

{ Desarrollo de Servicios REST con Java }

RESTful Java JAX-RS

Valladolid, 13 - 15 Septiembre 2017 (9 - 15 h)

Formador: Ezequiel Llarena Borges

HOW TO INSULT A DEVELOPER

gnek & pote



Desarrollo Servicios REST con Java

Algunos términos...

Web Service

SOAP

WSDL

HTTP

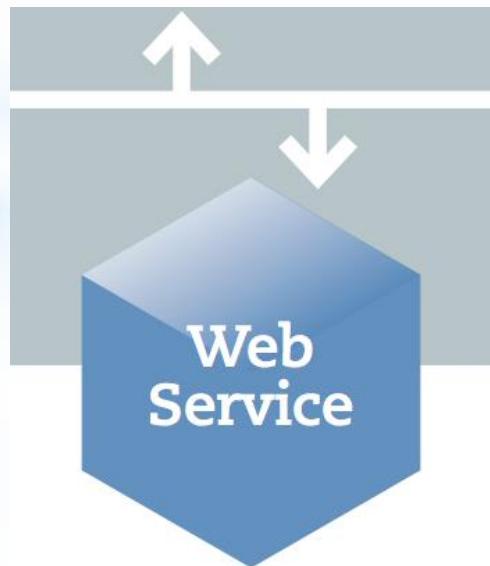
REST

URI

RESTful

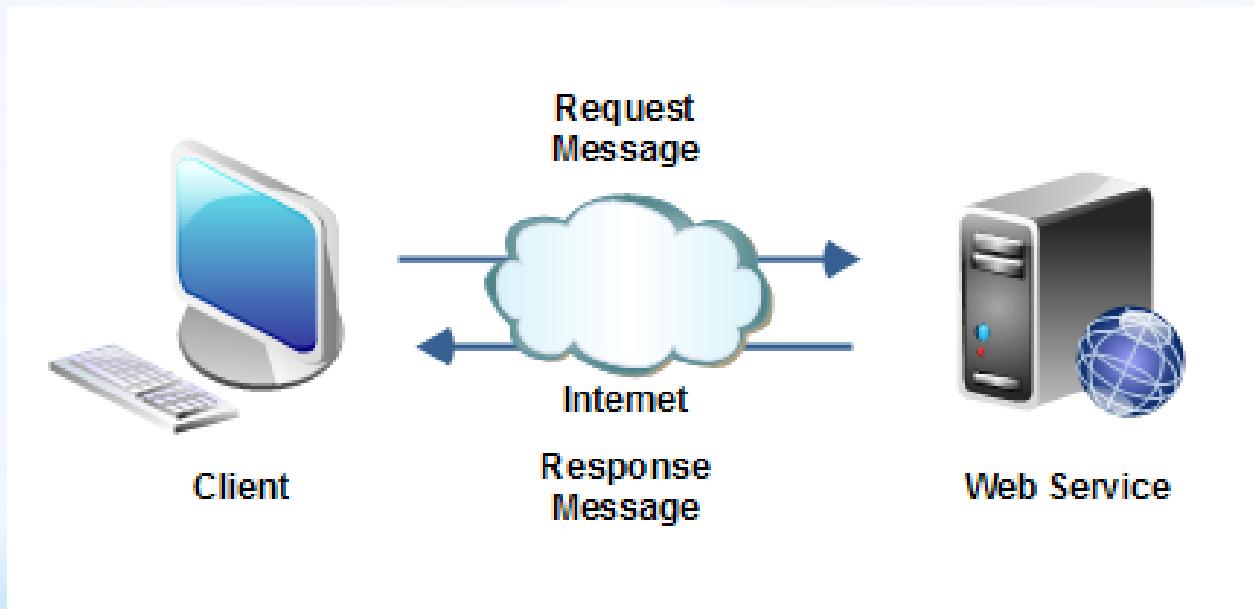
Desarrollo Servicios REST con Java

Servicio Web



Desarrollo Servicios REST con Java

Web Service



Desarrollo Servicios REST con Java

Web Service

- API online a la que se puede acceder programáticamente
- Código desplegado en diferentes máquinas
- Retorna datos en formato XML o JSON
- Facebook y Twitter publican web services consumidos por otros desarrolladores desde sus códigos
- Apps, Games, ...

Desarrollo Servicios REST con Java

Web Service

<http://www.twitter.com>



HTML



<http://api.twitter.com>



XML / JSON

JSON

```
{  
  "siblings": [  
    {"firstName": "Anna", "lastName": "Clayton"},  
    {"lastName": "Alex", "lastName": "Clayton"}  
  ]  
}
```

XML

```
<siblings>  
  < sibling>  
    < firstName>Anna</firstName>  
    < lastName>Clayton</lastName>  
  </ sibling>  
  < sibling>  
    < firstName>Alex</firstName>  
    < lastName>Clayton</lastName>  
  </ sibling>  
</ siblings>
```

