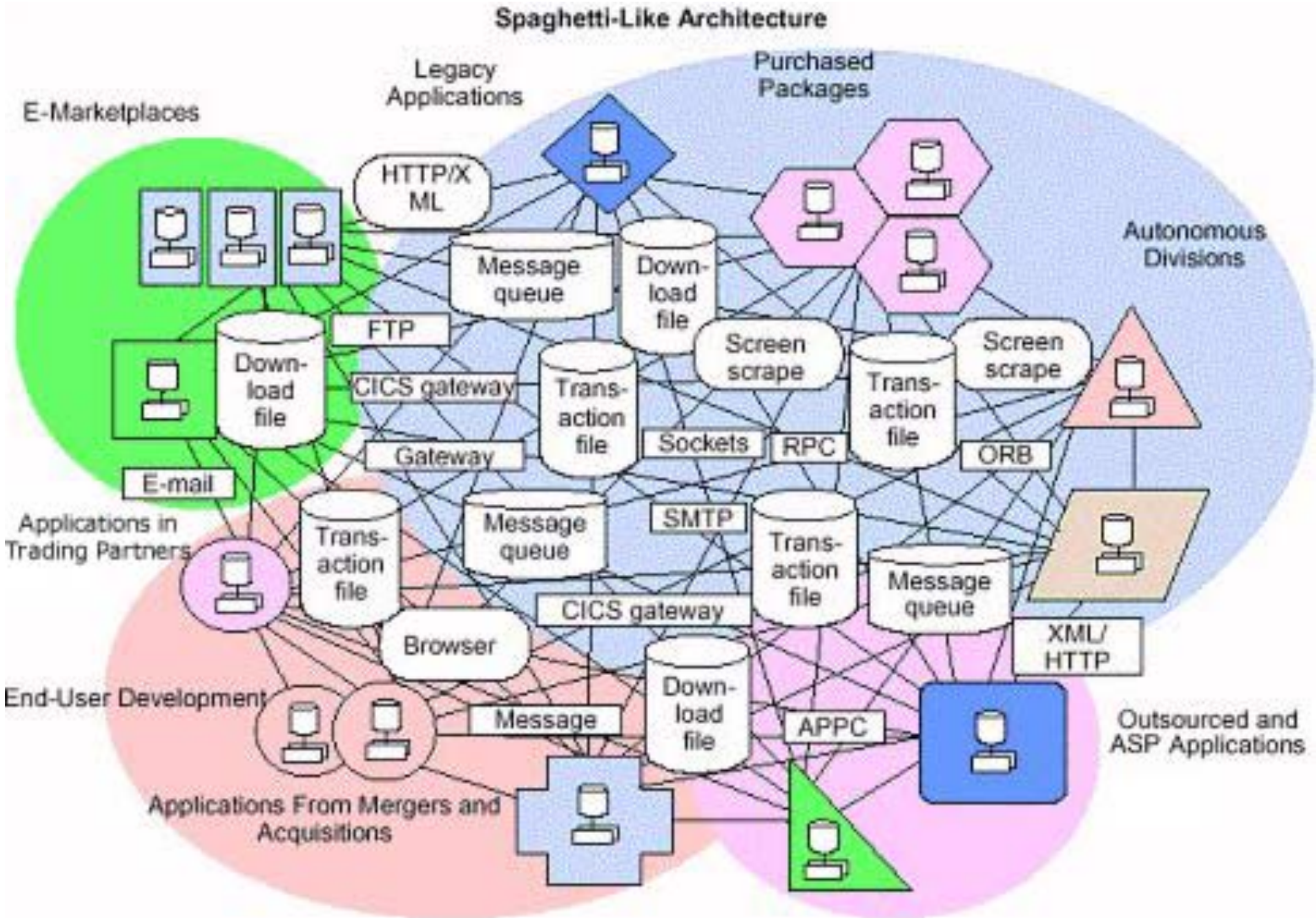
Independência

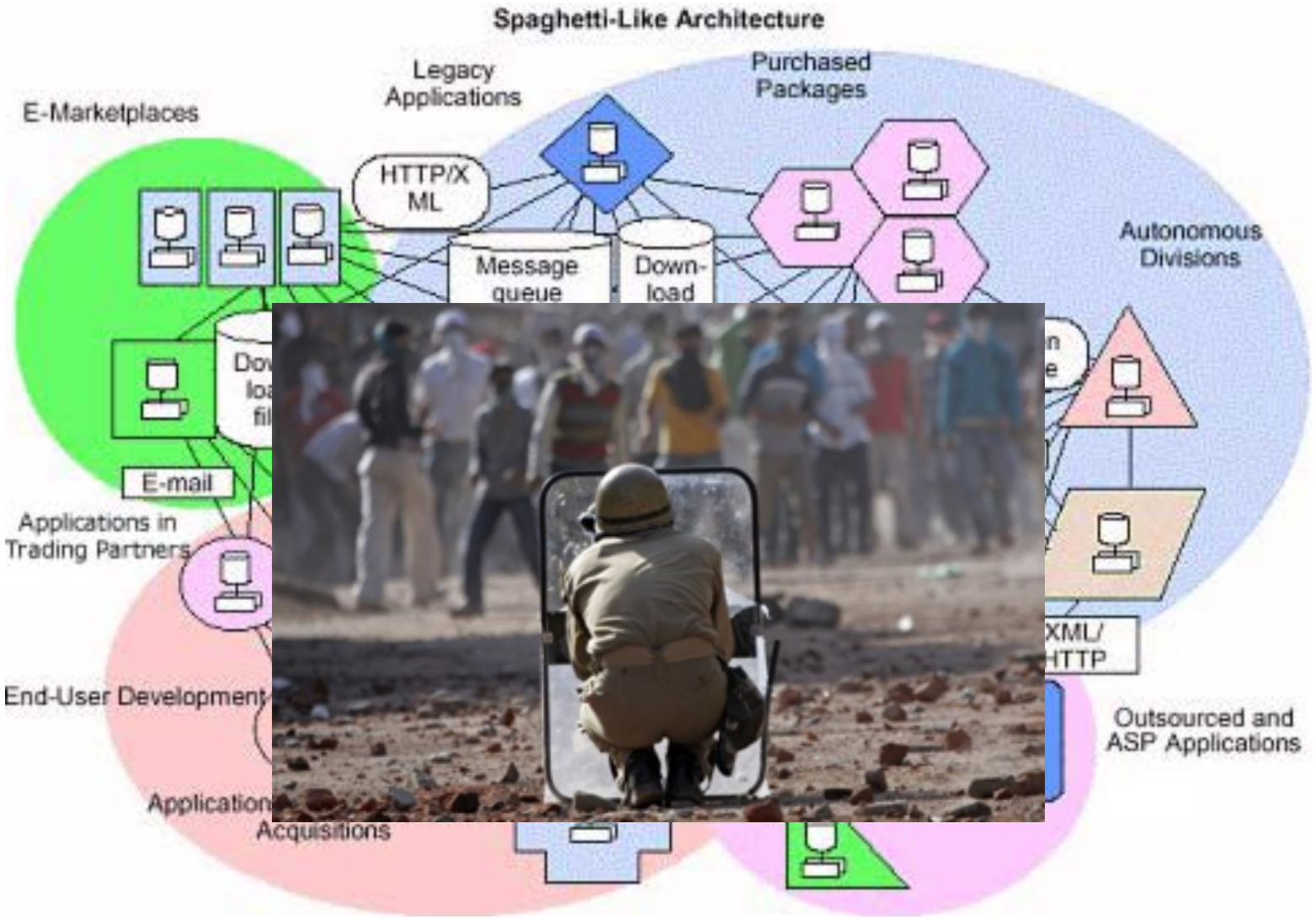
Sistema Operacional

http://www.treybailey.net/wp-content/uploads/2012/02/Broken_Promises_by_HerrFous.jpg

Como SOAP é...

http://www.ferapositivo.com.br/wp-content/uploads/2010/07/plug_br_intro1.jpg


Spaghetti-Like Architecture
Legacy Applications
Purchased Packages
E-Marketplaces
HTTP/XML
Message queue
Down-load file
Autonomous Divisions
FTP
Down-load file
Screen scrape
Trans-action file
Screen scrape
CICS gateway
Trans-action file
Sockets
RPC
ORB
Gateway
E-mail
Applications in Trading Partners
Trans-action file
Message queue
SMTP
Trans-action file
Message queue
CICS gateway
Browser
XML/HTTP
Down-load file
End-User Development
Message
APPC
Outsourced and ASP Applications
Applications From Mergers and Acquisitions


