REST

Norberto Fernández
Departamento de Ingeniería Telemática
http://www.it.uc3m.es/berto/

Motivación

- Complejidad de la pila de especificaciones de Servicios Web
- Los Servicios Web no aprovechan al máximo las características que hacen a la Web exitosa:
  - Web → acceso a recursos
    - SW = acceso a operaciones
  - Web → hiperenlaces
    - SW = difícil hiperenlazado de servicios
  - Servicios Web usan HTTP en transporte
    - Pero no aprovechan algunas características (Ej.: cachés)

Introducción

- REST = *REpresentational State Transfer*
- Tesis doctoral de Roy Fielding (2000)
  - Uno de los editores del RFC de HTTP
- Conjunto de principios de diseño para arquitecturas software distribuidas
  - *RESTful architectures*

Principios REST

- Acceso a recursos
- Uso de hipermedia para conectar recursos
- Operaciones estándar de acceso a recursos
- Comunicación sin estado

Acceso a recursos (I)

- Servicios Web → RPC
  - Se accede a un conjunto de **operaciones**
    - Interfaz de servicio definida en WSDL
- REST → Cliente accede a **recursos** ofrecidos por un servidor no a operaciones
  - Un recurso es cualquier cosa de interés para la aplicación: entidades, procedimientos, colecciones …
  - Recursos manipulables mediante el intercambio entre cliente y servidor de "representaciones" de esos recursos (usualmente mediante HTTP)

Acceso a recursos (II)

- ROA: *Resource Oriented Architecture*
  - En contraposición a SOA de Servicios Web
- Los recursos tienen un identificador único
  - Uso de estándares: URIs
    - Facilita su uso por terceros (mejor que esquema de nombramiento propio)
    - Se pueden enlazar, compartir por correo, añadir a marcadores, etc.
    - Permite la indexación y búsqueda de recursos por buscadores
    - Permite el uso de cachés (mejora escalabilidad)

Acceso a recursos (III)

- Ejemplo:

http://ejemplo.com/usuarios/1234
http://ejemplo.com/usuarios?nombre=Ana

- Una misma aplicación puede exponer multitud de recursos (con sus diferentes URIs)
  - Usualmente en servicios Web sólo una: *Access point*
    - La operación a invocar y sus parámetros en cuerpo HTTP

Acceso a recursos (IV)

- Los recursos pueden tener representaciones alternativas
  - Permitir a los clientes y servidores negociar el tipo de representación a usar en comunicación
    - Flexibilidad a la hora de exigir soporte de formatos
  - La negociación se hace mediante cabeceras HTTP
    - Accept + tipo MIME (Ej.: Accept: text/xml)
    - Por ejemplo:
      - Navegador → HTML / aplicación Java → XML o JSON

Acceso a recursos (V)

- Ejemplo:

```
GET /customers/1234 HTTP/1.1
Host: example.com
Accept: text/xml
```

```
<customer id="1234">
  <name>I am a user</name>
  <email>user@server.com</email>
</customer>
```

```
GET /customers/1234 HTTP/1.1
Host: example.com
Accept: text/x-vcard
```

```
BEGIN:VCARD
VERSION:2.1
N: I am a user
EMAIL: user@server.com
END:VCARD
```


Uso de Hipermedia (I)

- HATEOAS (*Hypermedia as the engine of application state*)
- Representaciones de recursos pueden enlazar a otros recursos
  - De la misma aplicación o de otras
- Utilidades:
  - Representar listas de recursos
  - Indicar al cliente futuros estados de la aplicación a los que puede ir (siguiendo el enlace)

Uso de Hipermedia (II)

- Ejemplo:

```
GET /orders/1234 HTTP/1.1
Host: example.com
Accept: text/xml
```

```
<order self='http://example.com/orders/1234' >
  <amount>23</amount>
  <product ref = 'http://example.com/products/4554' />
  <customer ref = 'http://example.com/customers/1234' />
</order>
```