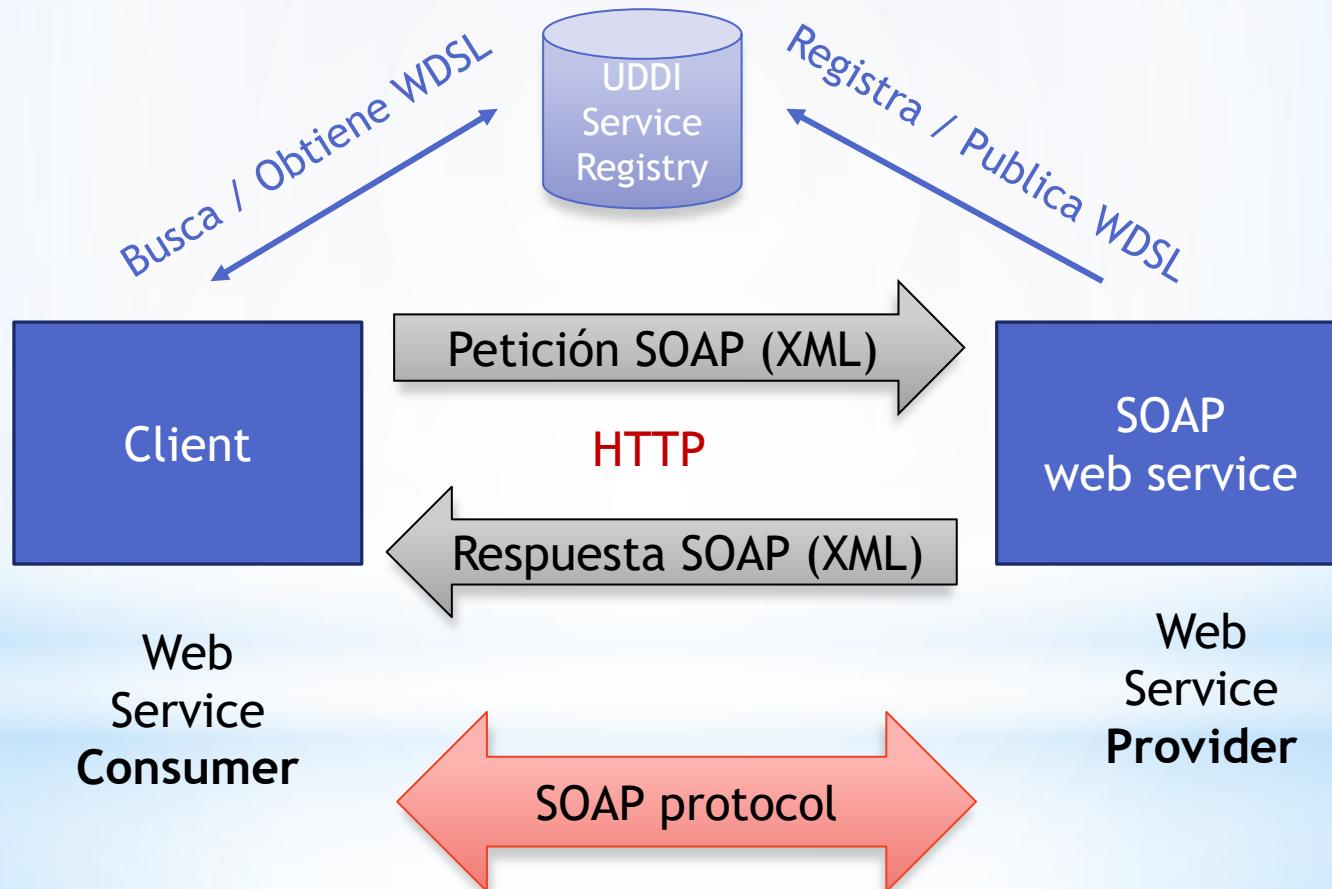
Desarrollo Servicios REST con Java

Estilos de Servicios Web

- **RPC (Remote Procedure Calls)**
 - Llamadas a procedimientos y funciones distribuidas (operación WSDL)
- **SOA (Service-Oriented Architecture)**
 - Comunicación vía mensajes (servicios orientados a mensajes)
- **REST (Representational State Transfer)**
 - Interacción con recursos con estado

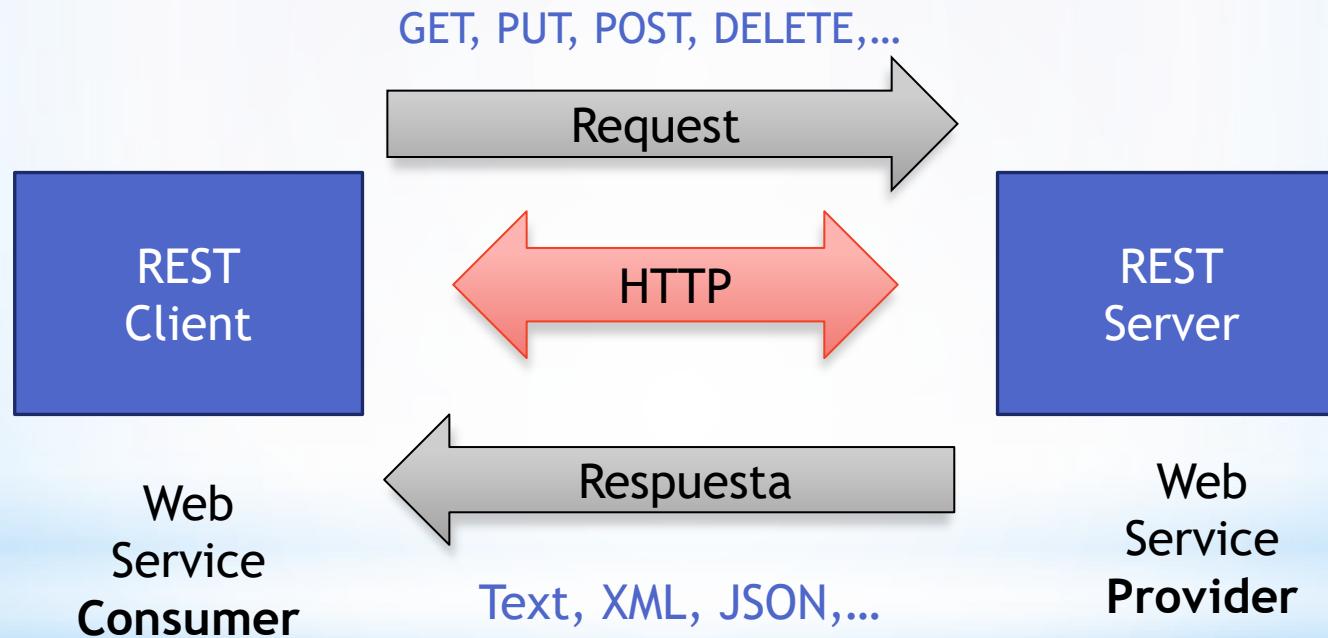
Desarrollo Servicios REST con Java

SOAP Web Service



Desarrollo Servicios REST con Java

REST Web Service



Desarrollo Servicios REST con Java

SOAP vs REST

#	SOAP Simple Object Access Protocol	REST Representational State Transfer
1	Protocolo de mensajes XML	Protocolo de estilo arquitectural
2	Usa WSDL en la comunicación entre el consumidor y el proveedor	Usa XML o JSON para enviar y recibir datos
3	Invoca a los servicios mediante llamadas a métodos RPC	Llamada a un servicio vía URL
4	Información que devuelve no legible para el humano	Resultado es legible por el humano (XML, JSON...)
5	Transferencia sobre HTTP y otros protocolos (SMTP, FTP, etc...)	Transferencia es sólo sobre HTTP
6	JavaScript permite invocar SOAP (implementación compleja)	Fácil de invocar desde JavaScript
7	El rendimiento no es tan bueno comparado a REST	Rendimiento mucho mejor que SOAP - menor consumo CPU, código más pulido, etc.

Desarrollo Servicios REST con Java

REST Web Services



La idea es... “ponérselo fácil a los consumidores”

Desarrollo Servicios REST con Java

1 - Introducción a Servicios REST

- REST y el renacimiento de HTTP
- Principios de arquitectura REST
 - ✓ El direccionamiento
 - ✓ Interfaces Constrained y Uniform*
 - ✓ HATEOAS

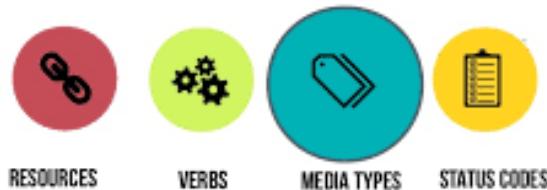
*REST usa un **conjunto reducido** y bien definido de **comandos** para gestionar los recursos.

