REST

CSC 342 - Web Technologies

Representational State Transfer

- Representational state transfer (REST) or RESTful Web services are a way providing interoperability between computer systems on the Web
- REST-compliant web services allow requesting systems to access and manipulate textual representations of web resources
- Requests are made to a resource's URI with the predefined HTTP verbs
- A response may provide hyperlinks to other related resources

REST Architectural Constraints

- Client-server
- Stateless
- Cacheble
- Layered System
- Uniform Interface
- Code on demand (optional)

Client-server

- The client-server model separates concerns.
- By separating user interface concerns from data storage concerns, portability of user interfaces is improved and scalability of server components is improved.
- Separation also allows the components to evolve independently.

Stateless

- The stateless constraint means that each request from the client contains all of the necessary information to service the request.
- Statelessness enables greater scalability since the server does not need to maintain any session state.
- Note that there is a difference between application state and resource state; the application state is data that can vary by client, but the resource state is constant across every client that requests the resource. The statelessness constraint refers to the application state.

Cacheable

- Clients and intermediaries can cache responses; this constraint requires responses to be defined as cacheable or not so that clients do not reuse stale or inappropriate data in future requests.
- Cacheable resources can reduce the number of client-server interactions improving scalability and performance.

Layered System

- In a layered system, the client cannot ordinarily determine whether it is connected directly to the end server or some intermediary server.
- Intermediary servers can improve system scalability by enabling load-balancing and by providing shared caches.

Uniform Interface

- The uniform interface constraint simplifies and decouples the architecture. There are four constraints for a uniform interface:
    - **Identification of resources:** individual resources are identified by requests. The resources are conceptually separate from the representations that are returned to the client.
    - **Manipulation of resources through representations:** When a client holds a representation of a resource, it has enough information to modify or delete the resource.
    - **Self-descriptive messages:** Each message includes enough information to describe how to process the message.
    - **Hypermedia as the engine of application state (HATEOS):** The client interacts with application through a fixed URL and all future actions a client may take are discovered within resource representations returned from the server.

Code on Demand

- Servers are able to temporarily extend or customize the functionality of the client by transferring logic to it that it can execute.
- Code on demand is the only optional constraint. If an application does not conform to the other five constraints, it is not strictly a REST application.

RESTful APIs

- Web services that adhere to the REST architectural constraints are called RESTful APIs.
- HTTP-based RESTful APIs are defined with the following aspects:
  - A base URL, such as `http://api.example.com/resources/`
  - An internet media type that defines state transition data elements – the current representation tells the client how to compose requests for transitions to the next available application states
  - Standard HTTP methods
- Unlike SOAP-based web services there is no standard for RESTful APIs. This is because REST is an architectural style, while SOAP is a protocol.

REST Conventions for CRUD

- URLs are resources and CRUD maps to HTTP verbs

Action	HTTP Verb	Example URL
Create	`POST`	`/customers`
Replace	`PUT`	`/customers/:id`
Update	`PATCH`	`/customers/:id`
Delete	`DELETE`	`/customers/:id`