

Put your systems to  
REST

Guilherme Silveira

# Guilherme Silveira



@guilhermecaelum

guilherme.silveira@caelum.com.br

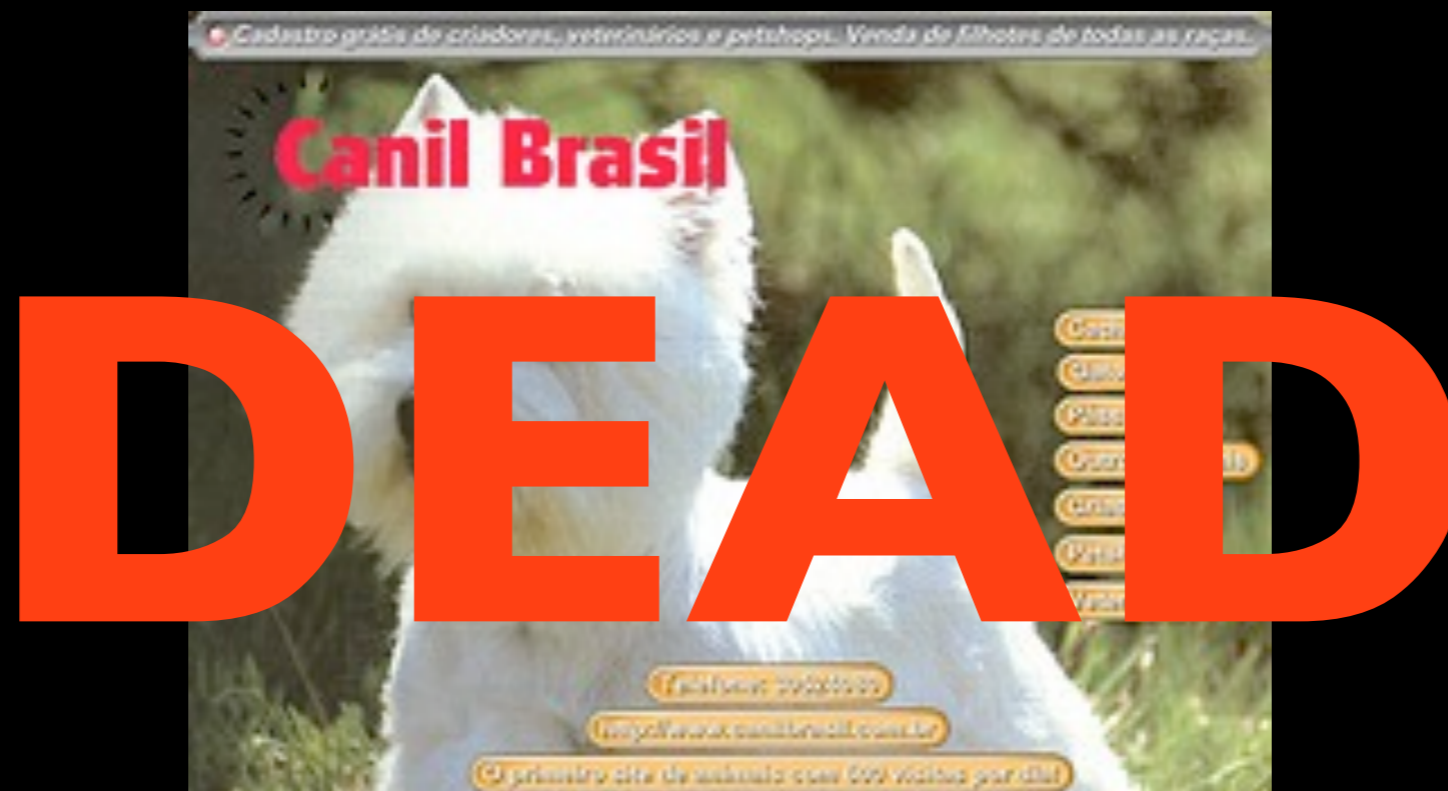
# são paulo



11 years old: learned “Basic”



# 1998 selling dogs



[www.canilbrasil.com.br](http://www.canilbrasil.com.br)

# 2002 guj.com.br



[www.guj.com.br](http://www.guj.com.br)

# 2003 vraptor

vraptor

ENGLISH

PORTUGUÊS

HOME

DOWNLOAD

DOCUMENTATION

BENEFITS

SUPPORT

VRAPTOR2



## TRY VRAPTOR3

- > A Java MVC web framework focused in fast development
- > Big support coming from community and VRaptor's developers.
- > Vast documentation available in Portuguese and English.



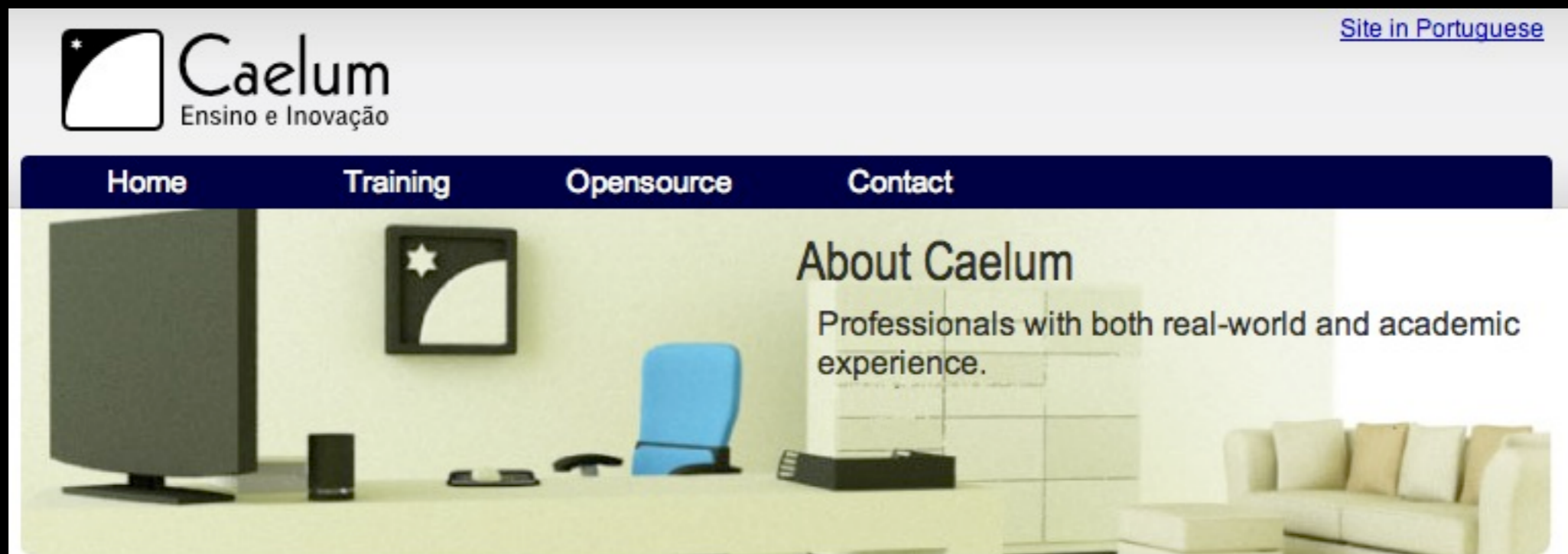
[GET TO KNOW VRAPTOR3](#)

**DOWNLOAD**

[www.vraptor.org/en](http://www.vraptor.org/en)



# 2004 caelum



[www.caelumobjects.com](http://www.caelumobjects.com)

# 2004 xstream



The screenshot shows the top portion of the XStream website. At the top left is the logo "XStream", where the "X" is a large, bold, green letter and "Stream" is in a black, italicized serif font. Below the logo is a navigation menu with three items: "Software", "About XStream", and "News". The "About XStream" item is highlighted with a green background. To the right of the navigation menu is a green horizontal bar containing the text "About XStream" in white. Below this bar, the text "XStream is a simple library to serialize objects to XML and back again." is displayed.

[xstream.codehaus.org](http://xstream.codehaus.org)

# 2004 extreme programming

# 2006 scrum

## Scrum (development)

---

From Wikipedia, the free encyclopedia

**Scrum** is an *iterative, incremental* methodology for project management often seen in *agile software development*.

Although Scrum was intended for management of software development projects, it can be used to run software maintenance teams, or as a general project/program management approach.

[http://en.wikipedia.org/wiki/Scrum\\_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development))

# 2009 restfulie

# Restfulie



QUIT PRETENDING

RAILS

JAVA

C#

SUPPORT

JAVA CHAPTER

TESTIMONIALS



## HYPERMEDIA

ALLOWS REAL LOOSE COUPLING BETWEEN CLIENTS AND SERVERS.



## SMALL

RESTFULIE IS SMALL.  
THE WEB IS YOUR STACK.



## CLIENT AND SERVER

EASY TO CODE CLIENTS, DEFAULT BEHAVIOR ON SERVERS.

<http://restfulie.caelumobjects.com/>

# 2009 lean

## Lean software development

From Wikipedia, the free encyclopedia



This article **needs additional citations for verification**.

Please help [improve this article](#) by adding [reliable references](#). Unsourced material may be [challenged](#) and removed.



It has been suggested that this article or section be [merged](#) into *Lean IT*. ([Discuss](#))

**Lean software development** is a translation of [Lean manufacturing](#) and [Lean IT](#) principles and practices to the [software development](#) domain. Adapted from the [Toyota Production System](#), a pro-lean subculture is emerging from within the [Agile](#) community.

[http://en.wikipedia.org/wiki/Lean\\_software\\_development](http://en.wikipedia.org/wiki/Lean_software_development)

# 2010 tectura



The screenshot shows the header of the tectura website. On the left is the logo, which consists of a blue square containing a white spiral, followed by the word "TECTURA" in a bold, italicized, sans-serif font. Below the logo is the tagline "DESENVOLVEDORES DISCUTINDO ARQUITETURA NA PRÁTICA". On the right side of the header, there is a navigation menu with the following items: "INÍCIO", "USUÁRIOS", "MINHAS DISCUSSÕES", "PREFERÊNCIAS", and "LOGOUT". Below the navigation menu is a search bar with the placeholder text "busca". At the bottom of the header, there is a row of social media-style tags: "#agile", "#arquitetura", "#java", "#csharp", "#design", "#dotnet", and "#rails".

<http://www.tectura.com.br>

# qcon 2010 são paulo, brazil

이지는 포르... 로 되어 있습니다. 번역하시겠습니까? 번역 번역 안함



The banner features a white background with a blue horizontal bar at the bottom. On the left, two blue silhouettes of people are shown; one is holding a laptop and emitting yellow signal waves towards the other. In the center, the text 'SÃO PAULO 2010' is written in large green letters, with '11 e 12 de setembro' in blue below it. The website 'www.qcon.com.br' is displayed in white on the blue bar. On the right, a blue silhouette of a person with a laptop is shown. To its right is the 'QCon' logo in green and blue, with 'SÃO PAULO' in grey below it. At the bottom right, the text 'THE ANNUAL INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE' is written in white on the blue bar.

