REST API

Tessema Mengistu (Ph.D.)
Department of Computer Science
Virginia Tech
Mengistu@vt.edu

Outline

- Introduction
- REST Constraints
- Java and REST API

Introduction

- API – Application Programming Interface
  - Exposes a set of data and functions to facilitate interactions between computer programs and allow them to exchange information
  - Client programs use APIs to communicate with web services/applications
- REST - Representational State Transfer
  - It is an **architecture style** for designing loosely coupled applications over the network
- REST API
  - A Web API conforming to the REST architectural style
  - Sometimes referred to RESTful APIs

REST Constraints

- REST API follows six fundamental constraints or architectural styles
  - Uniform interface:
    - The URLs used for access in the system need to be uniform, consistent, and accessible using a common approach such as GET
  - Client-server:
    - Clients and servers may evolve separately without any dependency on each other as long as the interface between them is not altered
  - Stateless:
    - The server does not store anything about the latest HTTP request made by the client. Every request is treated as a new request

REST Constraints

- Cacheable:
  - All resources must be cacheable. Caching can be implemented on the server or client side
- Layered system:
  - This system allows for a layered system architecture where a client cannot tell whether it is connected to an intermediary server or the end server
- Code on demand (optional):
  - Most of the time you will be sending static representations of resources in the form of XML or JSON. This optional constraint allows you to return executable code, such as JavaScript, to support part of your application

REST Constraints

- Resource
  - The key **abstraction of information** in REST
  - Any information that we can name can be a resource
  - For example, a document or image, a database table, etc.
- Resource identifiers
  - Identify each resource

REST Constraints

- Resource representation
  - The state of a resource at any particular instant, or timestamp
  - Consist of:
    - The data
    - The metadata describing the data
    - The hypermedia links that can help the clients in transition to the next desired state
  - Can be delivered to a client in any format
    - JavaScript Object Notation (JSON), HTML, XML, or plain text

REST Constraints

- The Web has Uniform Interface
  - Web components interoperate consistently within the uniform interface's four constraints:
    1. Identification of resources
       - Unique identifier – URI
    2. Manipulation of resources through representations
       - The same exact resource can be represented to different clients in different ways.
    3. Self-descriptive messages
       - Example: HTTP header
    4. Hypermedia as the engine of application state

REST Constraints

- REST APIs embrace all aspects of the HyperText Transfer Protocol
  - request methods
    - GET, POST, PUT, DELETE, HEAD
  - response codes
  - message headers
- Perform standard database functions
  - CRUD

REST Constraints

- URI Format

```
URI = scheme "://" authority "/" path [ "?" query ] [ "#" fragment ]
```

- Scheme:
  - identifies how the resource is accessed
  - Examples: http, file, ftp, etc.
- Authority:
  - It contains the subcomponents `userinfo`, `host`, and `port`.
- Path:
  - A sequence of path segments separated by a slash(`/`)
- Query:
  - preceded with a question mark (`?`)
- Fragment:
  - preceded by a hash (`#`) symbol
  - provides direction to a secondary resource

Java and REST API

- JAX-RS
  - Java specification of API for RESTful web services
  - Provides portable APIs for developing, exposing, and accessing Web applications
  - Designed and implemented in compliance with principles of REST architectural style
  - Uses **annotations** to java objects
    - Bind specific URI patterns
    - Bind specific HTTP operations
    - Handle input/output parameters
    - . . .

JAX-RS Annotations

- **@Path(‘resourcePath’)**
  - Match the URI path, which is relative to the base URI

```
@Path("/")
public class ApiResource {

            . . .
```

JAX-RS Annotations

- **@POST, @GET, @PUT, …**
  - Handle the HTTP method requests on the matching resource path

```
@GET
@Path("categories")
@Produces(MediaType.APPLICATION_JSON)
  public List<Category> categories(@Context
         HttpServletRequest httpRequest) {
    try {
         return categoryDao.findAll();
       }
         . . .
   }
```

JAX-RS Annotations

- **@PathParam(“parameterName”)**
  - Inject values (resource identifiers) from the URL into a method parameter

```
@GET    @Path("categories/{category-id}")
@Produces(MediaType.APPLICATION_JSON)
   public Category categoryById(@PathParam("category-id") long categoryId,
             @Context HttpServletRequest httpRequest) {
       try {
             Category result = categoryDao.findByCategoryId(categoryId);
             if (result == null) {
                    throw new ApiException(“”);
                }
              return result;
          }
              . . .
      }
```

JAX-RS Annotations

- **@Produces**
  - Defines which MIME type is produced by annotated resource method
- **@Consumes**
  - Defines which MIME type is consumed by annotated resource method

Java and REST API

- Different implementations
  - Jersey
  - RESTEasy
  - Apache CXF
  - . . .

References

- https://restfulapi.net/
- https://restfulapi.net/create-rest-apis-with-jax-rs/
- REST Original Document
    - https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
        - Chapter 5
- RESTful Web Services. L. Richardson, S. Ruby. O'REILLY.