Curso de Arquitectura de Software

RESTful Web Services

Estilo REST

- REST - REpresentational State Transfer (Roy Fielding, 2000)
- SOA sin Web Services, ni SOAP ni RPC
- Arquitectura con modelo de datos (recursos, URIs y representaciones XML)
- Composición de diversos estilos: repositorio replicado, *cache*, cliente-servidor, sistema en capas, sistema sin estado, máquina virtual, código a demanda e interfaz uniforme

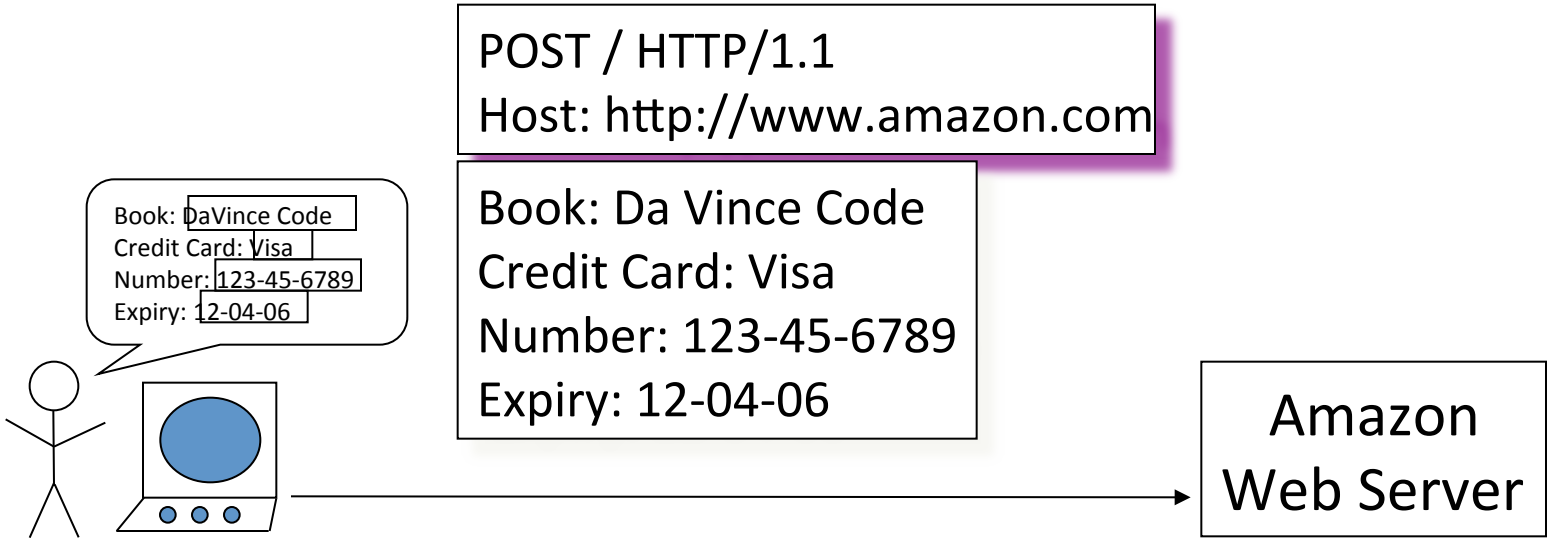
Modelo REST

- Una aplicación REST transfiere representaciones entre componentes usando conectores
- Componentes: incluyen agentes de usuario (Mozilla, cURL) y servidores de origen (Apache, IIS)
- Los componentes de REST obedecen estas restricciones:
  - las interaciones son *stateless*
  - los recursos se identifican mediante URIs
  - no hay tal cosa como servicios ni objetos, sólo recursos
  - no se permite el paso de *cookies* y se propone el reemplazo de MIME (Multipurpose Internet Mail Extensions) por su discrepancia arquitectónica con la semántica de HTTP
  - hypermedia es la máquina de estado de la aplicación
  - no hay necesidad de *toolkits*: sólo URIs y XML

Interacción Web básica usando HTTP GET

- El usuario digita en su browser: http://www.amazon.com
- El browser crea un HTTP header (no payload)
    - The HTTP header identifica:
        - La acción deseada: GET ("get me resource")
        - La máquina destino (target machine) (www.amazon.com)

Interacción Web básica usando HTTP POST



- El usuario llena la forma
- El browser crea un HTTP header con payload

    - El HTTP header identifica:
        - La acción deseada: POST ("here's some update info")
        - La máquina destino (amazon.com)
    - El payload contiene:
        - Los datos a ser publicados “POSTed”