**SÃO PAULO 2010**  
11 e 12 de setembro

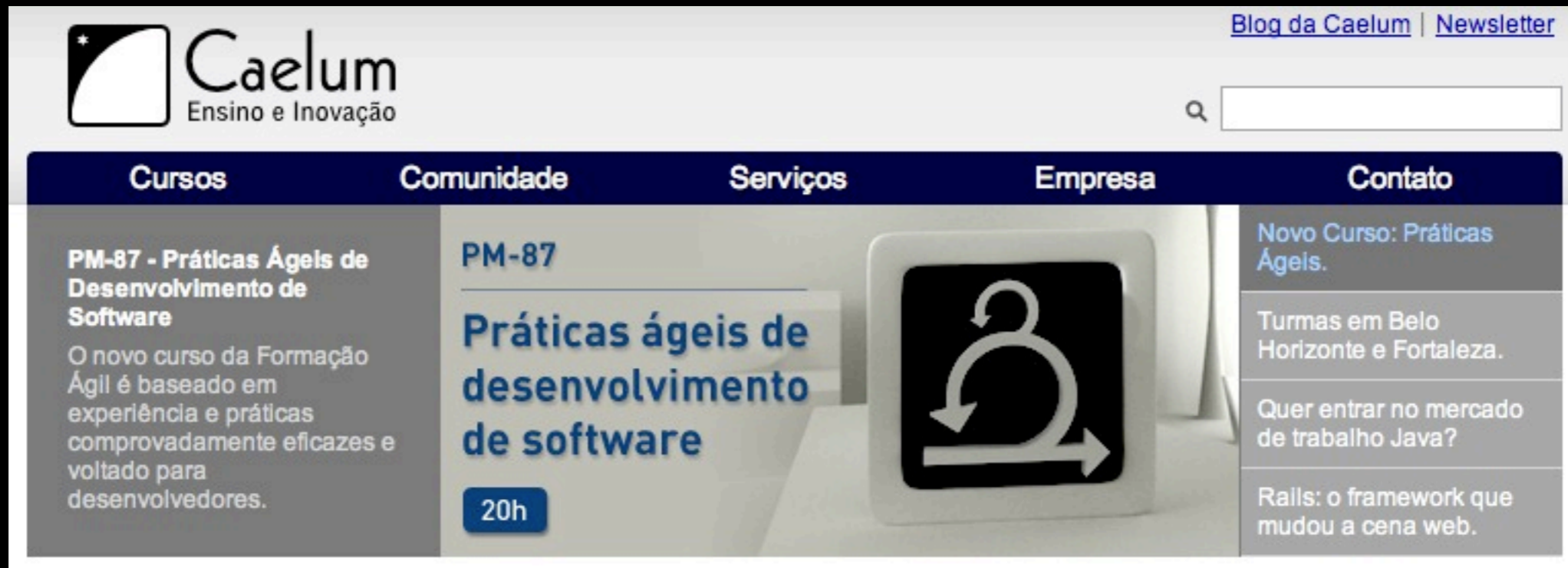
[www.qcon.com.br](http://www.qcon.com.br)

**QCon**  
SÃO PAULO

THE ANNUAL  
INTERNATIONAL  
SOFTWARE DEVELOPMENT  
CONFERENCE



# daily life



The screenshot shows the Caelum website interface. At the top left is the Caelum logo with the tagline 'Ensino e Inovação'. To the right are links for 'Blog da Caelum' and 'Newsletter', and a search bar. A navigation menu includes 'Cursos', 'Comunidade', 'Serviços', 'Empresa', and 'Contato'. The main content area features a course card for 'PM-87 - Práticas Ágeis de Desenvolvimento de Software', which is 20 hours long. A central image shows a person icon with a circular arrow. On the right, a sidebar lists 'Novo Curso: Práticas Ágeis.', 'Turmas em Belo Horizonte e Fortaleza.', 'Quer entrar no mercado de trabalho Java?', and 'Rails: o framework que mudou a cena web.'

<http://www.caelum.com.br>

<http://www.caelumobjects.com>

# blogs

[blog.caelumobjects.com](http://blog.caelumobjects.com)

[blog.caelum.com.br](http://blog.caelum.com.br)

[agilenomundoreal.com.br](http://agilenomundoreal.com.br)

bank.com calendar.com

$$\int_a^b \text{bank} + \text{travel} + \text{calendar} + \text{company}$$

travel.com company.com

CORBA

heaven?





EOJB

heaven?

SOAP

heaven?



what is REST?

what is the future  
of integration  
over the web?



what was REST?

# Restful Web

create a saas account

freeze account

reactivate account

## Services

Web

http

port 80

firewall heaven

Restful

Services

# Web Services

Restful

xml, json

get, post, ...

# Restful Web Services as of 2009

But what about Hypermedia?  
What about consumers?

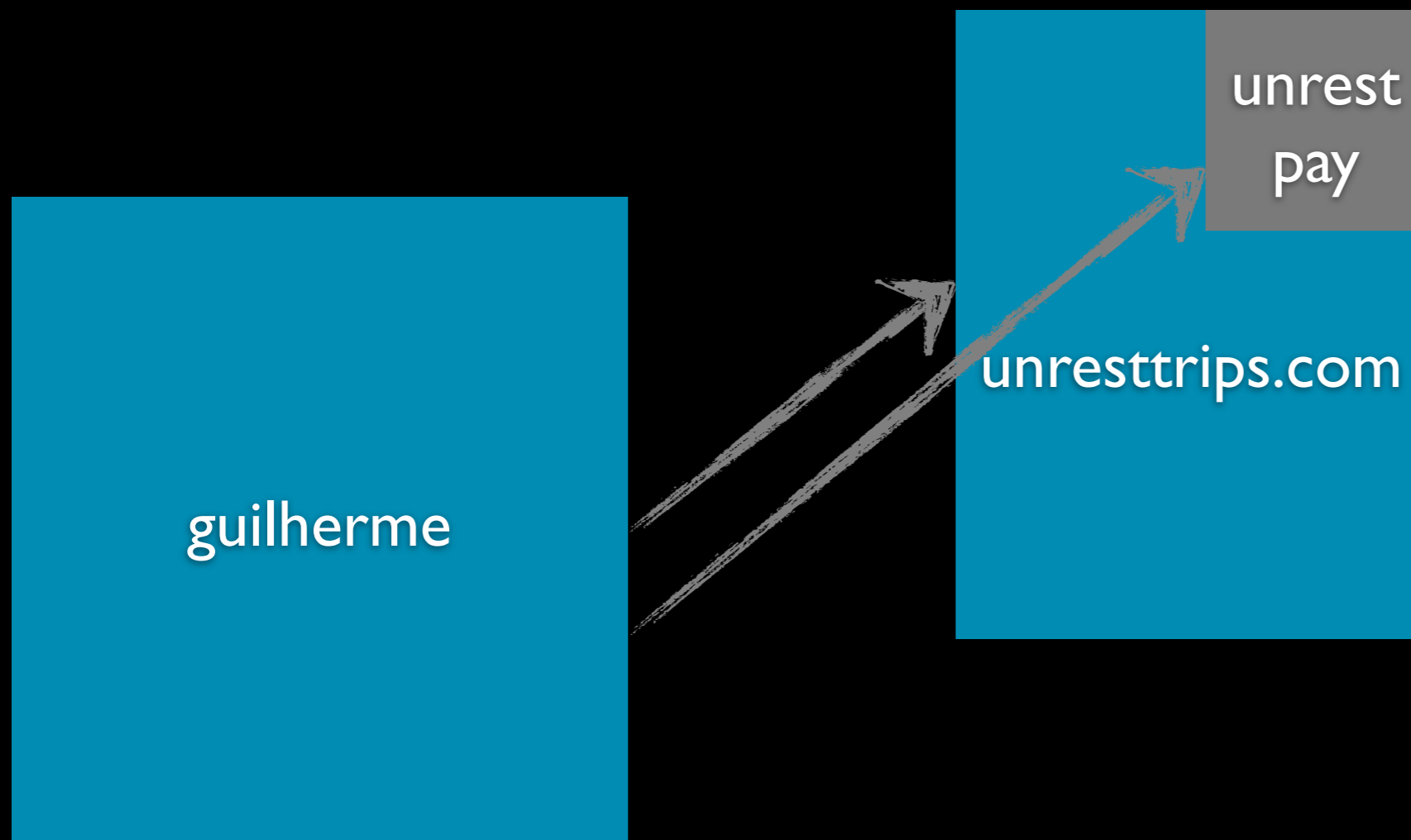
There is much more!

# unresttrips.com: flight details

```
<?xml version="1.0" encoding="UTF-8" standalume="yes"?>
<flight>
  <information>
    <from>sao paulo</from>
    <to>seoul</to>
  </information>
  <value>900.00</value>
</flight>
```

service locator when integrating:

coupling++



# resttrips.com: flight details

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<flight>
  <information>
    <from>sao paulo</from>
    <to>seoul</to>
  </information>
  <value>900.00</value>
  <link rel="payment"
        href="http://resttrips.com/payment/customer" />
</flight>
```



# resttrips.com: making the payment

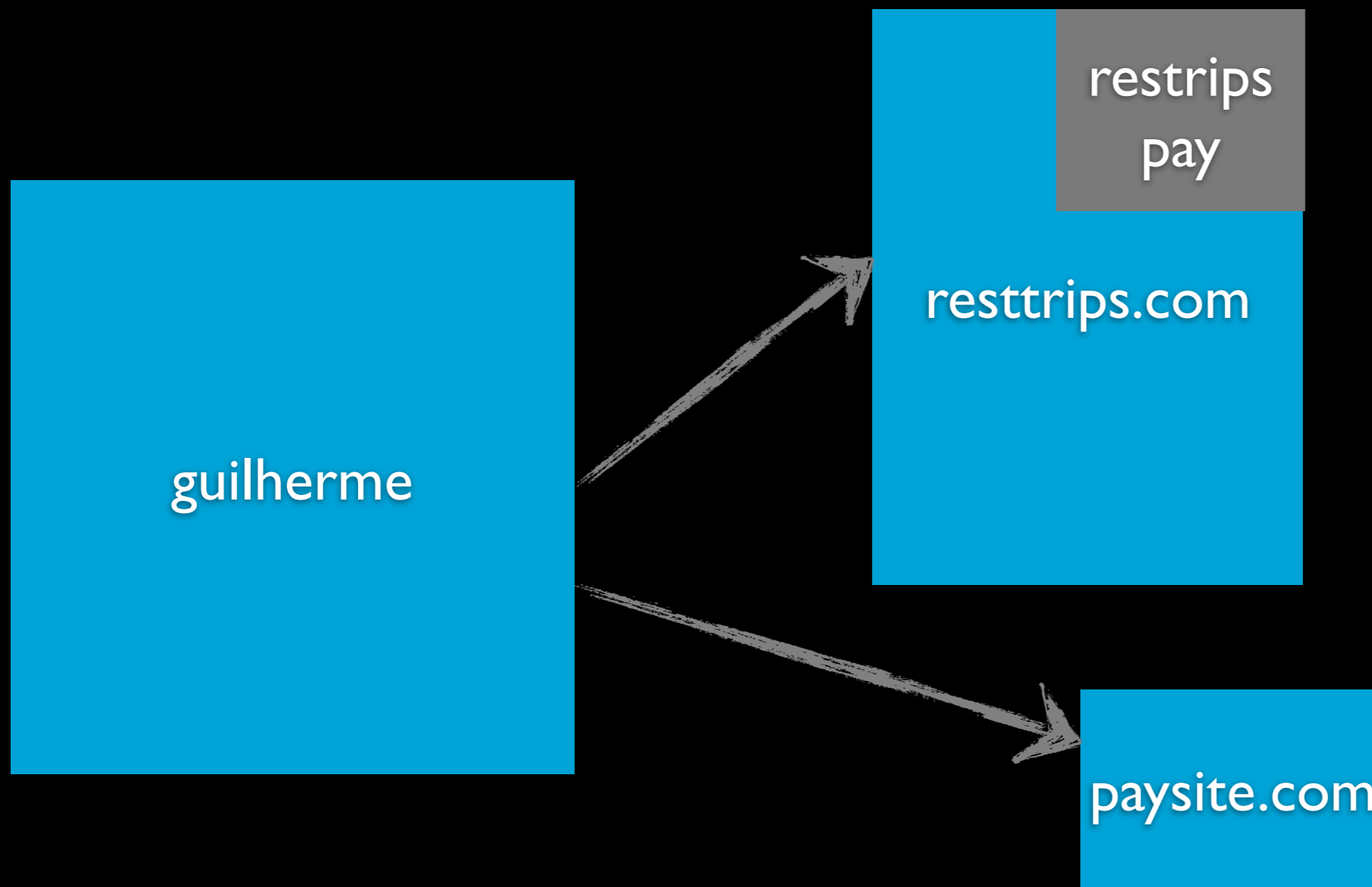
```
flight = Client.at( 'http://resttrips.com/f/574XR4' ).get();  
confirmation = flight.getLink( "payment" ).  
    patch(cardInformation, value);
```

# resttrips.com: changing its payment provider

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<flight>
  <information>
    <from>sao paulo</from>
    <to>seoul</to>
  </information>
  <value>900.00</value>
  <link rel="payment"
        href="http://paysite.com/custom" />
</flight>
```

dependency injection when integrating:

coupling--



**trip = resource**

<http://kayak.com/f/574XR4>

**payment = resource**

any uri unknown at compile time

KAYAK

Fri Mar 4 2011 – Tue Mar 8 2011

Toolbox

[Change your search](#)  
[Get a price alert](#)

A E L  
B R I T I  
T H O M S  
A I R C

Stops

- nonstop
- 1 stop
- 2+ stops

Flight Time

Take-off

Take-off (Depart Flight)

Fri 1:00

Toolbox

[Change your search](#)  
[Get a price alert](#)

Stops

- nonstop
- 1 stop
- 2+ stops

nonstop \$2047

1 stop \$1971

2+ stops

Flight Times

Take-off  Landing

Take-off (Depart Flight) [show all](#)

Fri 1:00a - Sat 12:00a

[Fly to Seoul from \\$179\\*](#)

Seoul Fares Just Dropped! Book Now to Lock In the Best Deals.  
[LowFares.com/Seoul-Fares](#)

566 of 569 roundtrips shown [show all](#)

« Prev – Next »

Filters Applied: Depart/Return, same airports

[Price\\*](#) ▲ [Airline](#) [Takeoff](#) [Landing](#) [Stops \(Duration\)](#)

**\$1971**

USD

Select

4 sites



Continental

GRU	10:25p	➔	ICN	4:30p	2 (30h 05m)
ICN	12:15p	➔	GRU	9:55a	2 (33h 40m)

[save](#)