Spaghetti-Like Architecture
Legacy Applications
Purchased Packages
E-Marketplaces
HTTP/X ML
Message queue
Down-load
Autonomous Divisions
E-mail
Applications in Trading Partners
XML/ HTTP
End-User Development
Outsourced and ASP Applications
Acquisitions


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

Princípios e Características REST

Cliente-servidor

Stateless

Cacheable

Interface Uniforme

Princípios e Características 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 (todo o) recurso existente
DELETE	Remover um recurso
PATCH	Atualizar (parte de) um recurso existente

O nosso sonho na integração de sistemas...

http://evelynbourne.com/wp-content/uploads/2012/05/peace-hands.jpg

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 consertado?

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

REST Sagrado

Nível 3: Controles Hypermedia

Nível 2: Verbos HTTP

Nível 1: Recursos

Nível 0: O pântano do POX

Nivel 0: O pântano do POX

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

```
POST /agendamentoService HTTP/1.1
[headers...]

<appointmentRequest>
  <slot doctor = "rcmito" start = "1400" end = "1450"/>
  <patient id = "ederi"/>
</appointmentRequest>
```

Nivel 1: Recursos

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
[headers...]

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

Level 2: Verbos HTTP

Muitas URIs, utilizando **corretamente** os **verbos HTTP**
Uso correto dos **códigos** de resposta
Expõe estado e não comportamento
CRUD

```
GET /doutores/rcmito/slots?date=20100104 HTTP/1.1
Host: jogano10.com

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

Roy, os níveis 0, 1 e 2 são RESTful?

NÃO!

Roy Thomas Fielding

Level 3: Controles Hypermedia

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

Recursos auto descritivos

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

O resto é HTTP e links

```
HTTP/1.1 201 Created
Location: http://jogano10.com/slots/1234/appointment
[various headers]

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

HATEOAS

Hypermedia / Mime-types / Media-types

Descre

Descrever
Ar

HATEOAS

W

"Em cada mens
próxima mensagem"

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)

Isto é RESTful !

Roy Thomas Fielding

Fernando Meyer @fmeyer 11h

Segunda feira vai ser o festival nacional do "eu vi isso na palestra do X, borá fazer igual"

Retweeted by Eder Ignatowicz

Expand

Eder, esta é a track de Desafios no Java e na JVM.

http://www.badhaven.com/wp-content/uploads/2012/07/263143_Angry-Crowd3.jpg

JAX-RS API

JAX-RS 1.0 é a API Java para RESTful WS

POJO-Based

(sem complicações ou contratos, só annotations)

