Web Application Design Principles – REST (**RE**presentational **S**tate **T**ransfer)

Dik Lun Lee

REST (**RE**presentational **S**tate **T**ransfer)

- REST is:
  - Not an architecture for building systems
  - Not a programming language or programming methodology
  - Not a framework, not a library, not a tool kit, …
- REST is a set of design criteria for interaction between two independent systems
  - It encourages a "new" way of thinking about the web (somewhat philosophical)
- REST is not tied to the ‘Web’ or HTTP, etc.
  - Just that HTTP 1.1 was designed with REST in mind and has been a very popular protocol
  - REST principles can be applied to other protocols

REST Principles

- Resources are identified by uniform resource identifiers (URIs)
- Resources are manipulated through their representations
- Multiple representations are accepted or sent
- Messages are self-descriptive and clients/servers are stateless
  - Self-descriptive: Each request or representation contains all the metadata needed so that the client or server knows what it to do
  - State and state transfer: Client's state is determined by the received representation, which should link to other resources; when a new representation is received, the client has a new "state"

- **Try to work with, not against, these principles**

Representations

- The client does NOT fetch a resource but one of the representations made available by a resource
- A representation of a resource is a sequence of bytes and headers to describe those bytes.
- The particular form of the representation can be negotiated between REST components:
- Client sets specific HTTP request headers to signal what representations it's willing to accept
  - **Accept:** XML/JSON, HTML, PDF, PPT, DOCX...
  - **Accept-Language:** English, Spanish, Hindi, Portuguese…

Is a web page (that can be displayed on your browser) a resource?

REST #4: Uniform Interface

- Provides 4 basic methods for CRUD (create, read, update, delete)

Method	Function	Response
GET	Retrieve representation of resource	Returns representation of resource
PUT	Update existing or create a new resource at a specific URI; message body is the content	Responds with status message or copy of representation or nothing at all
POST	Create a new resource under some 'parent' resource (e.g., Add new messages to a forum) ; message body is the content	Returns status message or copy of representation or nothing at all
DELETE	Delete an existing resource	Returns status message or nothing at all

- All of GET/POST/PUT/DELETE can be applied to all resources (of course, server can choose to ignore any one of them)
  - E.g., http://course.ust.hk?id=comp4021&op=delete (not good: operation is a parameter)
  - Or, http://course.ust.hk/delete?id=comp4021 (better but still not good: operation is a verb but object type to be acted on is not specified, a course or a product?)

Take Home Messages

- REST is a set of design principles for client-server systems
  - Web in the 90's was very simple and ad hoc, leading to web system developers to take shortcuts and do arbitrary things
  - REST attempts to set things straight
- REST is gaining popularity over w3c web service standard (too complicated)