**\$1971**

USD

Select

4 sites



Continental

GRU	10:25p	➔	ICN	4:30p	2 (30h 05m)
ICN	11:30a	➔	GRU	9:55a	2 (34h 25m)

[save](#)



i never travel alone

# my friend rick



# resttrips.com: sharing a trip

```
flight = Client.at('http://resttrips.com/f/574XR4').get();
```

```
confirmation = flight.getLink("payment").  
    patch(cardInformation, value/2);
```

```
/* send the payment link to another part of the web
```

```
flight = Client.at('http://resttrips.com/f/574XR4').get();
```

```
confirmation = flight.getLink("payment").  
    patch(cardInformation, value/2);
```



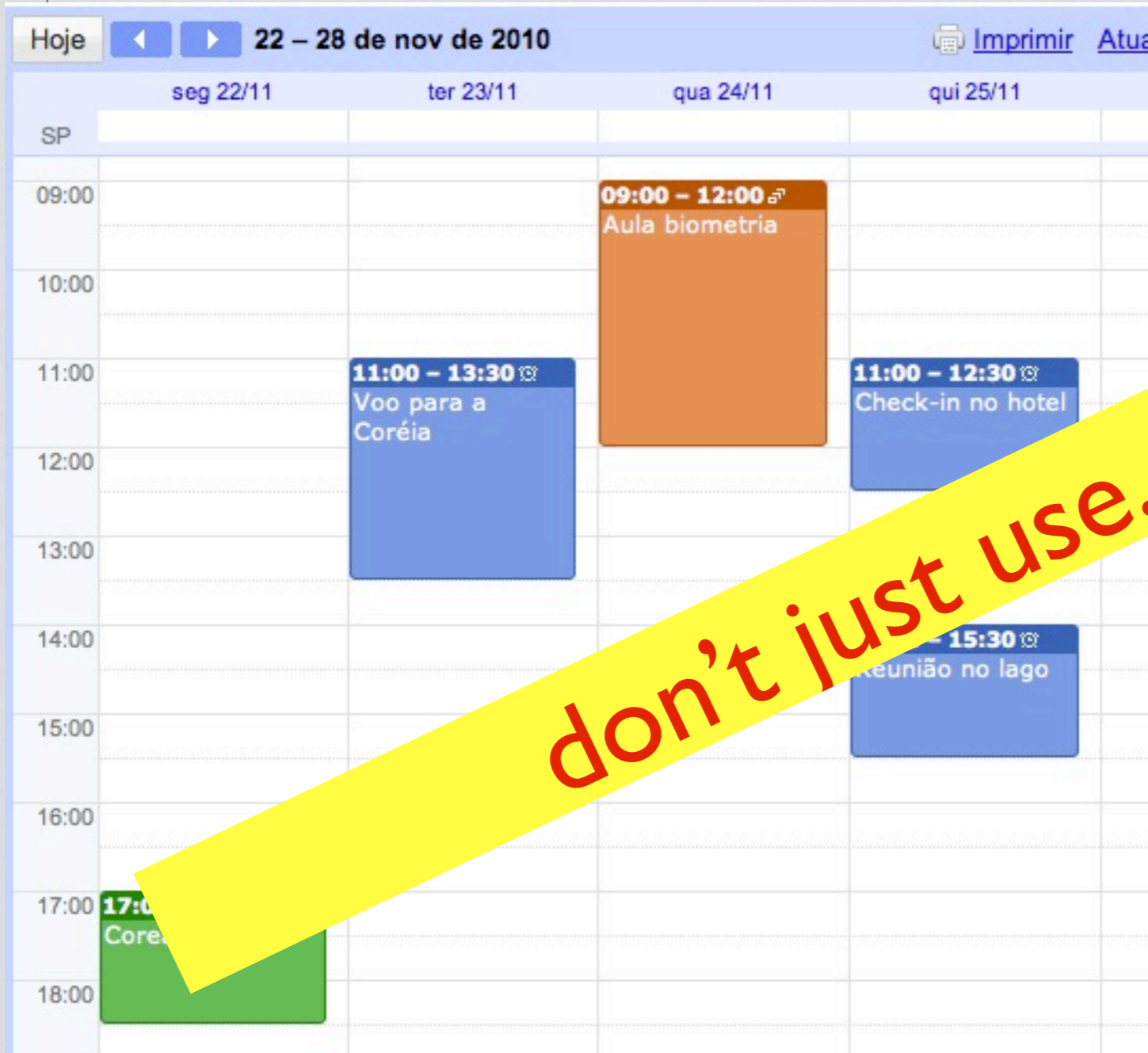
# calendar: integrating my systems

```
myself = Client.at( 'http://users.calendar.com' )  
                .with(auth).get();  
myself.link( "calendar" ).patch( flight.link( "self" ) );
```

# calendar: more examples

```
me.link("calendar").patch(link_to_birthday_list)
me.link("calendar").patch(link_to_hotel_reservation)
me.link("calendar").patch(link_to_trip_details)
```

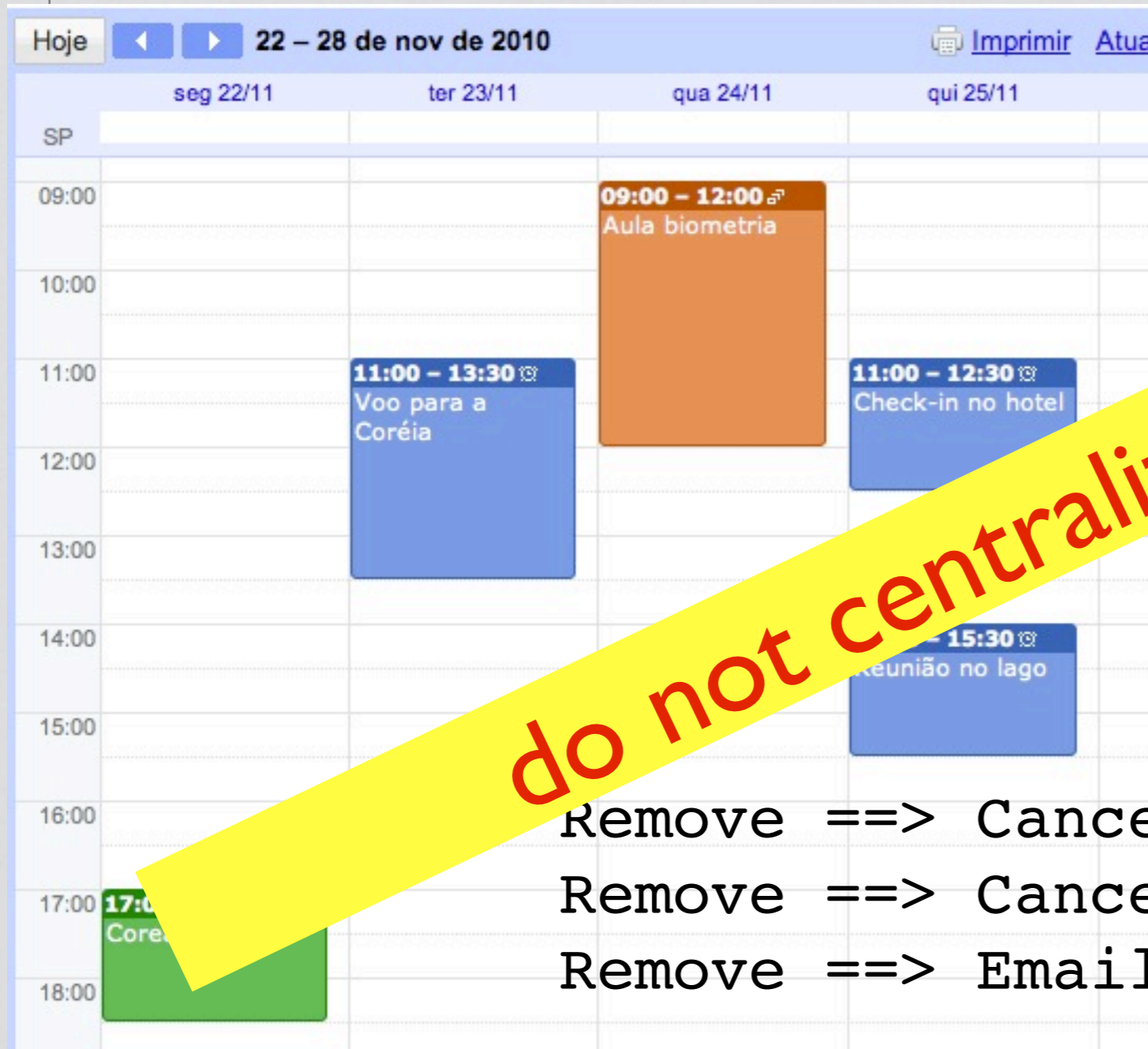
Any update on the flight ==> reflects here  
Any update on the hotel ==> reflects here  
Any update on the meeting ==> reflects here



**don't just use, integrate!**

so what?

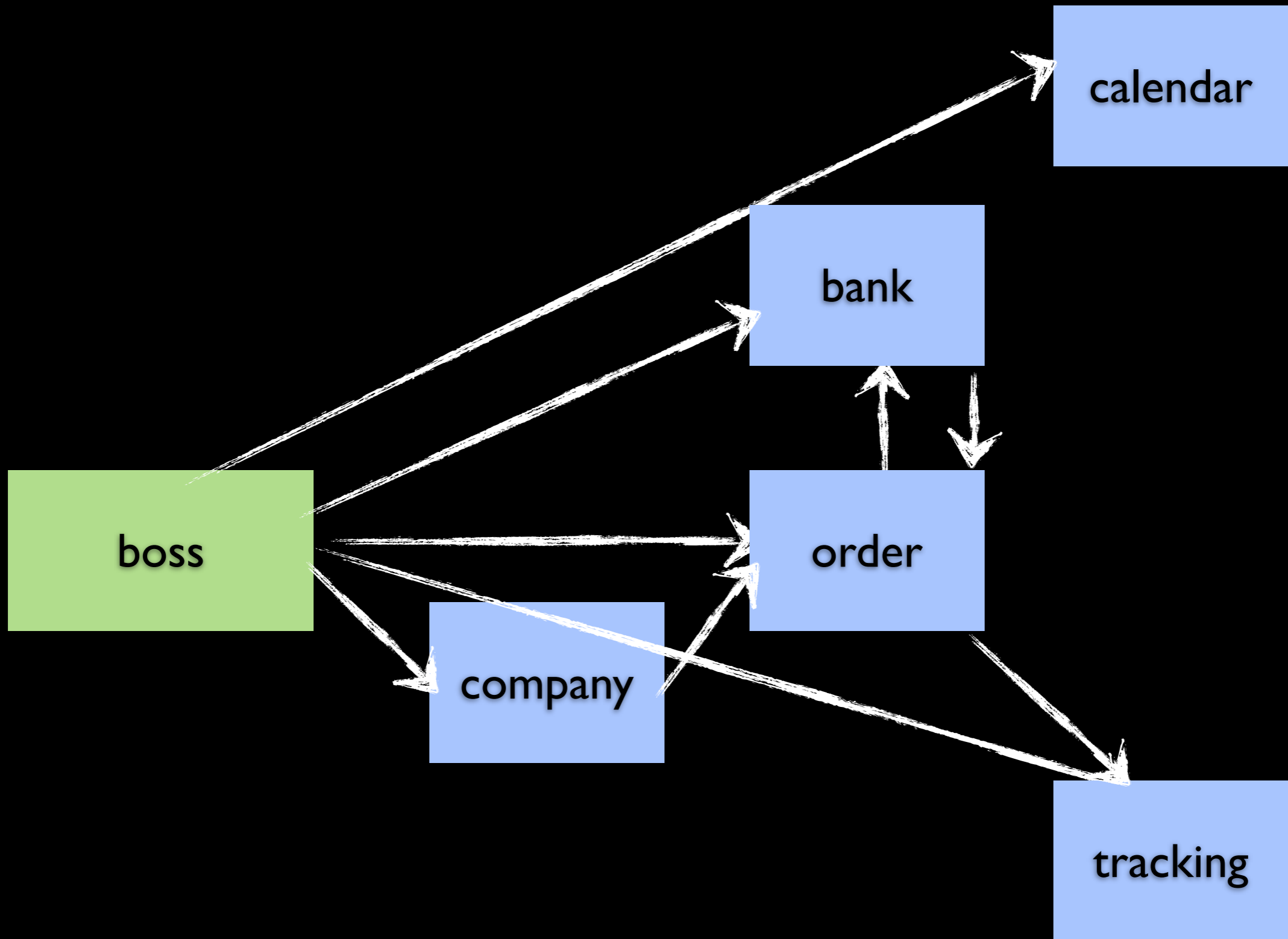
so what? that was just keeping an URI.