Desarrollo Servicios REST con Java

REST (REpresentational State Transfer)

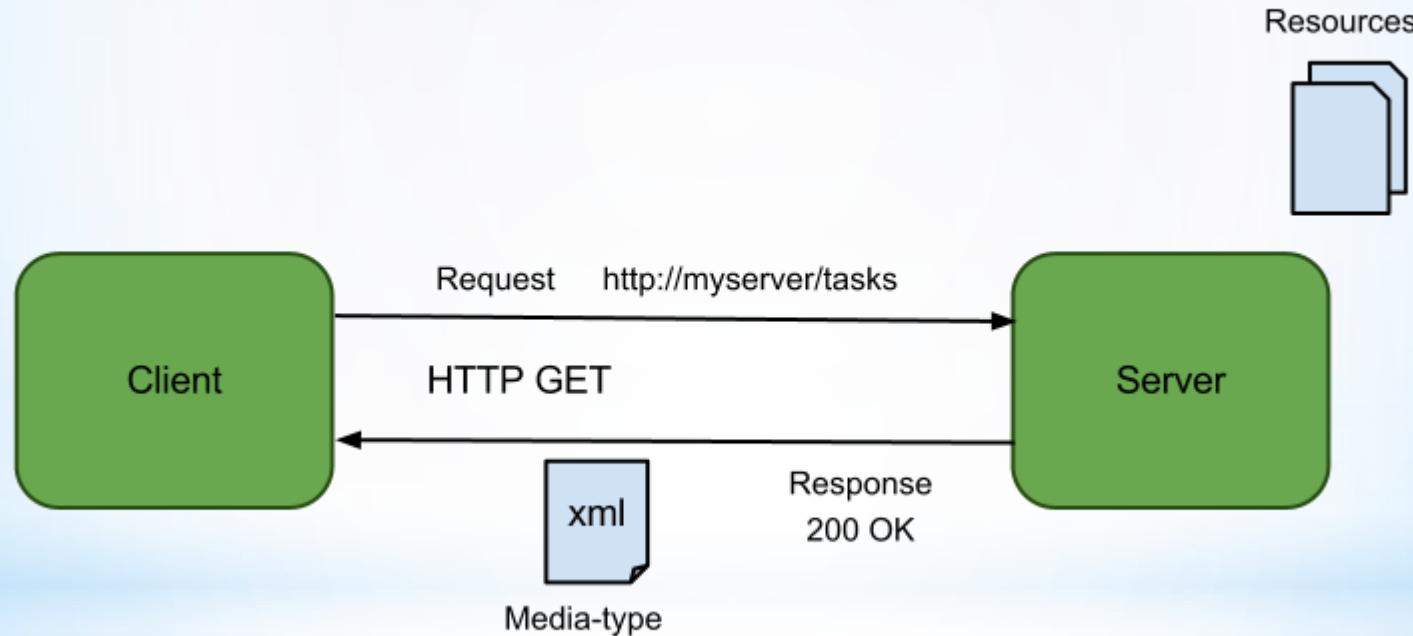
- Estilo de arquitectura de software para desarrollar servicios web
- Basado en **estándares web** y protocolo **HTTP**
- REST no es un estándar
- Todo es un **Recurso**
- Un servidor REST permite acceso a los recursos
- Recursos identificados mediante un ID global (**URIs**)
- Diferentes **representaciones** de los recursos (text, XML,JSON)

REST Style consists of ...