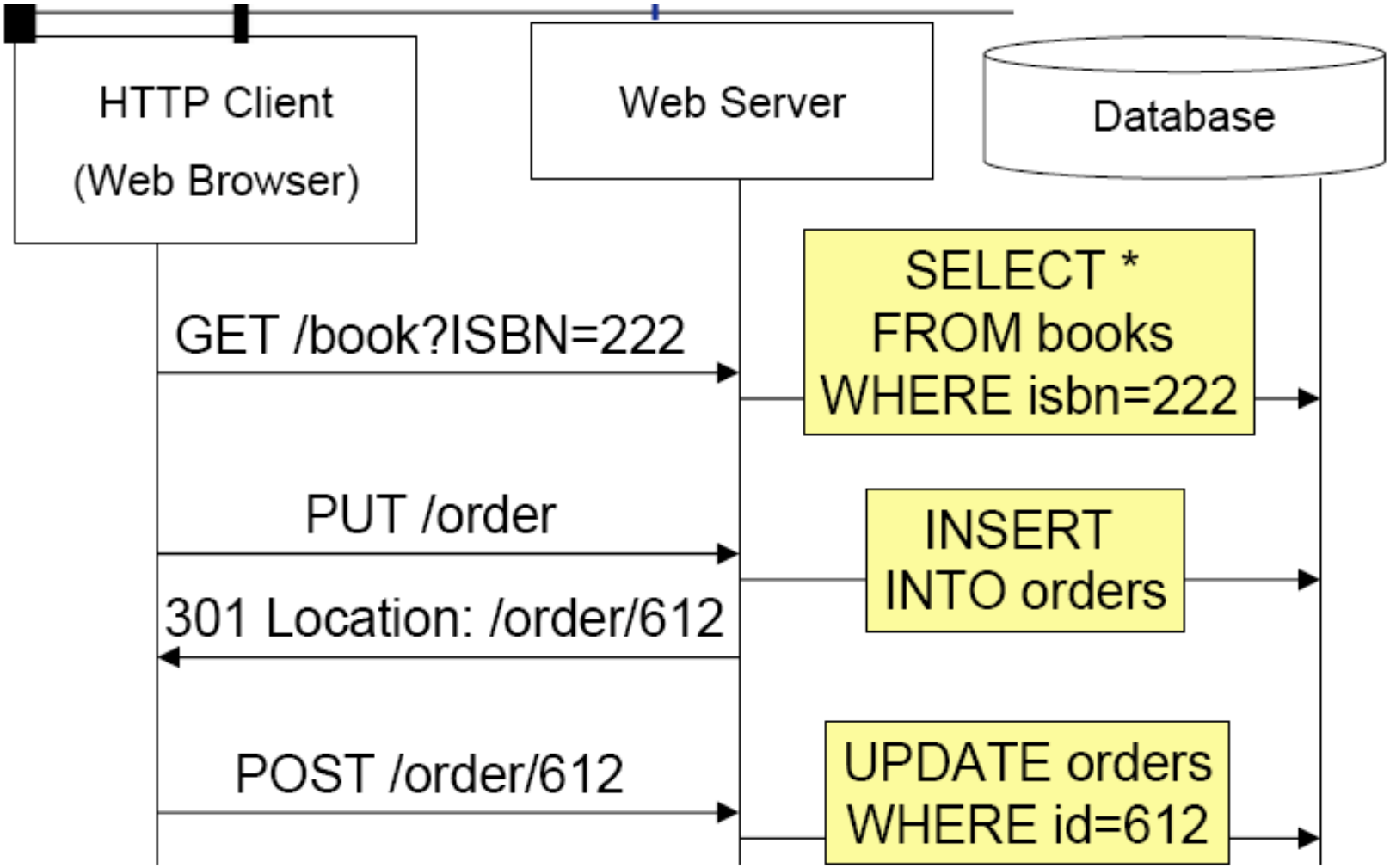
CRUD con Rest

CRUD	REST	
CREATE	POST	Create a sub resource
READ	GET	Retrieve the current state of the resource
UPDATE	PUT	Initialize or update the state of a resource at the given URI
DELETE	DELETE	Clear a resource, after the URI is no longer valid