integration over the web

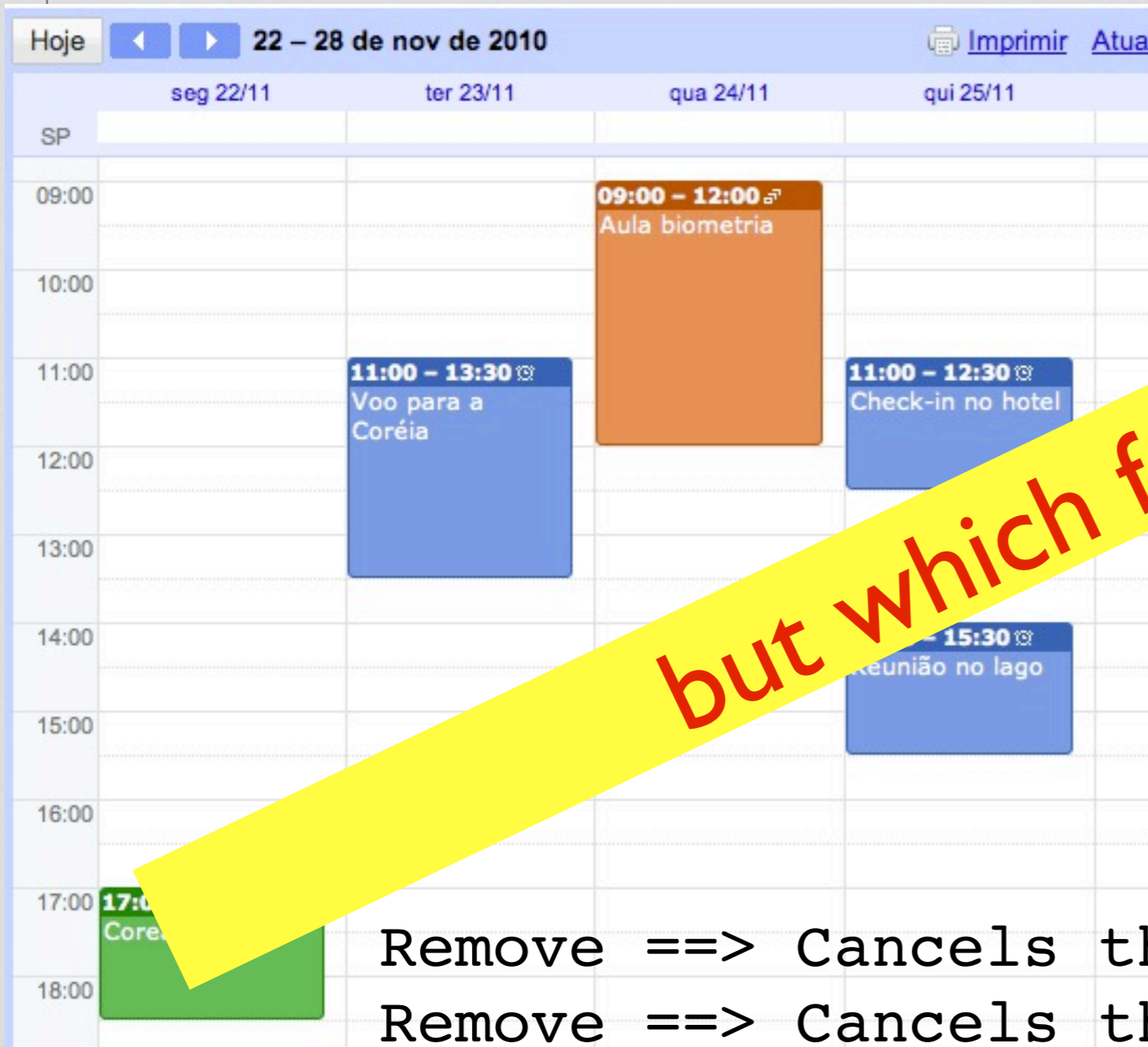
DECENTRALIZING the CONTROL  
delegating to multiple agents  
distributing your system

# integration over the web



does our 'rest'  
systems built in  
2010  
work this way?

so what? that was just keeping an URI.



- Remove ==> Cancels the flight
- Remove ==> Cancels the reservation
- Remove ==> Emails your coworkers



which payment or calendar?

#json, #xml,  
#soap #etc?

which #json, #xml  
#etc?

exercise

who is that guy?



who is that girl?

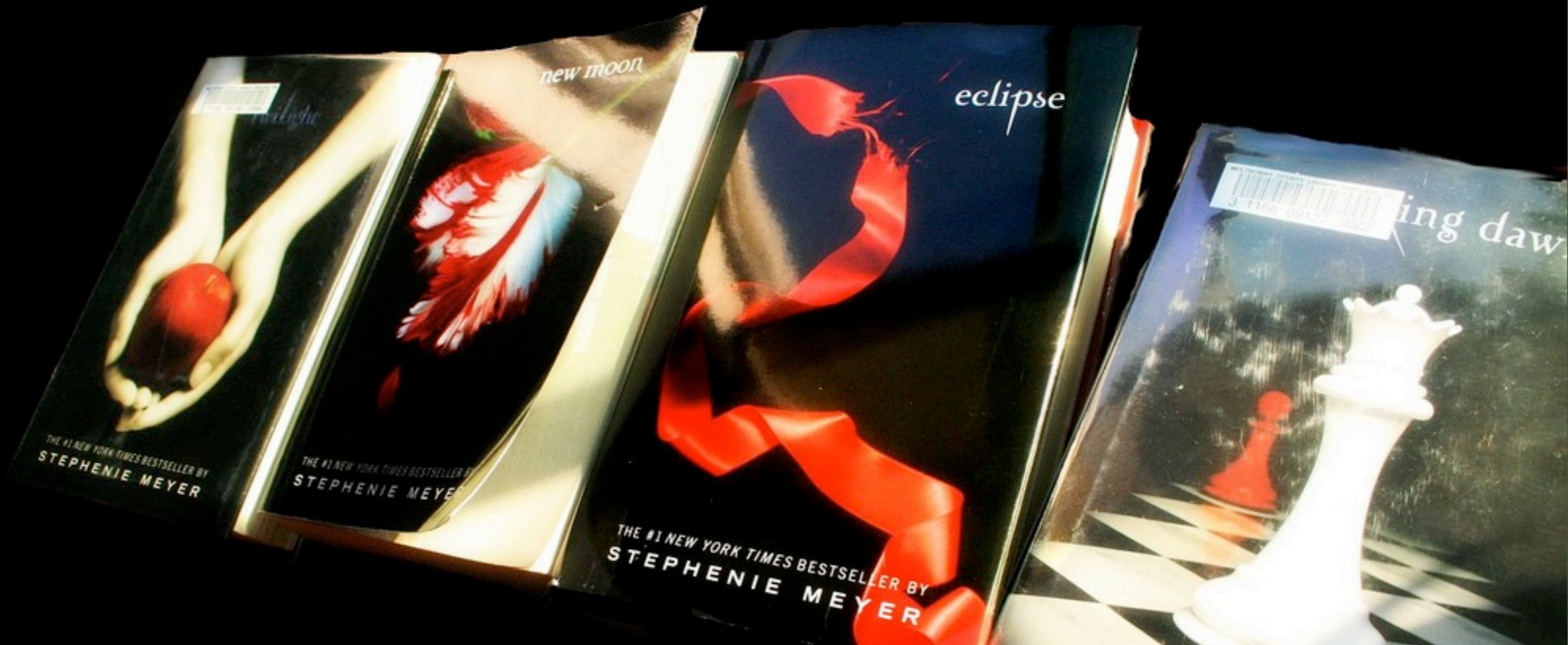


too easy, right? who is that little girl?



she is not my daughter

content without  
semantic has no value!





so what?



micro formats with hyperlinks!

micro formats, media types, rdf etc

# integration over the web

$$\int_{2010}^{+\infty} \text{web} = \text{semantics} + \text{rest}$$

# saving money

```
def flights_of(site)
  base = Restfulie.at(site).get
  flights = base.link("opensearch").get.fill("flight").with(:from =>
"GRU", :to => "JFK").get
end
```

```
voos = flights_of("kayak")
voos << flights_of("tam")
voos << flights_of("gol")
voos << flights_of("cvc")
```

```
cheapest_flight(voos).link("payment").post(card_hell)
```

# bank

```
flight = cheapest_flight(flights)

bank = Restfulie.at("my_bank").auth(myself).get

auth = bank.cards["lisa"]
        .authorization(1.min, flight.price)
flight.link("payment").post(auth)
```

# email

**Travel Document - Paris 11/12/10** avlao | X

★ **Orbitz** to Guilherme [show details](#)

**Images are not displayed.**  
[Display images below](#) - [Always display images from travelercare@orbitz.com](#)

**Your Travel Document**

Hello Guilherme, A

★ **Guilherme Silveira** to trips [show details](#)

Guilherme Silveira  
Caelum | Ensino e Inovação  
<http://www.caelum.com.br/>  
- Show quoted text -



## Devoxx Belgium 2010

Fri Nov 12 2010 to Wed Nov 24 2010

[My Trips](#) > Devoxx Belgium 2010

Friday, Nov 12 2010 - Paris, France

[Add event](#) ▾

6:15 pm

BRST



14h 25m

**São Paulo (GRU) to Paris (ORY)** - Conf #:

[Modify](#) ▾

Booking receipt received Thu Aug 26 2010. [report inaccuracies](#) [view original email](#)

This flight leaves on **Fri Nov 12 2010** and arrives on **Sat Nov 13 2010**.

**TAP Portugal**  
Flight 196

**Takeoff:** 6:15 pm BRST  
São Paulo (GRU) [map](#)

**Landing:** 6:15 am WET  
Lisbon (LIS) [map](#)

Seat(s): 15G

Layover in Lisbon (LIS) for 1 hour 55 minutes

**TAP Portugal**  
Flight 432

**Takeoff:** 8:10 am WET  
Lisbon (LIS) [map](#)

**Landing:** 11:40 am CET  
Paris (ORY) [map](#)

Seat(s): 4C

**Travelers:** GUILHERME SILVEIRA (Ticket# )

**Vendor info:** [Orbitz](#) (\$1,136.19 total cost) 408-757-5160 Ref#

[add notes](#)

Sat

**Devoxx Belgium 2010**

Fri, Nov 12 - Wed Nov 24 2010



TAP Portugal



Multiple



[Find a car](#) »

email

guilherme

super viagem

caelum

kayak



Goal me!



your process  
goal: buy something



# enter

The image is a screenshot of the Amazon.com homepage from November 5, 2010. At the top left is the Amazon logo. To its right, a personalized greeting reads: "Hello, Guilherme Silveira. We have [recommendations](#) for you. (Not Guilherme?)". Below this are navigation links: "Guilherme's Amazon.com", "Today's Deals", "Gifts & Wish Lists", and "Gift Cards". A search bar is positioned below the navigation, with "All Departments" selected in the dropdown menu. On the left side, a vertical menu lists various product categories: "Shop All Departments", "Books", "Movies, Music & Games", "Digital Downloads", "Kindle", "Computers & Office", "Electronics", "Home & Garden", "Grocery, Health & Beauty", "Toys, Kids & Baby", "Clothing, Shoes & Jewelry", "Sports & Outdoors", and "Tools, Auto & Industrial". Below this menu is a "Check This Out" section with a link for "Books for Kids & ...". The main content area features a large advertisement for the Kindle. On the left of the ad is a photograph of a Kindle device displaying "Chapter 1" of a book. To the right of the device, the text reads: "Kindle #1 Bestselling Product on Amazon". Below this text is a yellow "Order now" button with a play icon. The Amazon Kindle logo is in the bottom right corner of the ad.

# select

[Advanced Search](#)

[Browse Subjects](#)

[New Releases](#)

[Bestsellers](#)

[The New York Times® Bestsellers](#)

[Libros En Español](#)

[Bargain Books](#)

[Text](#)

## Harry Potter Paperback Box Set (Books 1-7) [Paperback]

[J.K. Rowling](#)  (Author)

★★★★☆  (520 customer reviews)

List Price: ~~\$86.93~~

Price: **\$50.85** & this item ships for **FREE with Super Saver Shipping**. [Details](#)

You Save: **\$36.08 (42%)**

**In Stock.**

Ships from and sold by **Amazon.com**. Gift-wrap available.