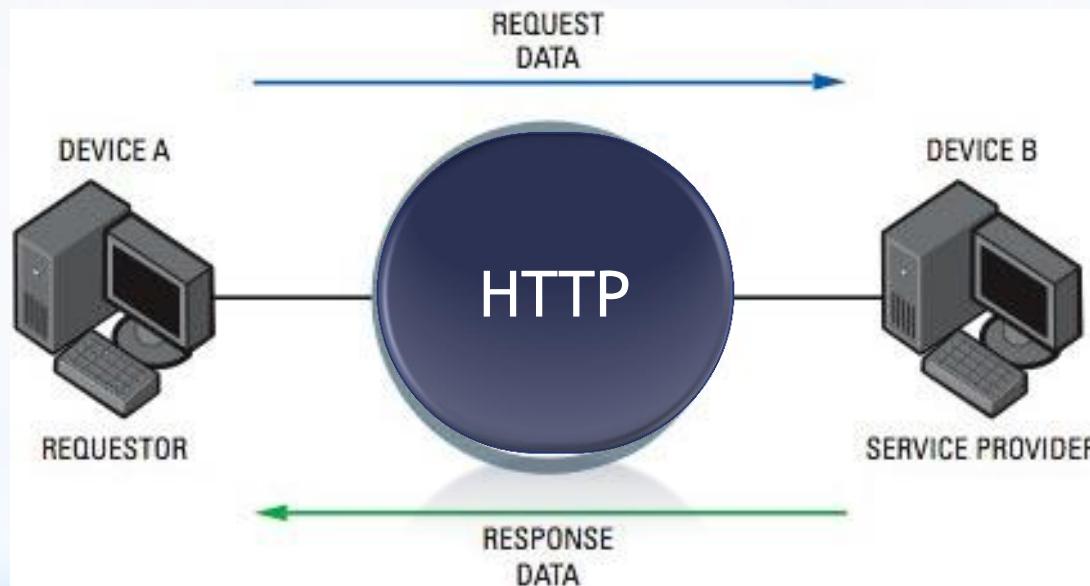
Desarrollo Servicios REST con Java

Funcionamiento de un Web Service REST



Desarrollo Servicios REST con Java

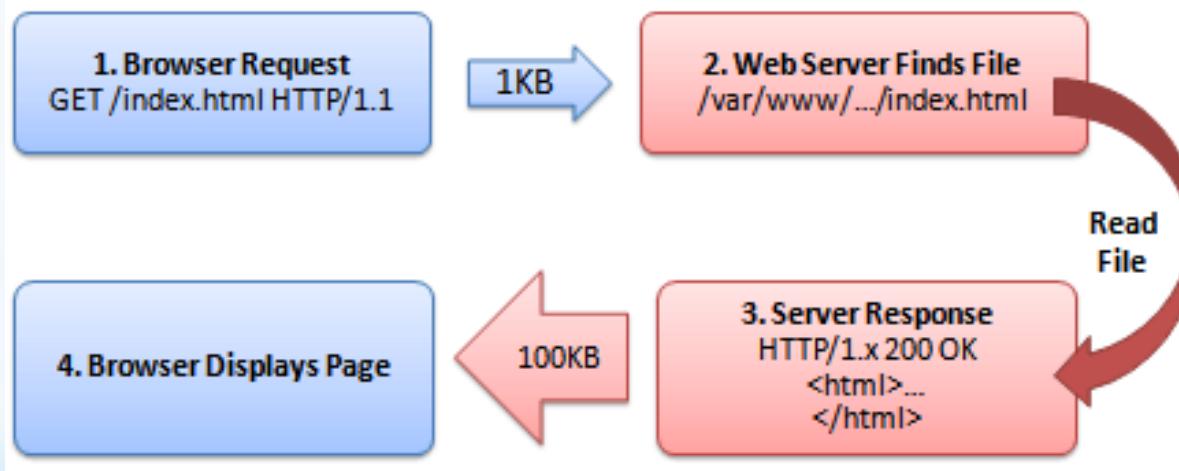
Arquitectura Cliente/Servidor



Desarrollo Servicios REST con Java

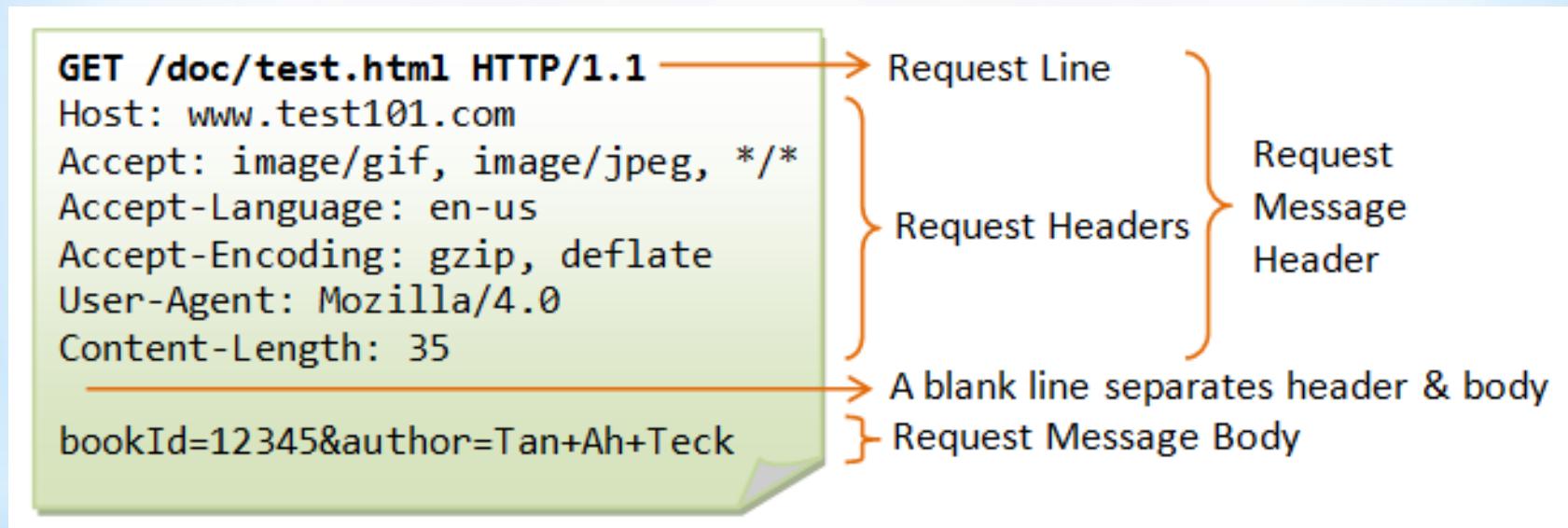
HTTP

HTTP Request and Response



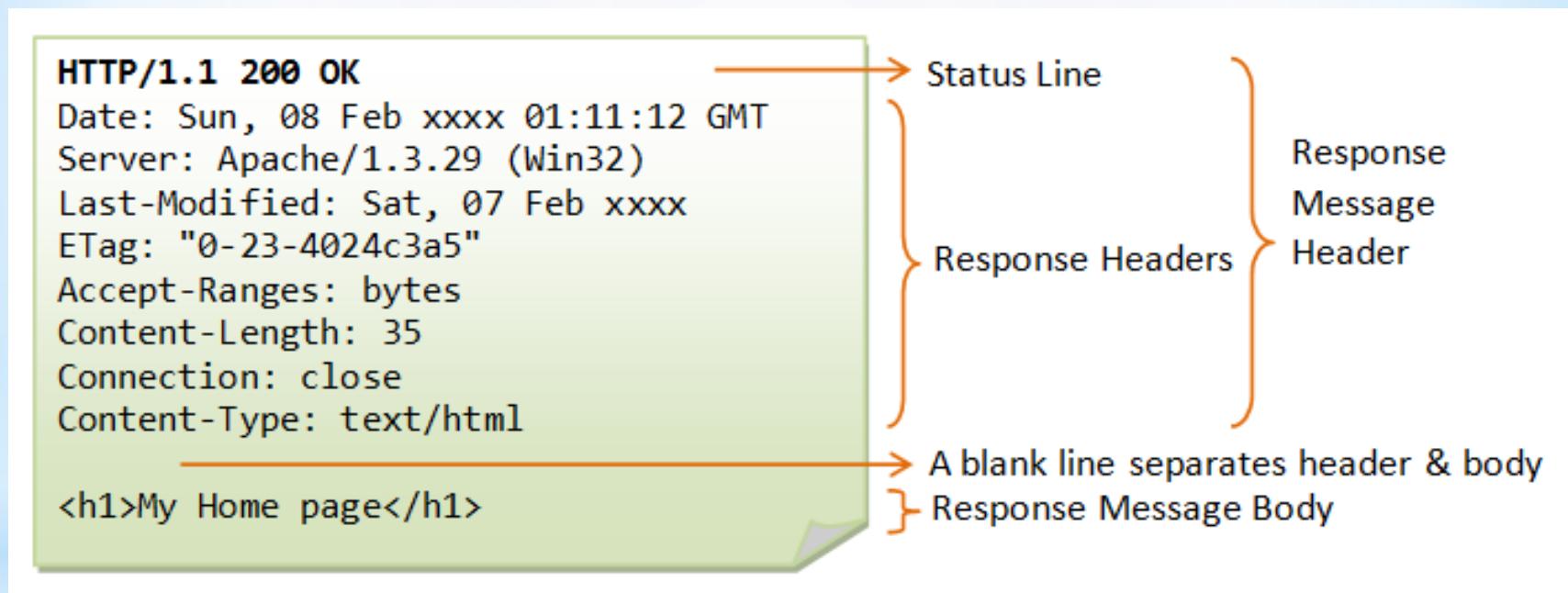
Desarrollo Servicios REST con Java

HTTP Request



Desarrollo Servicios REST con Java

HTTP Response



Desarrollo Servicios REST con Java

REST y HTTP

- Resource based URIs
- HTTP operations
 - ✓ GET
 - ✓ POST
 - ✓ PUT
 - ✓ DELETE
- HTTP status codes (*Metadata*)
 - ✓ 200 - Success
 - ✓ 500 - Server error
 - ✓ 404 - Not found
- Message headers
 - ✓ Content types: **text/xml, application/json**

Desarrollo Servicios REST con Java

Métodos HTTP

GET

- Obtiene un recurso
- Es seguro (no side-effects)
- Resultados “cacheables”
- Idempotente

POST

- Crea un recurso nuevo
- No es idempotente

PUT

- Actualiza un recurso existente
- Idempotente

DELETE

- Elimina un recurso
- Idempotente*

Desarrollo Servicios REST con Java

Idempotencia

- El cliente puede invocar repetidamente un método generándose siempre el mismo resultado.
- Las operaciones idempotentes producen siempre el mismo resultado en el servidor.

