IT4409: Web Technologies and e-Services
2018-2

RESTFUL Web service

Instructor: Dr. Thanh-Chung Dao

Slides by Dr. Binh Minh Nguyen

Department of Information Systems

School of Information and Communication Technology

Hanoi University of Science and Technology

GET: fetch information

- To fetch a web page, the browser does a GET on some URI and retrieves a representation (HTML, plain text, JPEG, or whatever) of the resource identified by that URI
- GET is fundamental to browsers
- REST requires a few more verbs to allow taking actions

Four verbs for every noun

- GET to retrieve information
- POST to add new information, showing its relation to old information
- PUT to update information
- DELETE to discard information

What's REST?

So what's REST already?

- REpresentational State Transfer
- An architectural style, not a toolkit
- “We don't need no toolkits!”
- A distillation of the way the Web already works

REST defined

- Resources are identified by uniform resource identifiers (URIs)
- Resources are manipulated through their representations
- Messages are self-descriptive and stateless
- Multiple representations are accepted or sent
- Hypertext is the engine of application state

HTTP Request/Response As REST

REST style

- Client-server
- Stateless
- Cached
- Uniform interface
- Layered system
- (Code on demand)

A web page is a resource?

- A web page is a *representation* of a resource
- Resources are just concepts
- URIs tell a client that there's a concept somewhere
- Clients can then request a specific representation of the concept from the representations the server makes available

State

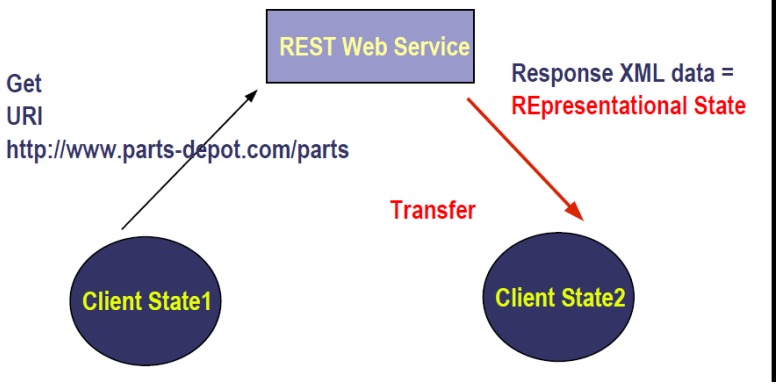
- “State” means application/session state
- Maintained as part of the content transferred from client to server back to client
- Thus any server can potentially continue transaction from the point where it was left off
- State is never left in limbo

Transfer of state

- Connectors (client, server, cache, resolver, tunnel) are unrelated to sessions
- State is maintained by being transferred from clients to servers and back to clients

What?

REpresentational State Transfer



REST and HTTP

- REST is a *post hoc* description of the Web
- HTTP 1.1 was designed to conform to REST
- Its methods are defined well enough to get work done
- Unsurprisingly, HTTP is the most RESTful protocol
- But it's possible to apply REST concepts to other protocols and systems

Other protocols

- Web interaction using other protocols is restricted to REST semantics
- Sacrifices some of the advantages of other architectures
  - Stateful interaction with an FTP site
  - Relevance feedback with WAIS search

Existing HTTP uses

- Web browsing (obviously)
- Instant messaging
- Content management
- What's outside its scope?

What do REST messages look like?

- Like what we already know: HTTP, URIs, etc.
- REST can support any media type, but XML is expected to be the most popular transport for structured information.
- Unlike SOAP and XML-RPC, REST does not really require a new message format

SOAP vs. REST

- Using Web Services and SOAP, the request would look something like this:

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
   soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
     <soap:body pb="http://www.acme.com/phonebook">
          <pb:GetUserDetails>
               <pb:UserID>12345</pb:UserID>
          </pb:GetUserDetails>
     </soap:Body>
</soap:Envelope>
```

SOAP vs. REST

- And with REST? The query will probably look like this:
  http://www.acme.com/phonebook/UserDetails/12345

- GET /phonebook/UserDetails/12345 HTTP/1.1
  Host: www.acme.com
  Accept: application/xml

- Complex query:
  http://www.acme.com/phonebook/UserDetails?firstName=John&lastName=Doe

SOAP vs. REST

SOAP	REST	
Meaning	Simple Object Access Protocol	Representational State Transfer
Design	**Standardized protocol with pre-defined rules to follow.**	**Architectural style with loose guidelines and recommendations.**
Approach	Function-driven (data available as services, e.g.: “getUser”)	Data-driven (data available as resources, e.g. “user”).
Statefulness	**Stateless by default, but it’s possible to make a SOAP API stateful.**	**Stateless (no server-side sessions).**
Caching	API calls cannot be cached.	API calls can be cached.
Security	**WS-Security with SSL support. Built-in ACID compliance.**	**Supports HTTPS and SSL.**
Performance	Requires more bandwidth and computing power.	Requires fewer resources.
Message format	**Only XML.**	**Plain text, HTML, XML, JSON, YAML, and others.**
Transfer protocol(s)	HTTP, SMTP, UDP, and others.	Only HTTP
Recommended for	**Enterprise apps, high-security apps, distributed environment, financial services, payment gateways, telecommunication services.**	**Public APIs for web services, mobile services, social networks.**
Advantages	High security, standardized, extensibility.	Scalability, better performance, browser-friendliness, flexibility.
Disadvantages	**Poorer performance, more complexity, less flexibility.**	**Less security, not suitable for distributed environments.**