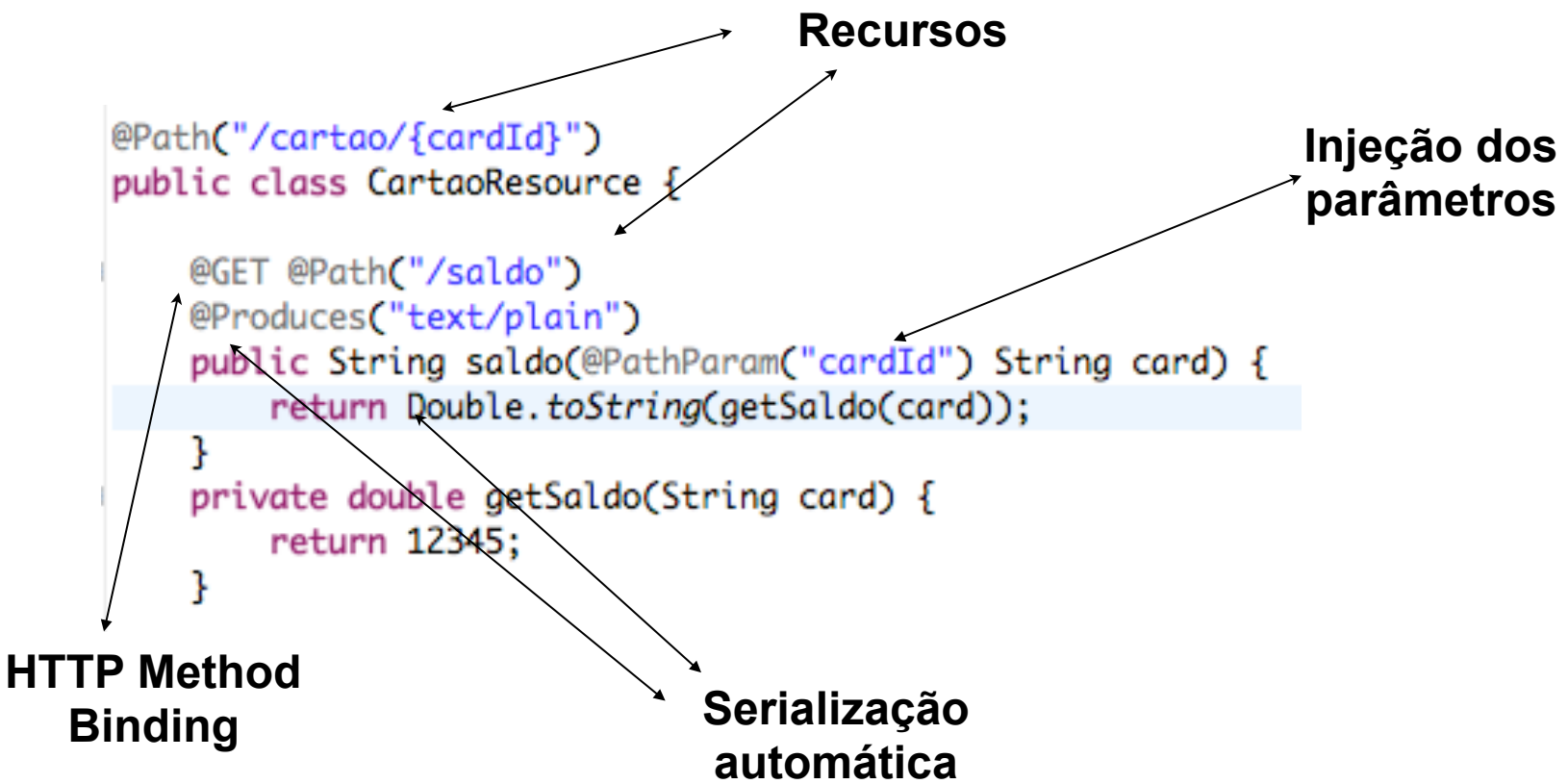
HTTP-Based

JAXB Based

Independente de container, formato

Parte do JavaEE

JAX-RS API




```
@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;
    }
```

Implementações JAX-RS

Triaxrs

Jersey

http://2.bp.blogspot.com/-JmDsZ1ESAWg/TyiHY8MMdzI/AAAAAAAAEDs/lFZxR0z5fGk/s1600/escalada1.jpg

Caching

Redução de latência e tráfego

Ampla infraestrutura de caching nos clients

browser, squid, http clients

HTTP Caching Features

allowed (or not)

expiration

intermediary caches allowed (or not)

validation

storable(or not)

Quando devo “Cachear”?

Expires header

```
Expires: Sun, 04 Aug 2012 16:00 GMT
```

Cache-control header

```
Cache-Control: no-cache
Cache-Control: public, max-age=3000
```

Validation Header

```
Last-Modified: Mon, 29 Jun 2012 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
```

Se não tiver nada disso, não “cacheia”

(por isto devemos implementar)

JAX-RS e Cache-Control

```
@Path("/doutores")
public class DoutoresService {
    @Path("/{id}")
    @GET
    @Produces("application/xml")
    public Response getDoutores(@PathParam("id") int id) {
        List<Doutor> doutores = //getDoutores
        CacheControl cc = new CacheControl();
        cc.setMaxAge(3000);
        return Response.ok(doutores).cacheControl(cc).build();
    }
}
```

JAX-RS e Conditional GETs

```
Last-Modified: Mon, 29 Jun 2012 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
```

O cache está valido?

304 "Not Modified"

Não está?