**Want it delivered Tuesday, May 25?** Order it in the next **6 hours and 0 minutes**, and choose **One-Day Shipping** at checkout. [Details](#)

**34 new** from \$49.99    **15 used** from \$50.84

**Formats**

Amazon Price    New from    Used from

Quantity:



Add to Cart

or

[Sign In](#) to turn on 1-Click order

or



Add to Cart with  
FREE Two-Day Shipping

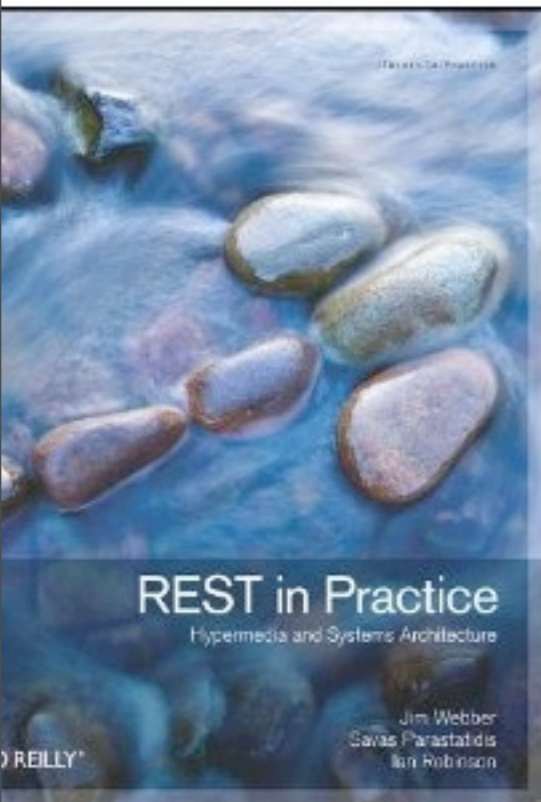
**Amazon Prime Free Trial**  
required. Sign up when you  
check out. [Learn More](#)

Add to Wish List

Add to Baby Registry



# select another one



## REST in Practice: Hypermedia and Systems Architecture [Paperback]

[Jim Webber](#) (Author), [Savas Parastatidis](#) (Author), [Ian Robinson](#) (Author)

List Price: ~~\$39.99~~

Price: **\$26.39** & this item ships for **FREE with Super Saver Shipping.** [Details](#)

You Save: **\$13.60 (34%)**

Pre-order Price Guarantee. [Learn more.](#)

**This title has not yet been released.**

You may pre-order it now and we will deliver it to you when it arrives.  
Ships from and sold by **Amazon.com**. Gift-wrap available.

Quantity:

[Pre-order: Add to Cart](#)

or

[Sign in](#) to turn on 1-Click order

or

[Pre-order with  
FREE Two-Day Shipping](#)

**Amazon Prime Free Trial**  
required. **Sign up when you**  
check out. [Learn More](#)

[Add to Wish List](#)

**Express Checkout with PayPh**

“[Guilherme's Numerous Articles](#)”

buy



Photos8.com

# sequence

```
list = Restfulie.at("http://localhost:3000/items").accepts("application/xml").get
basket = {:items => [[:id => list.item[1].id]]}
basket = list.basket.post!(basket, :root => "basket")
payment = {:cardnumber => "4850000000000001", :cardholder => "guilherme silveira", :amount => basket.price}
receipt = basket.payment.post!(payment.to_xml(:root => "payment"))
```

# analyze



# choose





# execute



# what comes next?



# what comes next?



do not expect!

adaptation

# scenarios

When there is a required item  
And there is a basket  
But didnt create a basket  
Then create the basket

When there is a required item  
And there is a basket  
Then add to the basket

When it is a basket  
And there are still products to buy  
Then start again

When there is a payment  
Then pay

mikyung (restfulie's simple rule engine)

# steps

```
When "there is a basket" do |resource|  
  resource.values.first.links.basket  
end
```

```
When "it is a basket" do |resource|  
  resource.keys.first == "basket"  
end
```

```
Then "pay" do |resource|  
  payment = {:payment => {:cardnumber => "4850000000000001"},  
            resource.basket.links.payment.post! payment  
end
```

# rule engines



Sample me!

search me

# describing my search

“This is how you search me”

1. fill in this form
2. send it to this uri using get
3. you will get the results in xml
4. if you want it in json, use this other uri, please
5. if you want the second page, do not forget this parameter

and my result

**in your desired format**

# opensearch descriptor

```
<Url type="application/atom+xml"  
    template="http://localhost:3000/products?q={searchTerms}  
&pw={startPage?}&format=atom" />
```

# opensearch descriptor

```
<Url type="application/atom+json"  
    template="http://localhost:3000/products?q={searchTerms}  
&pw={startPage?}&format=json" />
```

# opensearch descriptor

```
$ curl http://localhost:3000/products/opensearch.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>  
  <OpenSearchDescription xmlns="http://a9.com/-/spec/  
opensearch/1.1/">  
    <ShortName>Restbuy</ShortName>  
    <Description>Restbuy search engine.</Description>  
    <Tags>restbuy</Tags>  
    <Contact>admin@restbuy.com</Contact>  
    <Url type="application/atom+xml"  
      template="http://localhost:3000/products?q={searchTerms}  
&pw={startPage?}&format=atom" />  
  </OpenSearchDescription>
```



# opensearch result

```
curl http://localhost:3000/products?q=apple -H "Accept: application/atom+xml"
```

```
<feed xmlns="http://www.w3.org/2005/Atom">  
  <entry>  
    <link href="http://localhost:3000/products/1" rel="self" type="application/atom+xml"/>  
    <id>1</id>  
    <name>Apple macbook pro</name>  
    <price>1000.0</price>  
  </entry>  
  <link href="http://localhost:3000/orders" rel="order" type="application/xml"/>  
</feed>
```

# consuming opensearch

```
uri = "http://localhost:3000/products/opensearch.xml"  
type = 'application/opensearchdescription+xml'  
description = Restfulie.at(uri).accepts(type).get.resource
```

```
items = description.use("application/atom+xml").  
    search(:searchTerms => what, :startPage => 1)
```

```
items.resource.entries.size
```

# consuming another system

```
uri = "http://localhost:3000/products/opensearch.xml"  
type = 'application/opensearchdescription+xml'  
description = Restfulie.at(uri).accepts(type).get.resource
```

```
items = description.use("application/atom+xml").  
    search(:searchTerms => what, :startPage => 1)
```

```
items.resource.entries.size
```

# consuming another system

```
uri = "http://localhost:3000/products/opensearch.xml"  
type = 'application/opensearchdescription+xml'  
description = Restfulie.at(uri).accepts(type).get.resource
```

```
items = description.use("application/atom+xml").  
    search(:searchTerms => what, :startPage => 1)
```

```
items.resource.entries.size
```

consume multiple  
searches

costs?

# 201 search variation

Sample me!



# calendar server example

```
content = "BEGIN:VCALENDAR
CALSCALE:GREGORIAN
PRODID:iCalendar-Master
VERSION:2.0
BEGIN:VEVENT
DTSTART;VALUE=DATE:#{time_format(@order.expected_delivery)}
DTEND;VALUE=DATE:#{time_format(@order.expected_delivery)}
SUMMARY:Delivering order #{@order.id}"
```

# calendar client example

Hoje 29 de nov – 5 de dez de 2010 Imprimir Atualizar Dia Semana Mês 4 dias Compro

seg 29/11 ter 30/11 qua 1/12 qui 2/12 sex 3/12 sáb 4/12 dom 5/12

Q Q S S D  
1 2 3 4 5  
8 9 10 11 12  
15 16 17 18 19  
22 23 24 25 26  
29 30 31 1 2  
5 6 7 8 9

agendas  
me Silveira  
Caelum  
Configurações

agendas  
uma agenda de um am  
n Holidays  
stbuy.herokuapp...

SP  
11:00  
12:00  
13:00  
14:00  
15:00  
16:00  
17:00  
18:00  
19:00  
20:00

aniversario mae  
11:00 – 12:30  
Frances  
11:30 – 12:30  
pagar aluguel  
12:30 – 15:00  
paino  
19:30 – 21:00  
Piano

rest helps products for  
humans too

evolve your calendar into  
gmail

# server code

---

```
X-GOOGLE-CALENDAR-CONTENT-TITLE:Buying order #{@order.id}
X-GOOGLE-CALENDAR-CONTENT-ICON:http://www.google.com/calendar/
X-GOOGLE-CALENDAR-CONTENT-URL:#{order_url(@order)}
X-GOOGLE-CALENDAR-CONTENT-TYPE:text/html
X-GOOGLE-CALENDAR-CONTENT-WIDTH:640
X-GOOGLE-CALENDAR-CONTENT-HEIGHT:480
```

# client view

# client view++

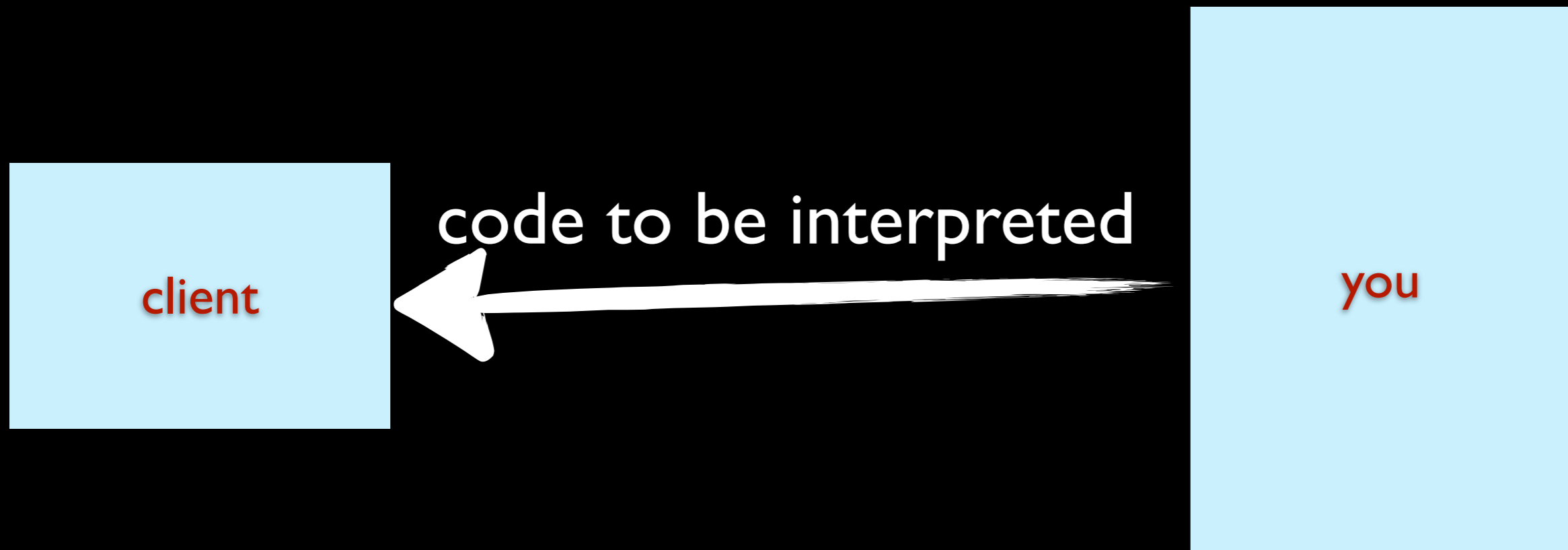
**your process**



**your flow**

# a domain application process

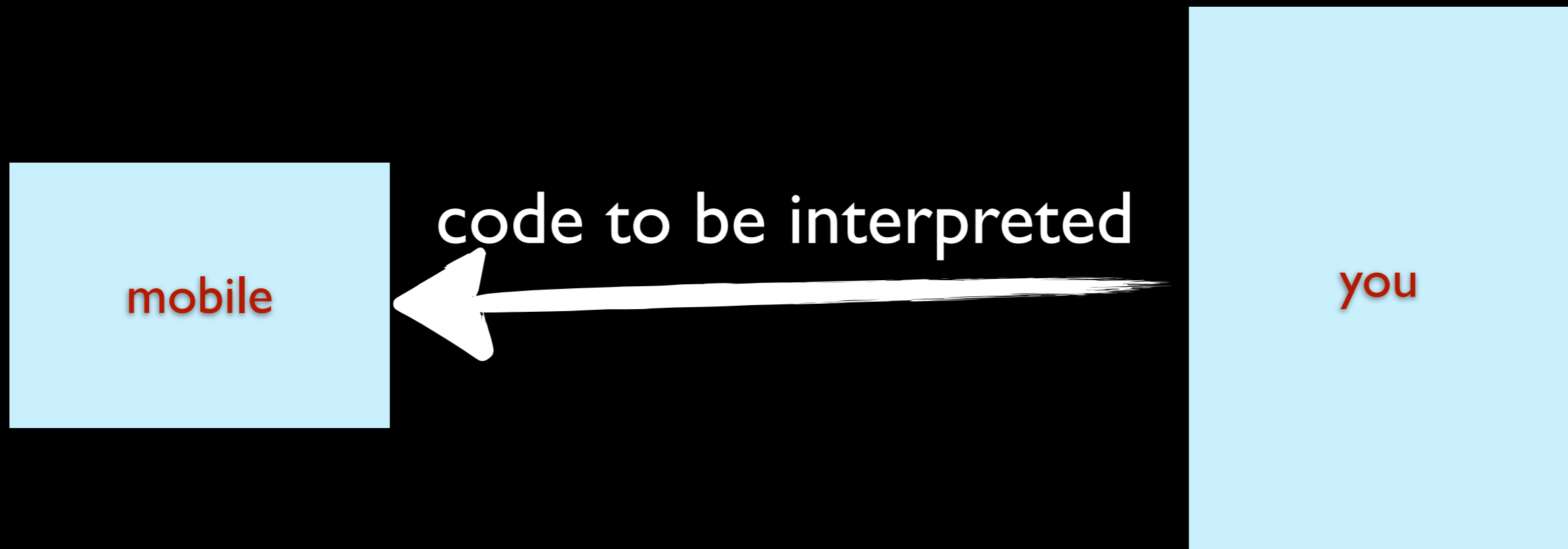
# teach your client



one existing example?