Idempotente	No Idempotente
<ul style="list-style-type: none">• GET• PUT• HEAD• OPTIONS	<ul style="list-style-type: none">• POST• DELETE

Desarrollo Servicios REST con Java

Resource locations

URI = Uniform Resource Identifier

Resource based URI

weatherapp.com/zipcodes/12345

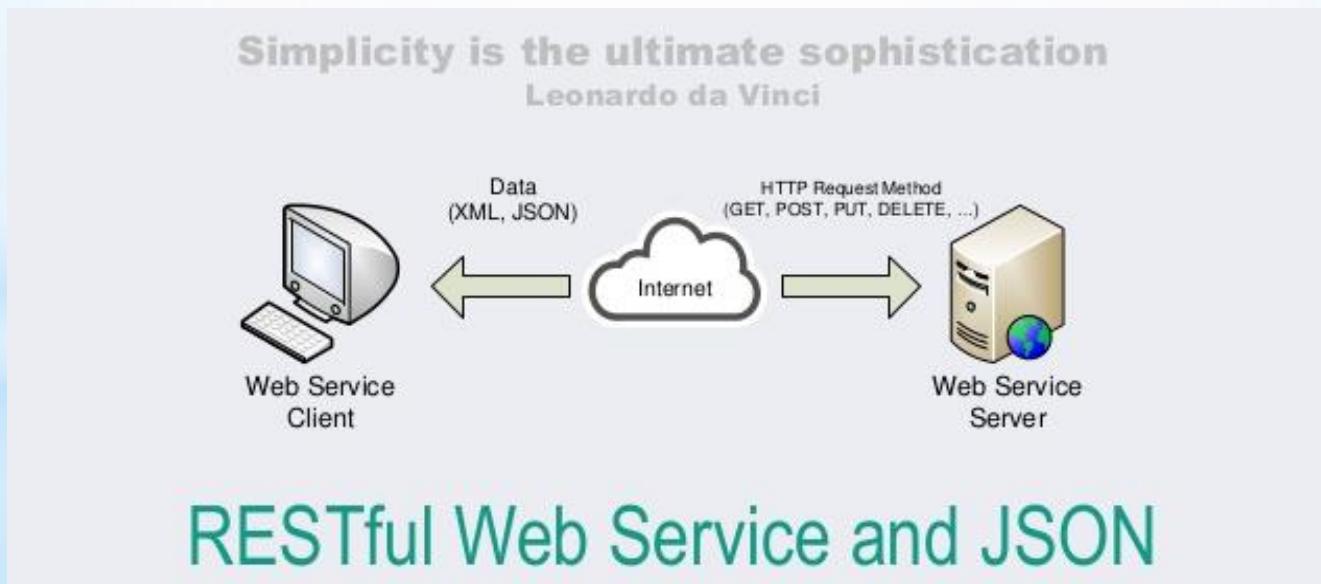
weatherapp.com/zipcodes/56789

weatherapp.com/countries/brazil

Desarrollo Servicios REST con Java

RESTful

RESTful
REST + Web Services = web services



Desarrollo Servicios REST con Java

RESTful

- Define URI base para los servicios
- Basado en métodos HTTP y concepto REST
 - ✓ GET
 - ✓ POST
 - ✓ PUT
 - ✓ DELETE
- Soporta MIME-types
 - ✓ XML
 - ✓ text
 - ✓ JSON
 - ✓ User-defined, ...

Desarrollo Servicios REST con Java

HATEOAS

Accessing messages

Client



message
Resource URI

Response

```
{  
    "id": "20",  
    "message": "Hello World!",  
    "date": "01Jan2015",  
    "author": "koushik",  
    "commentsUri": "api/messages/20/comments",  
    "likesUri": "api/messages/20/likes",  
    "sharesUri": "api/messages/20/shares",  
}
```

Desarrollo Servicios REST con Java

HATEOAS - the “rel” attribute

```
{  
    firstname: "M.",  
    lastName: "Ibrahim",  
    age: 35,  
    departement: "HR",  
    - links: [  
        - {  
            rel: "self",  
            href: "http://localhost:8080/person/2"  
        },  
        - {  
            rel: "account",  
            href: "http://localhost:8080/person/2/account"  
        },  
        - {  
            rel: "profile",  
            href: "http://localhost:8080/person/2/profile"  
        }  
    ]  
}
```

Desarrollo Servicios REST con Java

Richardson Maturity Model

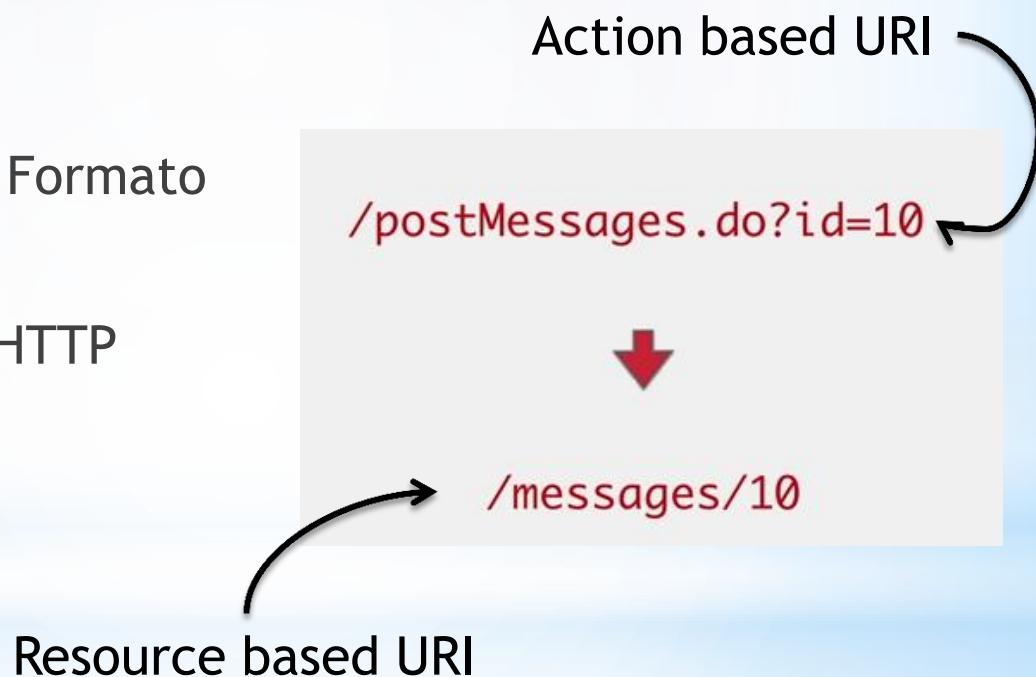
Is this API “fully RESTful”?

Level	Características	
0	<ul style="list-style-type: none">• One URI• Message request body contains all the details	Is not a RESTful API
1	<ul style="list-style-type: none">• Resource URI	Individual URIs for each resource
2	<ul style="list-style-type: none">• HTTP Methods	Use the right HTTP methods, status codes
3	<ul style="list-style-type: none">• HATEOAS	Responses have links that the clients can use