200 "OK" + recurso válido

```java
@Path("/doutores")
public class DoutoresService {
    @Path("/{id}")
    @GET
    @Produces("application/xml")
    public Response getDoutores(@PathParam("id") int id,
            @Context Request request) {
        EntityTag tag = // get tag mais atualizada
        ResponseBuilder builder = null;

        builder = request.evaluatePreconditions(tag);
        if (builder != null){
            return builder.cacheControl(cc).build();
        }
        Object doutores = // getDoutores
        return Response.ok(doutores).cacheControl(cc).build();
    }
}
```

JAX-RS e Cache-Control

```
@Path("/doutores")
public class DoutoresService {
    @P
    @GE
    @Pr
    pub                                          ) {


                                                 ild();
    }
}
```

O mesmo princípio se aplica para PUTs e POSTs condicionais (updates concorrentes)

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);
}
```

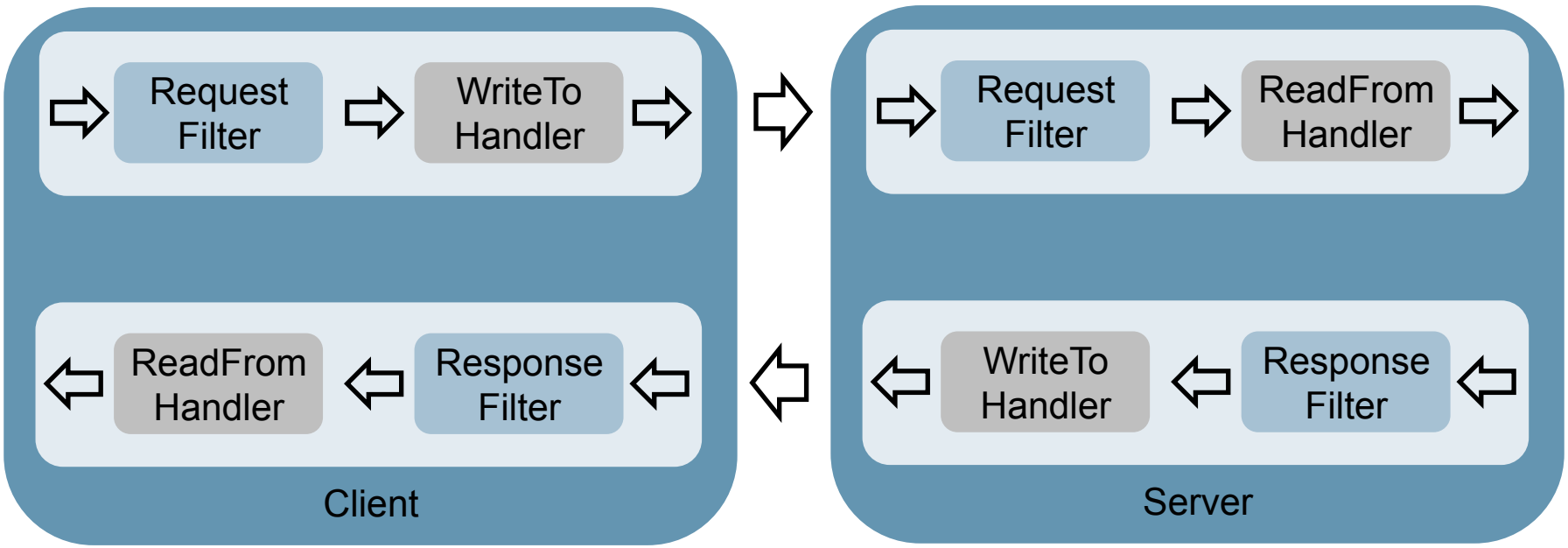
Interceptors/Handlers

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



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() {...}
}
```

HTTP “assíncrono”

O problema de ter uma conexão/request por thread

Alguns recursos demoram a responder (jdbc, web service)

Opção de solução:

Servidor envia um 202 (Accepted) como resposta
Posteriormente cliente busca resposta da requisição

Servlet 3.x

Thread é "congelada"

Após o job assíncrono ser processado, a thread é "reativada"

Servlet 3.x

```
@WebServlet("/foo" asyncSupported=true)
public class MyServlet extends HttpServlet {
     public void doGet(HttpServletRequest req,
HttpServletResponse res) {
          ...
          AsyncContext aCtx = request.startAsync(req, res);
          ScheduledThreadPoolExecutor executor = new
ThreadPoolExecutor(10);
          executor.execute(new AsyncWebService(aCtx));
     }
}
```

Mais informações no post do Rafael Sakuray
"Novidades no Servlet 3.1" no InfoQ Brasil

Async

Suporte na API Client

```
// 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) { … }
      });
```

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

Obrigado!!!

 @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

Bill Burke Scaling Jax-RS