# custom mobile applications

# teach your mobile client



learn

hundreds of existing  
examples?



human web+-

JSONP

code on demand

same security issues

machine web

javascript

**lisp**

suggested by Pedro Teixeira

but my client



my client is not clever

**my client is a browser**

we are talking about js

but my client

my client is not clever



# client runs somewhere



somewhere

pay attention

www2010

<http://iansrobinson.com/>

<http://www.amundsen.com/blog/>



# javascript is everywhere



javascript



# subbu's blog

[subbu.org](http://subbu.org)

# composite media types

atom

xhtml

websockets

events

negative



positive

must ignore

Thoughtworks anthology book

# bottom up

design by committee

mime type?

microformat

controlled vocabulary

# hypermedia

integration over the web

# code on demand

extend on demand

[guilherme.silveira@caelum.com.br](mailto:guilherme.silveira@caelum.com.br)

<http://www.caelumobjects.com>

thank you

Put your server to  
REST

leonard richardsons model

# Server Maturity

1 uri, 1 http verb

```
/services.do?action=install&...
```

```
@Path("/services")
```

```
public class Services {
```

```
    @GET
```

```
    public Response services(
```

```
        @QueryParam("action") String action) {
```

```
        ServiceFactory factory = new ServiceFactory();
```

```
        Service service = factory.getServiceFor(action);
```

```
        return service.execute();
```

```
    }
```

```
}
```



# Server Maturity

1 uri, 1 http verb

```
public class InstallService {  
  
    public Response execute() {  
        return Response.ok()  
            .type("application/xml")  
            .entity("<service>...</service>")  
            .build();  
    }  
}
```

But?

1 uri, 1 http verb

/services.do?action=install&...



I know them all beforehand.

1 uri, 1 http verb

/services.do?action=install&...



Any change will break all our clients!

a priori knowledge

=

coupling++

# Server Maturity

Multiple uris, 1 http verb

```
/install?...
```

```
@Path("/services")  
public class Services {  
  
    @GET @Path("install")  
    public Response install() {  
        return new InstallService().execute();  
    }  
  
    @GET @Path("uninstall")  
    public Response uninstall() {  
        return new UninstallService().execute();  
    }  
}
```

but...

GET well

GET should not imply in  
undesireable effects



POST well

GET could help us with  
cache!

# Server Maturity

Multiple uris, multiple verbs

```
POST /cloud  
GET /cloud/1547437/software  
POST /cloud/1547437/software
```

# rails + restfulie

rubyonrails.org

```
class SoftwaresController < ApplicationController

  acts_as_restfulie
  respond_to :xml, :json

  def create
    @item = Item.create(params[:item])
    respond_with @item, :status => :created
  end

  def show
    @item = Item.find(params[:id])
    respond_with @item
  end
end
```

# Vraptor

[vraptor.org/en](http://vraptor.org/en)

```
@Resource
```

```
public class SoftwareResource {
```

```
    public void install() {
```

```
        Software software = SoftwareRepository.register  
                                                    (software);
```

```
        response.created(software);
```

```
    }
```

```
    public void uninstall() {
```

```
        response.deleted(software);
```

```
    }
```

```
}
```

# JAX-RS

```
@Path("/softwares")
public class SoftwareResource {

    @POST @Consumes("application/xml")
    public Response install(Software software) {
        software = SoftwareRepository.register(software);
        long id = software.getId();
        URI uri = UriBuilder.fromPath("/softwares/" + id)
                            .build();

        software.install();
        return Response.created(uri).build();
    }
}
```

# JAX-RS

```
@DELETE @Path("{id}")
public Response uninstall(@PathParam("id") Long id) {
    Software software = SoftwareRepository.retrieve(id);
    software.uninstall();
    return Response.ok().build();
}
```

Uniform Interface++  
Resources++

# Http verbs example

- POST creates
- PUT replaces
- PATCH updates
- DELETE removes
- GET retrieves
- OPTIONS tells me what i can do
- ...



Is a Restful service  
a cute CRUD?

yes

but there is more!

# hypertextmedia

enter amazon.com

follow links

search (GET)

search (GET)

create a basket (POST)

fill forms

create a payment (POST)

# Cloud API

GET /user/15

retrieves an user

follow GET

accesses its  
machines

follow POST self

installs a new  
machine

follow POST

pay for it

# retrieves an user



```
$ curl http://localhost:9998/user/574 -i  
HTTP/1.1 200 OK  
...
```

# user

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<user>
  <link rel="machines" href="/user/574/machines" />
  <name>guilherme silveira</name>
  <website>www.caelum.com.br</site>
  <link rel="payment" href="/payment/custom" />
  ...
</user>
```



# maschinemaschinemaschine

```
$ curl http://localhost:9998/user/574/machines -i  
HTTP/1.1 200 OK  
Content-Type: application/xml
```



# maschine

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<machines xmlns:ns2="http://www.w3.org/2005/Atom">
  <maschine>
    <ns2:link rel="self" href="/user/574/machines/1"/>
    <ns2:link rel="software" href="/machines1/softwarees"/>
    <id>1</id>
    <host>www.caelum.com.br</host>
    <softwarees>
      <software>
        <id>1234</id>
        <ns2:link rel="self" href="/softwarees/1234"/>
      </software>
    </softwarees>
  </maschine>
</machines>
```

# payment

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<user>
  <link rel="machines" href="/user/574/machines" />
  <name>guilherme silveira</name>
  <website>www.caelum.com.br</site>
  <link rel="payment" href="/payment/custom" />
  ...
</user>
```

**#buthow?**

```

@XmlRootElement
@XmlType(propOrder= {"id", "host", "softwares", "links"})
public class Machine {
    private final all variables here:

    @XmlElement(name="link", namespace="http://www.w3.org/2005/Atom")
    public List<Link> getLinks() {
        return Arrays.asList(
            Link.to("/machines/" + getId(), "self"),
            Link.to("/machines/" + getId() + "/softwares", "softwares")
        );
    }

    @XmlElementWrapper(name="softwares")
    @XmlElement(name="software")
    public List<Software> getSoftwares() {
        return softwares;
    }

    public void install(Software software) {
        getSoftwares().add(software);
    }

    public void uninstall(Software software) {
        getSoftwares().remove(software);
    }
}

```

# Machine.java

# MachineResource

```
@Path("/machines")
public class MachineResource {

    @Path("/{id}/softwares")
    public SoftwareResource softwares(@PathParam("id") Long id) {
        Machine machine = new MachineRepository().retrieve(id);
        if (machine != null) {
            SoftwareResource softwareResource = new SoftwareResource();
            softwareResource.setMachine(machine);
            return softwareResource;
        }
        throw new WebApplicationException(404);
    }

    @POST @Consumes("application/xml")
    public Response create(Machine machine) {
        Long id = new MachineRepository().save(machine);
        return Response.created(UriBuilder.fromPath("/") + id).build().build();
    }

    @GET @Path("/{id}")
    @Produces("application/xml")
    public Machine show(@PathParam("id") Long id) {
        return new MachineRepository().retrieve(id);
    }