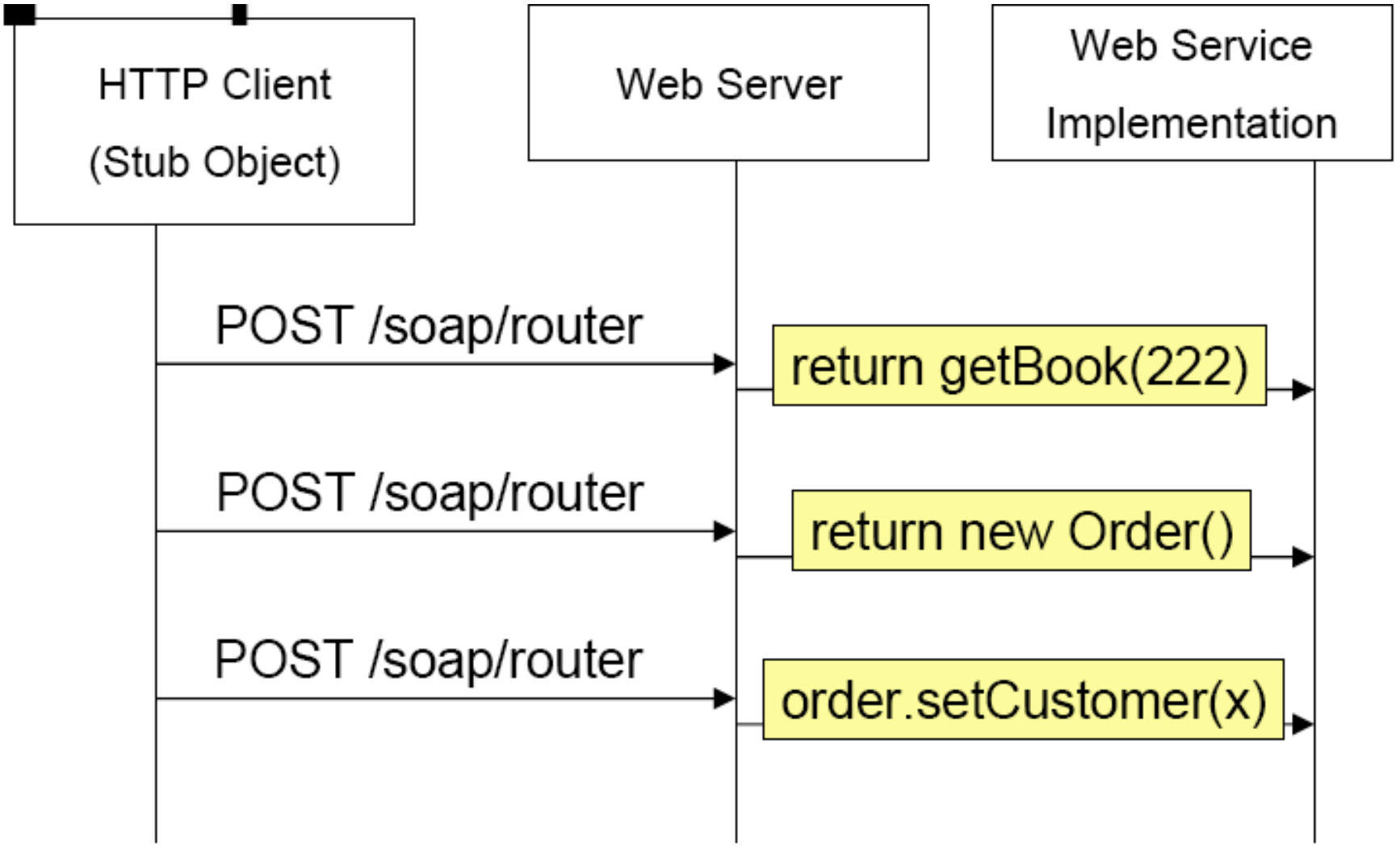
URI's


http://tools.ietf.org/html/rfc3986
URI Scheme
Authority
Path
https://www.google.ch/search?q=rest&start=10#1
Query
Fragment

Ejemplo secuencia REST



Ejemplo SOAP desde perspectiva REST



SOAP *versus* REST

- Polémica sobre el estilo dominante en SOA.
  - REST enfatiza el papel de los intermediarios: caches, proxies, gateways, etc
  - REST es adecuado para transferencias y transformaciones simples. Transacciones complejas requieren sin duda SOAP.
  - No hay herramientas para implementar REST rigurosamente; sí en cambio las hay para SOAP RPC.
  - Idem en relación con middlewares

SOAP vs REST

- ebXML adoptó SOAP como protocolo de base
- SOAP incorporó ideas de ebXML a través de WS-*
  - Integridad de mensajes, no-repudio, mensajería confiable, flujo de procesos de negocios, negociación de protocolos
- SOAP puede alcanzar mejor performance porque puede ser *stateful* y no tiene que trasmitir información de estado.
- A la larga el comité de WSA en W3C no optó entre REST y SOAP, sino que reformuló SOAP 1.2 adoptando ideas de REST
- "Intercambio asincrónico de documentos" ha ganado *mindshare* a expensas de "RPC"
- "Web services como objetos distribuidos" tiene cada vez menos adeptos
- DTD y tecnologías similares han sido eliminadas

Estándares REST y SOAP