Desarrollo Servicios REST con Java

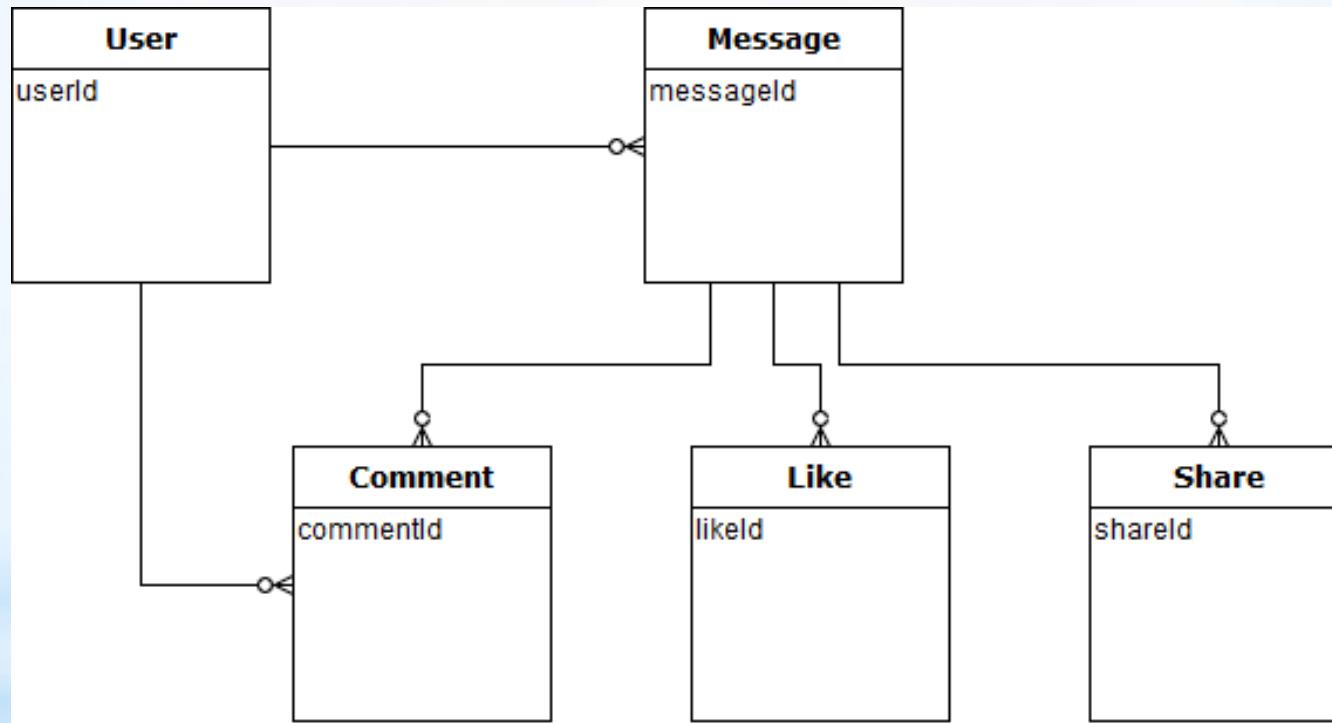
2 - Diseño de Servicios REST

- El modelo de objetos
- Formato de datos
- Leer y actualización de Formato
- Crear Formato
- Asignación de Métodos HTTP



Desarrollo Servicios REST con Java

2 - Diseño de Servicios REST (Ejemplo)



Desarrollo Servicios REST con Java

2.1 - Instance resource URI

Instance



/profiles/{profileName}

/messages/{messageId}

/messages/{messageId}/likes/{likeId}

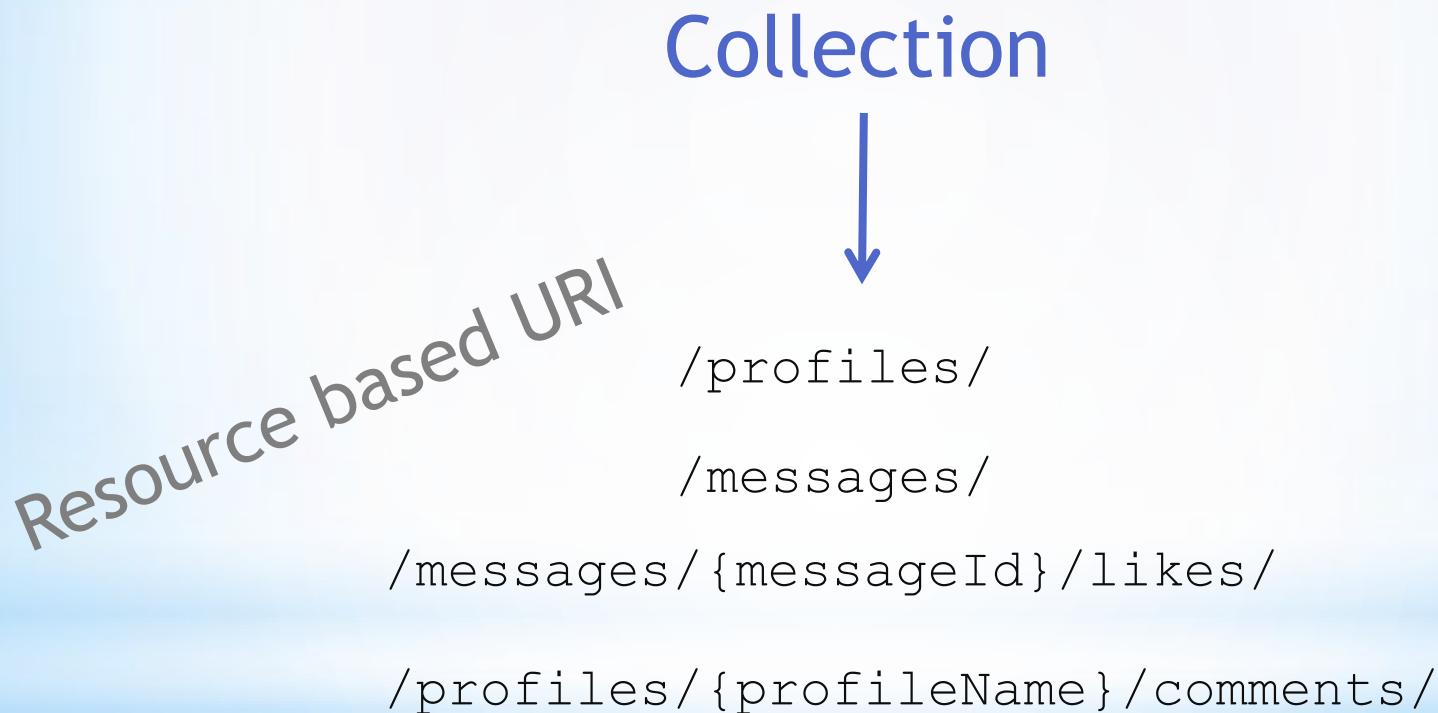
/coordinates/lats/{latVal}

/coordinates/lons/{lonVal}

Resource based URI

Desarrollo Servicios REST con Java

2.2 - Collection resource URI



Desarrollo Servicios REST con Java

2.3 - Filtering results

Query Parameters



/messages/?offset=5&limit=10



starting point page size

Custom Filters



/messages/?year=2017



Filtering Collection

Desarrollo Servicios REST con Java

2.4 - Formato de datos



```
<user>
  <id>1</id>
  <name>Me</name>
  <email>me@gmail.com</email>
</user>
```

content-type: **text/xml**



```
{
  "id" : 1,
  "name" : "Me",
  "email" : "me@gmail.com"
}
```