    @GET
    @Produces("application/xml")
    public Machines list() {
        Machines machines = new Machines();
        machines.setMachine(new MachineRepository().list());
        return machines;
    }
}
```

# Server Model

according to

Leonard Richardson, 2008

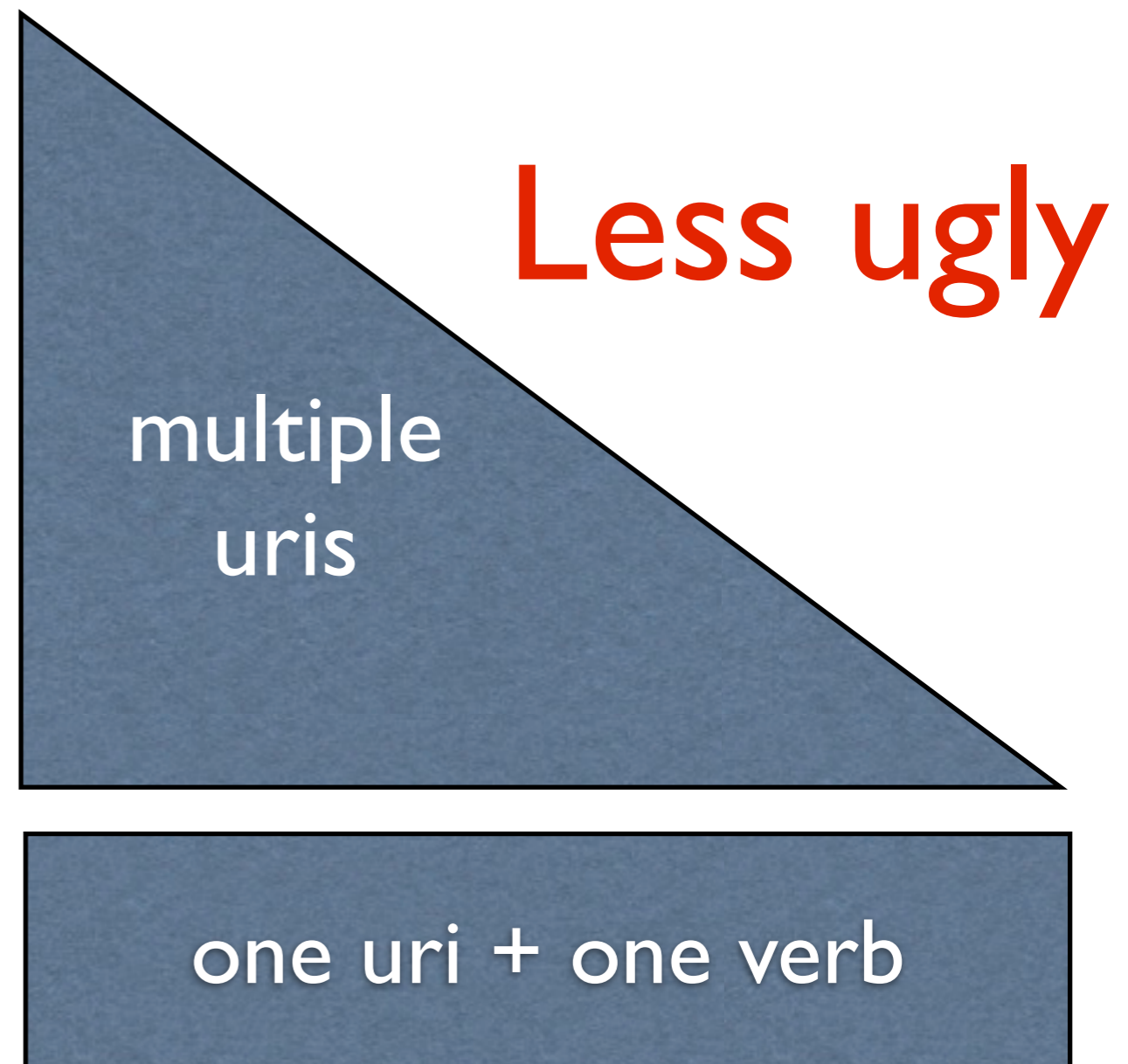
Ugly

one uri + one verb

# Server Model

according to

Leonard Richardson, 2008

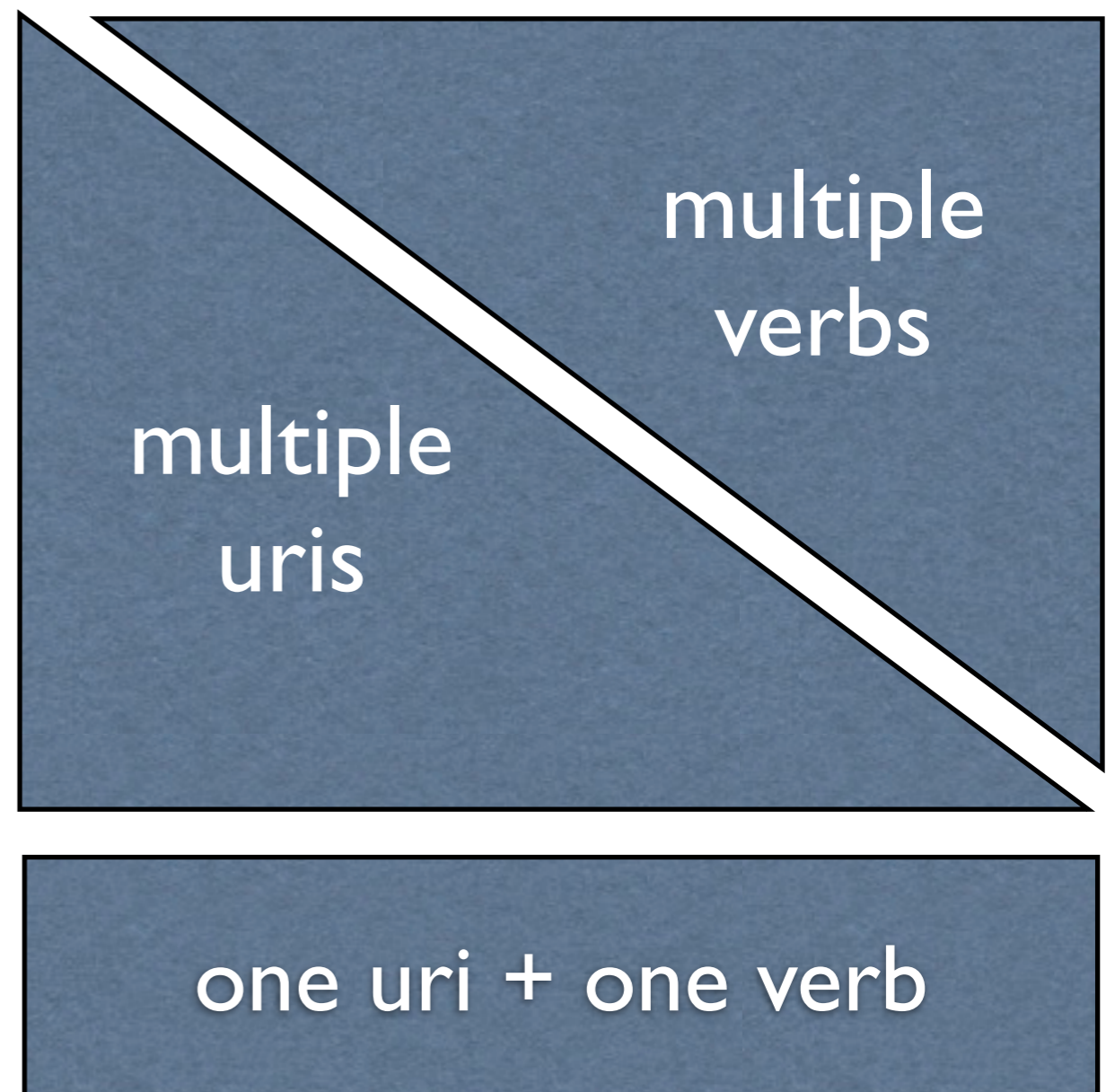


# Server Model

according to

Leonard Richardson, 2008

cute **CRUD**



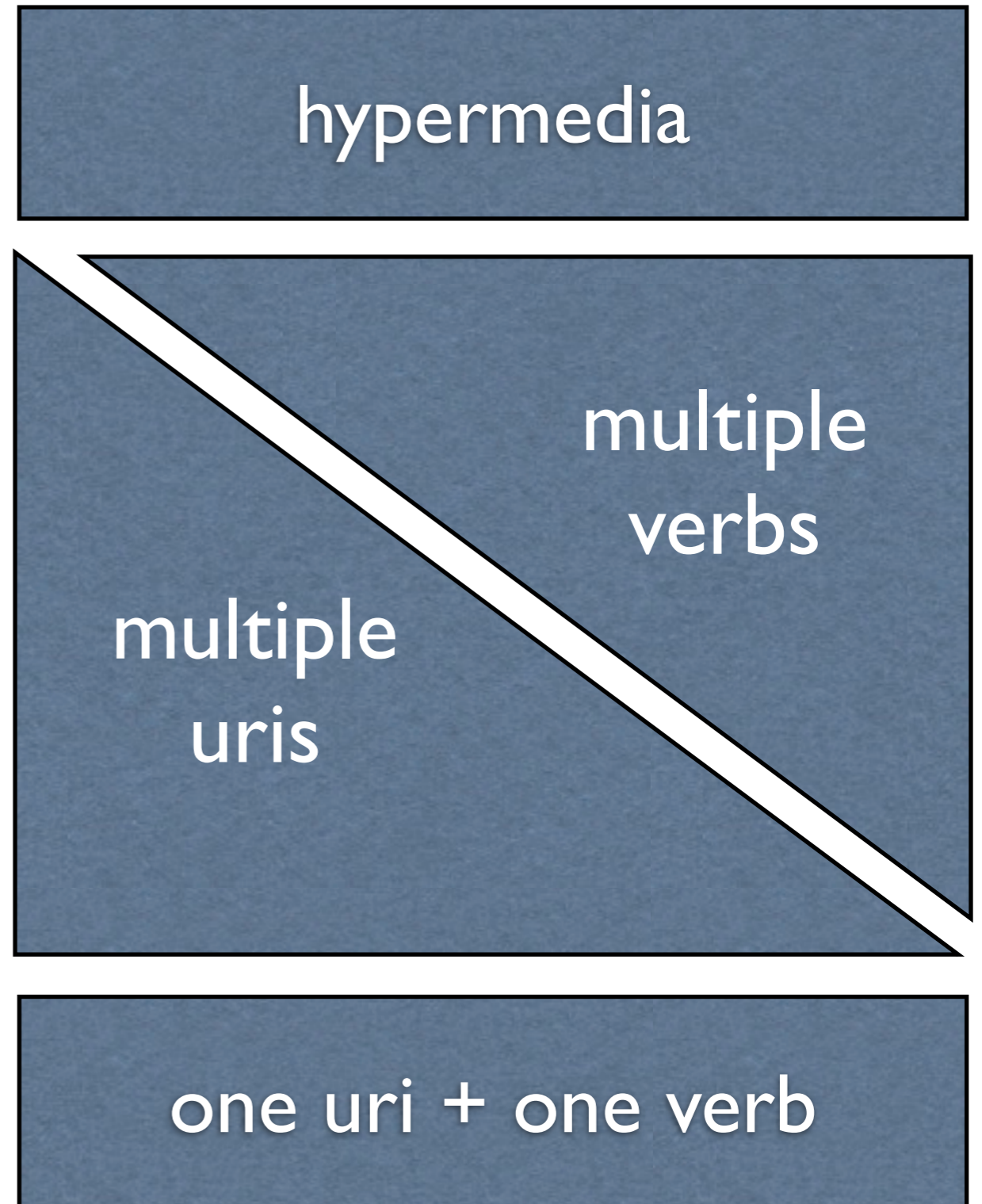


# Server Model

according to

Leonard Richardson, 2008

rest



# JAX-RS wishlist

VRaptor and Restfulie

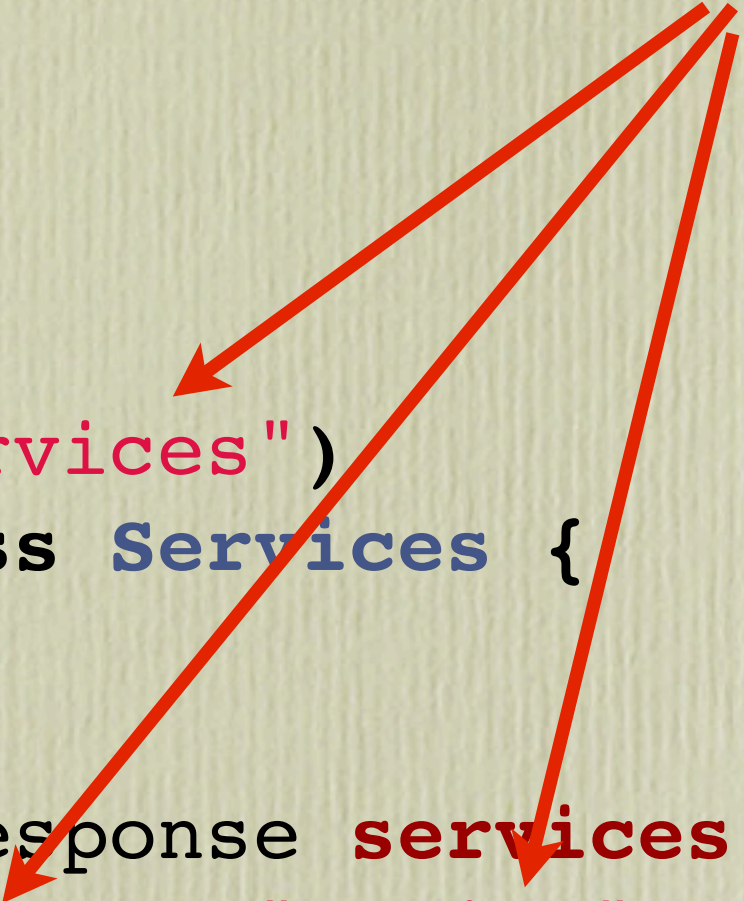
examples in java  
[vraptor.org/en](http://vraptor.org/en)

# I. conventions?

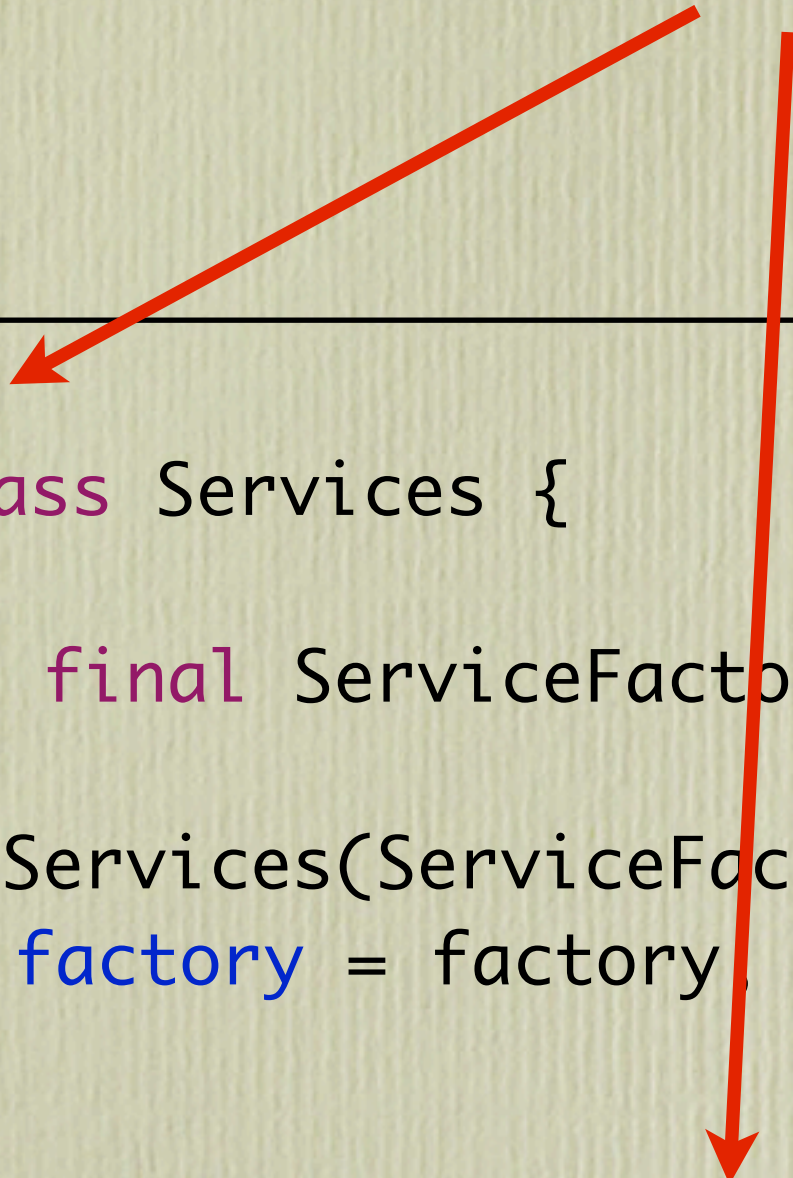
```
@Path("/services")
public class Services {

    @GET
    public Response services(
        @QueryParam("action") String action) {

        ServiceFactory factory = new ServiceFactory();
        Service service = factory.getServiceFor(action);
        return service.execute();
    }
}
```



# Convention over Configuration



```
@Resource
public class Services {

    private final ServiceFactory factory;

    public Services(ServiceFactory factory) {
        this.factory = factory;
    }

    public void services(String action) {
        factory.getServiceFor(action).execute();
    }