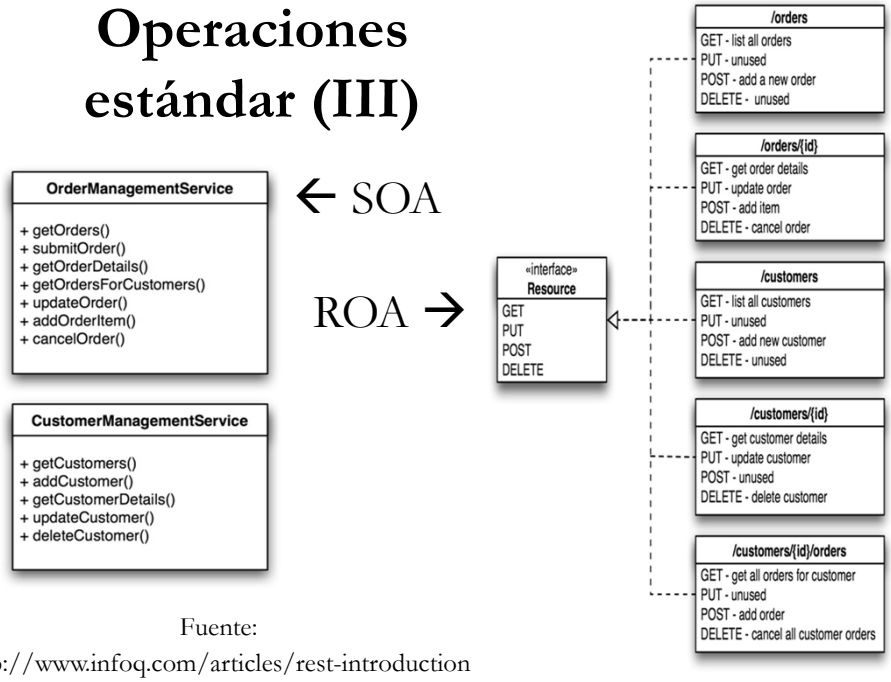

Operaciones estándar (I)

- Operaciones de acceso estándar
  - Independientes del tipo de recurso
- Son los métodos estándar de HTTP:
  - GET → Lectura del recurso
  - POST → Creación de un nuevo recurso (en una URL a decidir por el servidor)
  - PUT → Actualización de un nuevo recurso (o creación en una URL dada si no existe)
  - DELETE → Eliminación de un recurso

Operaciones estándar (II)

- Ventajas:
  - Cualquier aplicación que "entienda" HTTP puede acceder al servicio
  - No es necesario definir las operaciones en WSDL
    - Aunque sí saber qué recursos ofrece la aplicación
    - … y en la práctica hay formatos para ello como WADL (*Web Application Description Language*)
      - Actualmente sin demasiado soporte

Operaciones estándar (III)



Fuente:
http://www.infoq.com/articles/rest-introduction

Comunicación sin estado

- Cada solicitud /respuesta HTTP debe contener toda la información que sea necesaria
- Ventajas:
  - No mantener sesiones en servidor
    - Mejora la escalabilidad del servidor
  - Desacoplar el servidor del cliente
    - Cualquier servidor vale para atender la petición del cliente
      - No hay ninguno "especial" que tenga la información de sesión
      - Escalabilidad mediante el uso de granjas de servidores

Críticas al modelo

- Necesidad de modelar todas las operaciones mediante los métodos HTTP
- ¿Cómo resolver?
  - Transacciones atómicas distribuidas
  - Envío fiable de mensajes
    - Pero GET, PUT y DELETE son idempotentes
      - Llamar varias veces mientras no sepamos si se han ejecutado o no
  - Seguridad
    - Pero se puede usar HTTPS

Referencias

- RESTful Web Services
  - Leonard Richardson & Sam Ruby
  - Editorial O'Reilly, 2007
  - ISBN: 978-0596529260
- A Brief Introduction to REST
  - http://www.infoq.com/articles/rest-introduction
  - Consultado el 30/Nov/2011