content-type: **application/json**

Desarrollo Servicios REST con Java

2.4 - Operaciones HTTP

/getProducts.do?id=10 



/products/10 

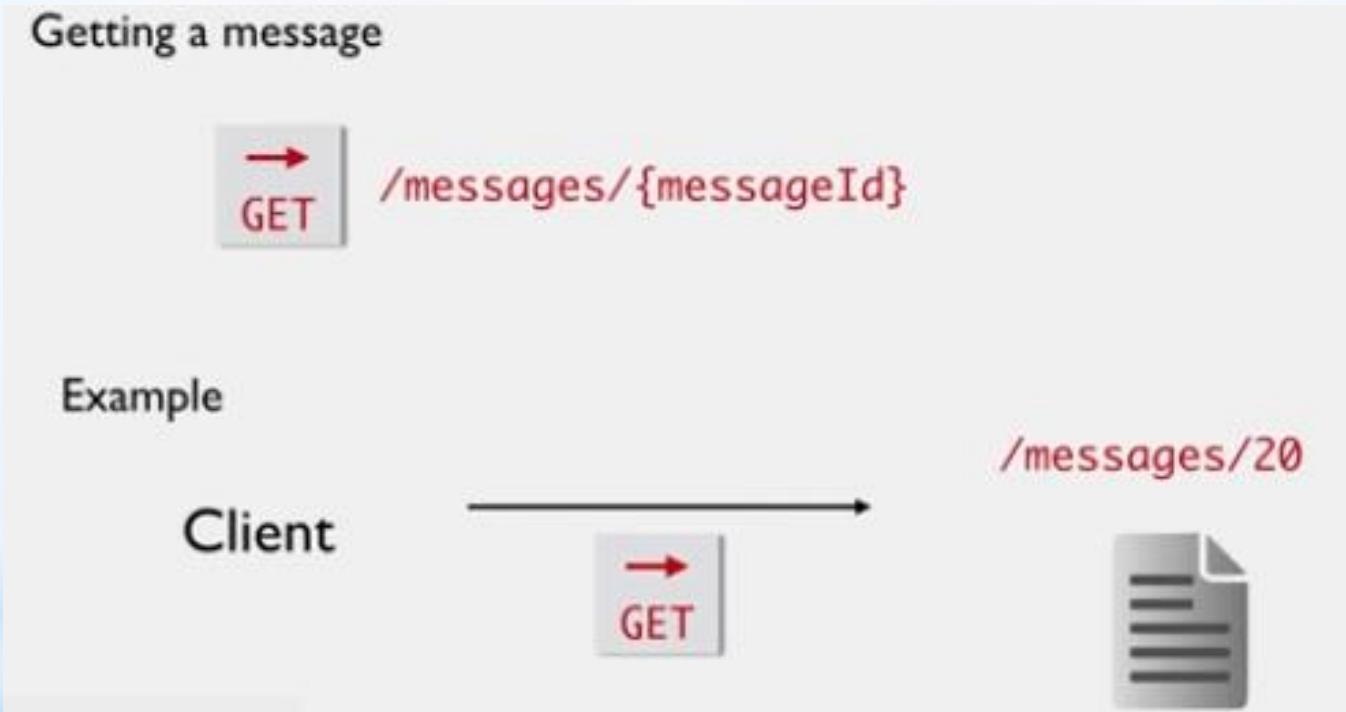
/deleteOrder.do?id=10 



/orders/10 

Desarrollo Servicios REST con Java

2.4 - Operaciones HTTP GET



Desarrollo Servicios REST con Java

2.4 - Operaciones HTTP PUT

Updating a message



/messages/{messageId}

Example

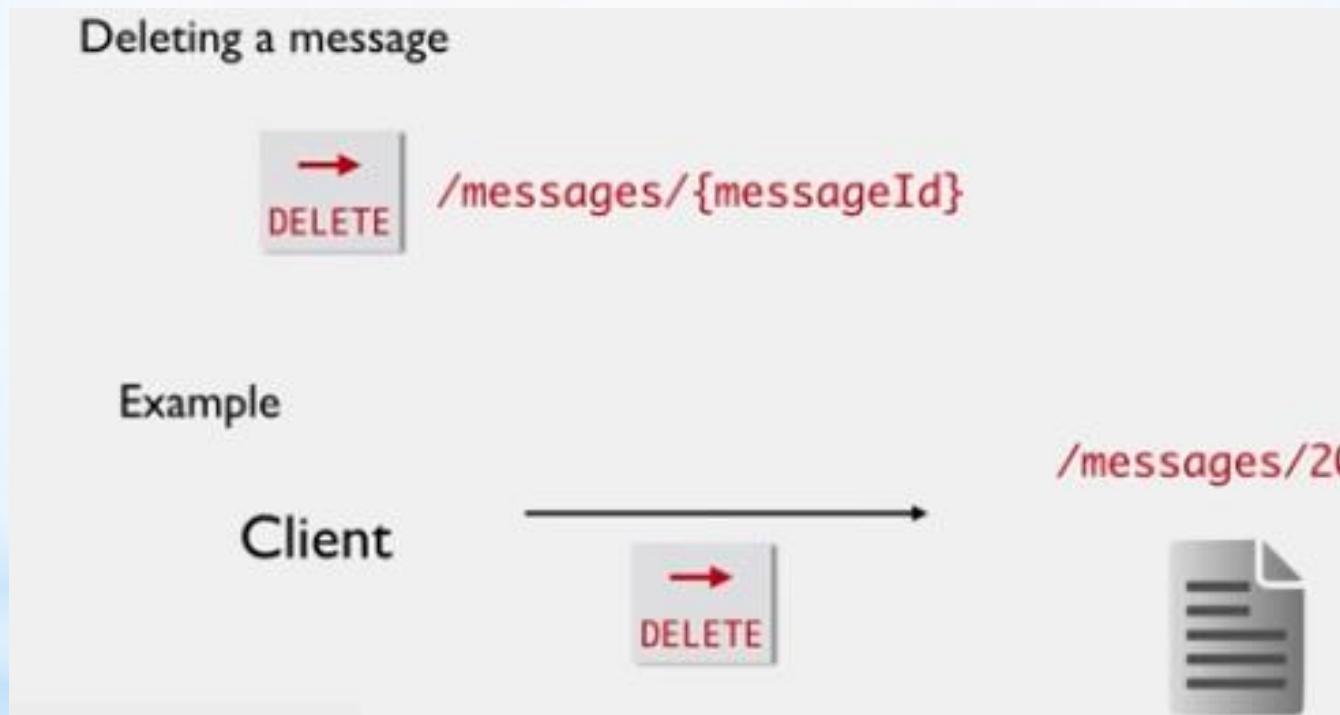
Client



/messages/20

Desarrollo Servicios REST con Java

2.4 - Operaciones HTTP DELETE



Desarrollo Servicios REST con Java

2.4 - Operaciones HTTP POST

Creating a new message



Example



Desarrollo Servicios REST con Java

2.4 - Operaciones HTTP

Collection URI scenarios

→
POST

/messages/10/comments

creates a new comment for message 10

→
PUT

/messages/20/comments

replaces *all* comments for message 20 with a new list

Desarrollo Servicios REST con Java

3 - JAX-RS Servicio

- El desarrollo de un servicio JAX-RS REST
- La clase Data
- JAX-RS Servicio
- JAX-RS e Interfaces Java
- Herencia
- Implementación de nuestro servicio
- Escritura de un cliente