}
```

do you want to avoid  
copy+paste?



yes!

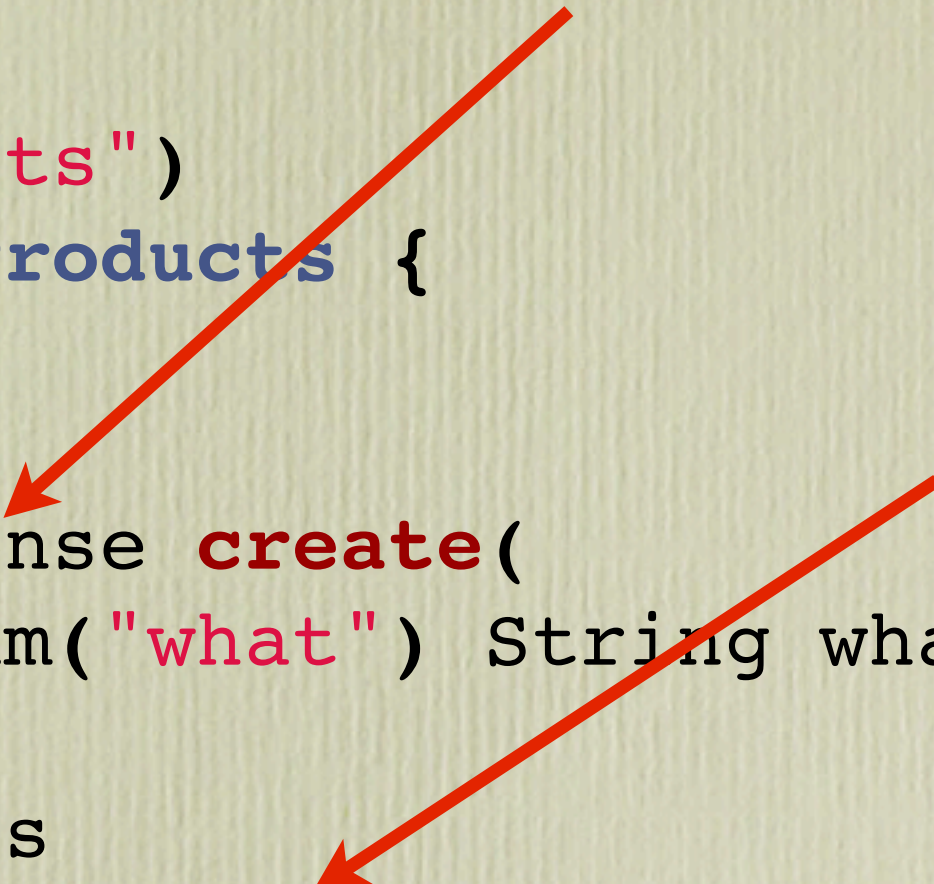
## 2. TDD: hard to test

```
@Path( "/products" )
public class Products {

    @GET
    public Response create(
        @QueryParam( "what" ) String what ) {

        // persists
        return Response.ok()
            .type( "application/xml" )
            .entity( "<product>...</product>" )
            .build();
    }
}
```

coupled to the implementation






# TDD: mock it

coupled to the interface  
couple--

```
@Resource  
public class Services {  
  
    private final Response response;  
  
    public Services(Response response) {  
        this.response = response;  
    }  
  
    public void services(String action) {  
        response.getServiceFor(action).execute();  
    }  
  
}
```



# 3. Content negotiation by hand



```
@Path("/softwares")
public class SoftwareResource {

    @POST @Consumes("application/xml")
    public Response install(Software software) {
        software = SoftwareRepository.register(software);
        long id = software.getId();
        URI uri = UriBuilder.fromPath("/softwares/" + id)
                            .build();

        software.install();
        return Response.created(uri).build();
    }
}
```

# Let us do it for you.

```
@Resource
public class SoftwareResource {

    @Post @Consumes
    public void install(Software software) {
        software = SoftwareRepository.register(software);
        response.created(software);
    }
}
```




# 4. URI coupling

writing the URI once

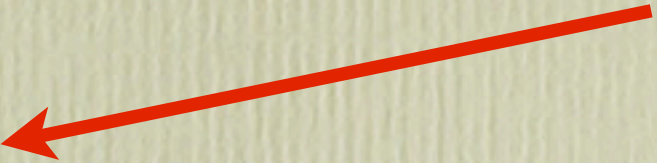
...

```
@GET @Path("/softwares/{id}")
public Response install(@QueryParam("id") Software
software) {
    // ...
}
```



# 4. URI coupling


writing the URI again  
several times



```
@Path("/softwares")
public class SoftwareResource {

    @POST @Consumes("application/xml")
    public Response install(Software software) {
        software = SoftwareRepository.register(software);
        long id = software.getId();
        URI uri = UriBuilder.fromPath("/softwares/" + id)
                            .build();

        software.install();
        return Response.created(uri).build();
    }
}
```



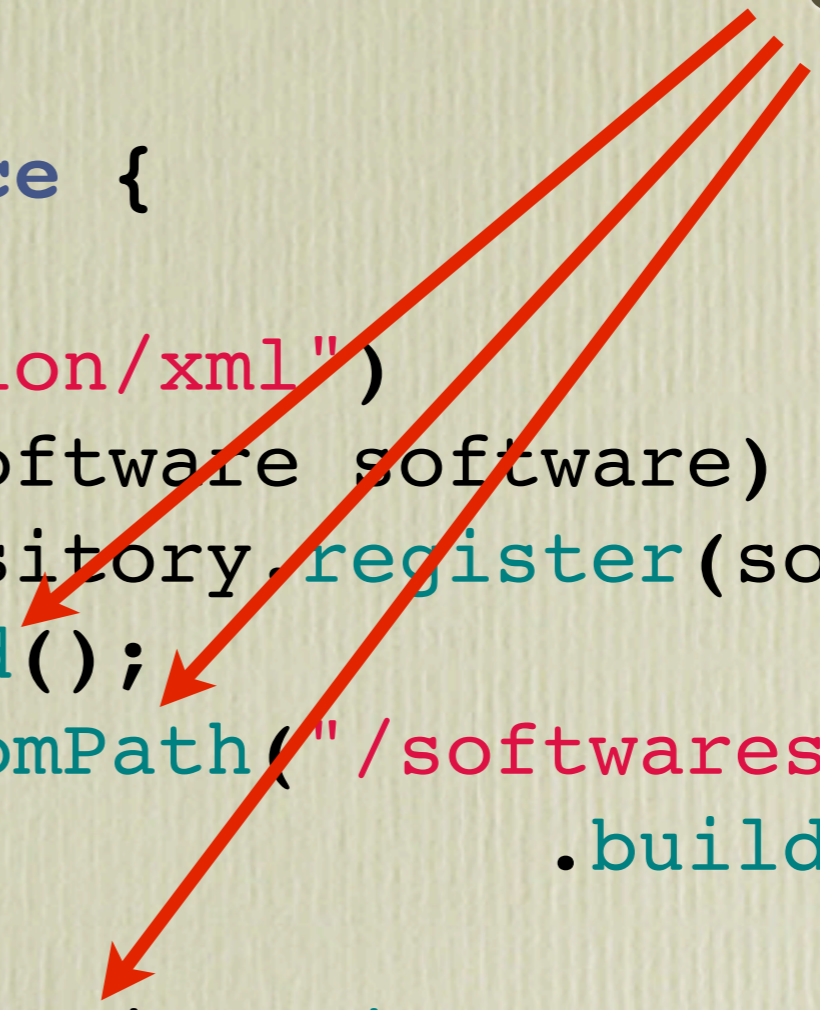
# 4. URI coupling

```
@Path("/softwares")
public class SoftwareResource {

    @POST @Consumes("application/xml")
    public Response install(Software software) {
        software = SoftwareRepository.register(software);
        long id = software.getId();
        URI uri = UriBuilder.fromPath("/softwares/" + id)
                            .build();

        software.install();
        return Response.created(uri).build();
    }
}
```

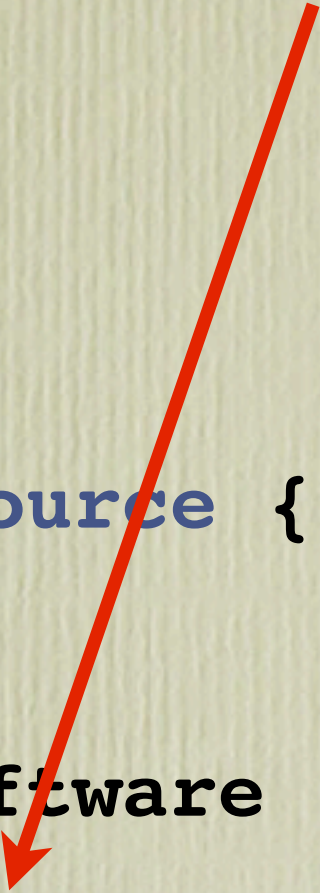
code



# ZERO uri replication

```
@Resource
public class SoftwareResource {

    @Post @Consumes
    public void install(Software software) {
        // ...
        response.use(SoftwareResource.class).show(software);
    }
}
```




# 5. Parameter list

```
@Path("/machines")
public class MachineResource {

    @Path("/{id}/softwares")
    public SoftwareResource softwares(@PathParam("id") Long id) {
        Machine machine = new MachineRepository().retrieve(id);
        if (machine == null) {
            throw new WebApplicationException(404);
        }
        // ...
    }

}
```





# Yes, we can do it.

```
@Resource
public class MachineResource {

    @Post (" {m.id}/softwares" )
    public SoftwareResource softwares(Machine m) {
        Machine machine = new MachineRepository().retrieve(m)
        // ...
    }
}
```



parameter converter chain of responsibility

example

[restfulie.caelumobjects.com](http://restfulie.caelumobjects.com)

```
Response response = client.at  
    ("http://localhost:9998/user/574").get();  
  
User user = response.getResource();  
System.out.println("user: " + user.getName());
```

## 6. Client internal DSLs

```
User user = response.getResource();
System.out.println("user: " + user.getName());

Link link = resource(user).getLink("machine");
response = link.follow().post(new Machine());
```

```
Link link = resource(user).getLink("machine");  
response = link.follow().post(new Machine());
```

```
double amount = resource(user).refresh().  
getAmountDue();
```

```
double amount = resource(user).refresh().  
getAmountDue();
```

```
link = resource(user).getLink("payment");  
Payment payment = new Payment(amount);  
response = link.follow().post(payment);
```

```
link = resource(user).getLink("payment");  
Payment payment = new Payment(amount);  
response = link.follow().post(payment);  
  
System.out.println("payment completed");
```

```
link = resource(user).getLink("payment");  
Payment payment = new Payment(amount);  
response = link.follow().post(payment);  
  
System.out.println("payment completed");
```

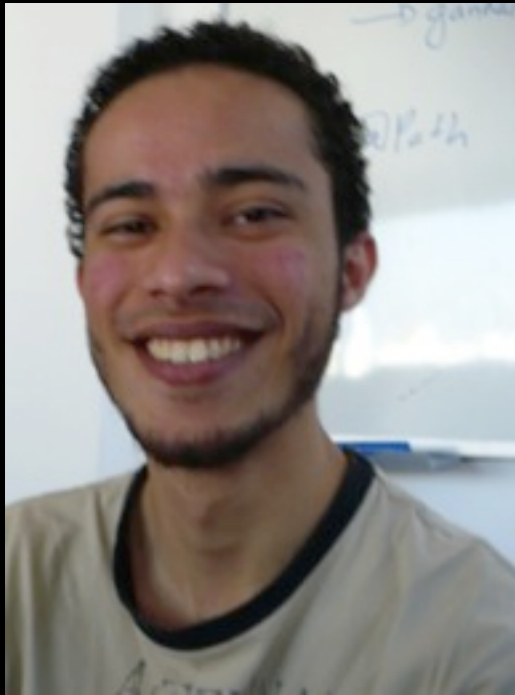


i was looking for a DSL  
and i did not know it!



now show me the ruby  
code!

# code samples



Lucas Cavalcanti  
(java server)  
@lucascs

Anderson Leite (ruby)  
@anderson\_leite



# Further reading

- Roy Fielding dissertation
- Rest in Practice, Jim Webber and others
- Restful Webservices Cookbook, Subbu Allamaraju
- Restful Web Services, Richardson and Ruby
- JAX-RS specification
- Infoq articles on REST
- Restfulie guide

# Interviews and Presentations

- Stefan Tilkov interview on InfoQ
  - <http://bit.ly/9RUXKL>
- Leonard Richardson's presentation at QCon
  - <http://bit.ly/dj2W66>
- Martin Fowler on REST
  - <http://bit.ly/bx61ci>

# Interviews and Presentations

- Ian Robinson and Jim Webber interview

- <http://bit.ly/aEuzj3>

- Jan Algermissen classification

- <http://bit.ly/cycFBF>

- Guilherme Silveira on REST clients

- <http://bit.ly/aHCgLv>

# code samples

- JAX-RS

- <http://jcp.org/en/jsr/detail?id=311>

- Rails

- <http://rubyonrails.org>

- Restfulie

- <http://restfulie.caelumobjects.com>

- VRaptor

- <http://vraptor.org/en>