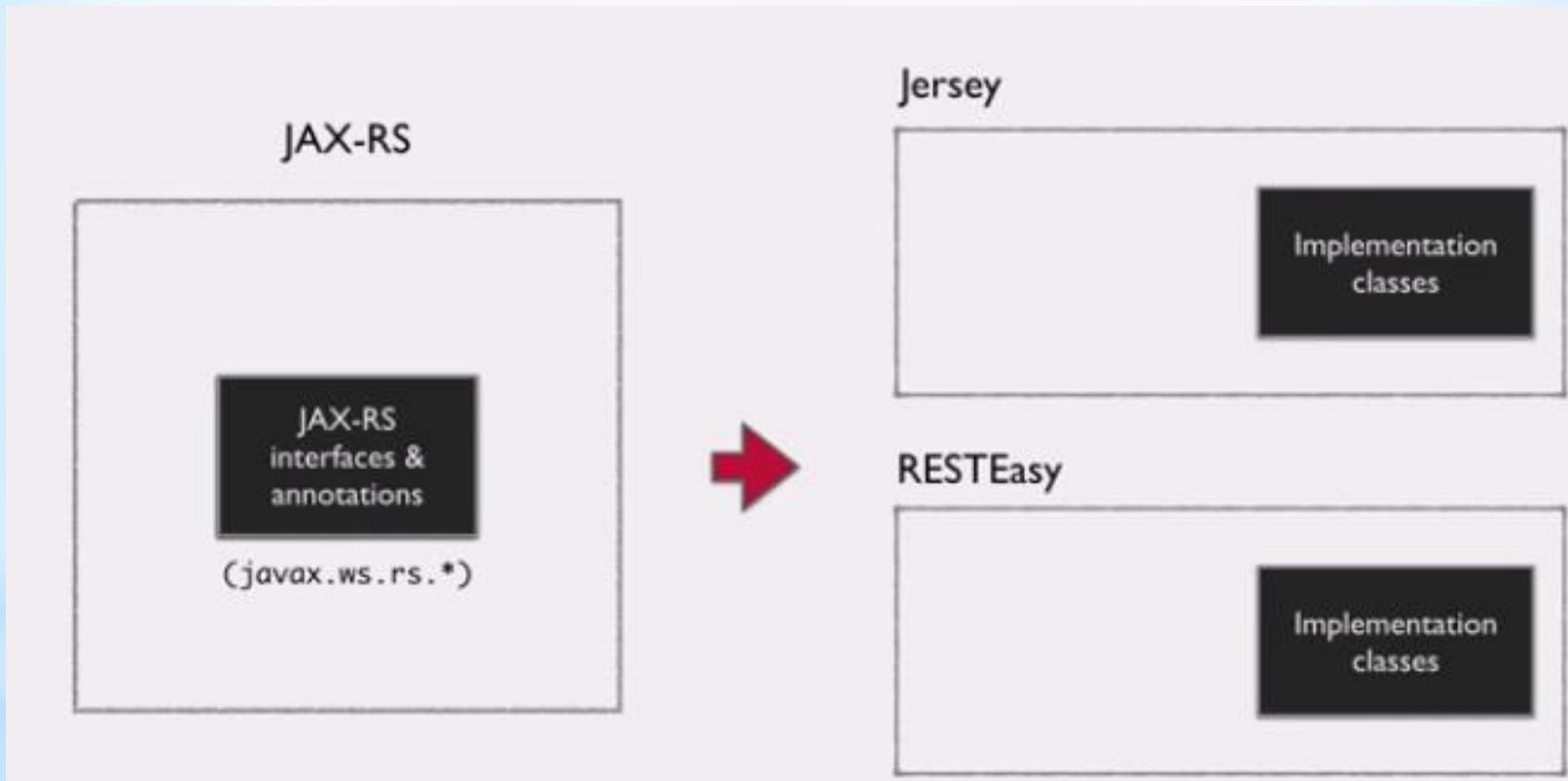
Desarrollo Servicios REST con Java

3 - JAX-RS

- Java define soporte REST vía JSR 311
- JAX-RS: *The Java API for RESTful Web Services*
- API de Java para servicios web REST forma parte JEE6
- JAX-RS usa anotaciones para aplicar REST en clase de Java

Desarrollo Servicios REST con Java

3 - JAX-RS Servicio



Desarrollo Servicios REST con Java

Jersey

- Librería para implementar RESTful webservices en un contenedor de Java Servlets
- Implementación de referencia para la especificación JSR 311
- Provee implementación de un servlet que busca clases predefinidas para identificar recursos RESTful
- Basada en anotaciones en clases y métodos
- Incluye una librería cliente para comunicarse con RESTful webservice

Framework que simplifica el uso de JAX-RS extendiéndola y dándole más funcionalidades para los desarrolladores

Desarrollo Servicios REST con Java

4 - Método HTTP y URI Matching (I)

- **@GET, @PUT, @POST, @DELETE** asocian métodos Java a una operación HTTP específica
- **@Path** asocian un patrón URI a un método Java

Desarrollo Servicios REST con Java

4 - Método HTTP y URI Matching (II)

- **@Path** permite definir patrones de “URI matching” complejos que puedan ser mapeados a un método Java (expresiones regulares)
- **@HttpMethod** permite crear nuevas operaciones HTTP a parte de GET, POST, DELETE, HEAD, y PUT.

Desarrollo Servicios REST con Java

5 - Inyección JAX-RS

Anotación	Descripción
@Path	url donde responderá el webservice
@Get	Método que responderá a peticiones Get
@Post	Método que responderá a peticiones Post
@Put	Método que responderá a peticiones Put
@Delete	Método que responderá a peticiones Delete
@Head	Método que responderá a peticiones Head
@PathParam	Representa un parámetro extraído de el request de la url

Desarrollo Servicios REST con Java

6 - JAX-RS Controladores de Contenido

- JAX-RS permite **convertir** automáticamente **objetos** de Java a un tipo de **formato** específico como parte de la respuesta HTTP.
- Permite leer el cuerpo de peticiones HTTP y **crear objetos** de Java que **representen la petición**.
- JAX-RS integra un número de controladores, pero solo podemos escribir *marshallers* y *unmarshallers* de usuario.

Desarrollo Servicios REST con Java

7 - Respuestas del Servidor y Control de Excepciones

- JAX-RS incluye **códigos de respuesta** por defecto *success* y *error*.
- Para respuestas más complejas, los métodos de recursos JAX-RS pueden retornar objetos **javax.ws.rs.core.Response**.
- JAX-RS permite lanzar Excepciones **javax.ws.rs.WebApplicationException** o derivarlas al contenedor del servlet de la capa subyacente.
- O crear objeto **ExceptionMapper** para mapear una excepción concreta como respuesta HTTP.

Desarrollo Servicios REST con Java

8 - API cliente JAX-RS

- Primera versión sin API cliente
- Conjunto clases *java.net.URL* para invocar RESTful ws
- Apache HTTP Client (not JAX-RS aware)*
- API cliente propietarias
- HTTP client API (JAX-RS 2.0)

Desarrollo Servicios REST con Java

RESTful Java Clients

- `java.net.URL`
- Apache HttpClient
- RESTEasy client
- Jersey client
- Retrofit, Postman...

Desarrollo Servicios REST con Java

9 - Negociación de contenido HTTP

- El Cliente puede pedirle al servidor el **formato** de las respuestas.
- El Cliente puede **negociar** el tipo de contenido del cuerpo del mensaje HTTP:
 - *Encoding*
 - *Language*
- Este protocolo se llama **HTTP Content Negotiation, conneg.**

Desarrollo Servicios REST con Java

Ejemplo: REST API GitHub

<https://developer.github.com/v3/>

Desarrollo Servicios REST con Java

Bibliografía

O'REILLY®



RESTful Java with JAX-RS 2.0

DESIGNING AND DEVELOPING DISTRIBUTED WEB SERVICES

Bill Burke