REST API Development

B. Mason
Netapp E-Series
Disclaimer

Opinion expressed here are mine and do not necessarily represent Netapp
Who am I??

• Software Engineer at Netapp E-Series AppAware
• Designer and lead for the REST API for E-Series
• Have done various API in C/C++/Java/SOAP/REST
• I am not selling a book or anything
Agenda

- What is a REST API?
  - How are they different from previous API protocols?
  - Why are they so useful?
- Technology Primer for REST
- How to build a REST API
- Documentation Standards
- Using a REST API as a client
Why we do care about the API?

- Integration, Integration, Integration

- IDC Predicts we are in the “Golden Age of APIs”

- “We don’t need a fancy GUI” we need it to plugin to X

- Enterprises don’t care about GUI, they want hardware to plugin to their Enterprise systems
  - CINDER
  - VASA
  - Etc....

- Classically handled by CLI
What is a REST API?

• Wikipedia: **Representational State Transfer (REST)** is a **software architecture style for building scalable web services.**

• Objects are exposed as Uniform Resource Identifier (URI/URL)

• Object data is accessed via HTTP(S) and encoded in something easy to parse (Plain Text/JSON/XML)

• Other attributes
  • Client/Server
  • Stateless
  • Cacheable
  • Uniform
Why are they different/more useful?

- **REST IS SIMPLE**
- Like SOAP and XMLRPC, it’s “Text Based”
  - No weird binary formats to parse
  - Easy to consume by any language
  - Relies on standard compression algorithms for speed
- Unlike SOAP, it is not overdesigned
  - It’s not even designed, it’s a pattern
  - No committees, grass roots
- It does not have a standard description language
  - No IDL, WSDL, MIDL
Explore Simple Web Service

+ 

Let's look at a Demo
Technology Primer for REST
Definitions

- HTTP – Hyper Text Transfer Protocol
- HTTPS – Secure HTTP (AKA, HTTP over SSL)
- Mime Type - is a two-part identifier to standardize file-formats across the Internet: (text/plain, text/html, application/json)
- SSL/TLS – Secure Socket Layer / Transport Layer Security
- URL/URI – The address of a resource (http://host:port/path)
- Query String – Part of the URL after the question mark. Contains key/value data
  - http://somehost.com/resource?key=value&key=value
- JSON – Java Script Object Notation
HTTP Verbs

- **GET** – Gets a Resource (What happens when you surf)
- **POST** – Creates a new Sub Resource
- **PUT** – Updates a resource
- **DELETE** – Deletes or Resets a Resource
- **HEAD** – Like GET but just gets the HEADERS
- **OPTION** – Used in CORS
- **TRACE / CONNECT** – Not really used in REST
Sample GET Request

HTTP GET

Host: somehost.domain.com
Accept: application/json
Accept-Language: en-us

Status: ok
Content-Type: application/json

{
    "message": "Hello REST"
}
Common HTTP Headers

- Host – Target Host
- Content-Type – Mime Type for the inbound content
- Accept – Mime Types that are acceptable responses
- Accept-Encoding – Acceptable Encoding (zip, etc…)
- Status – The Status code for the response (200,400,500…)
How to build a REST API
Building a REST Server

- All you really need is a way to generate dynamic content

- Frameworks can be a huge help
  - Handles URL mapping to handlers
  - Handles Language Object to Payload and back (JSON, XML etc…)

- REST Frameworks are everywhere
  - Django for Python
  - Certainly ones for .net
  - Several Java Frameworks

- We will focus on Java because that is what I know
Simple Servlet

```java
@WebServlet(value ="/test", name = "SimpleRest")
public class SimpleRest extends HttpServlet {

protected void doGet(HttpServletRequest req,
                      HttpServletResponse resp) {
    PrintWriter out;

    out=new PrintWriter(response.getOutputStream());
    resp.setHeader("Content-Type","application/json");
    out.println("{"message":"Hello World"}");

    out.flush();
    out.close();
}
}
```
JAX-RS

- Java Specification for REST API
- JSR 339
- Set of annotations to define REST API
- Makes creating REST APIs pretty easy
  - Don’t tell my boss
- Jersey is an implementation of JSR 339
- Jersey with Jackson is a frequent combination
@Path("hello")
public class HelloJersey {
    @GET
    @Produces("application/json")
    public ResponseOne handleGet() {
        ResponseOne ret;
        ret = new ResponseOne("Hello Jersey");
        return ret;
    }
}

@XmlElement
public class ResponseOne implements Serializable{
    private String message;

    public ResponseOne() {
    }

    public ResponseOne(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
Documenting REST APIs

- Good documentation is a Key to user acceptance!
- Quick search will find many options
- WADL – Web Application Description Language
- Swagger – A Open Source project for REST
- Various commercial offerings
Swagger World

- Swagger has a language neutral JSON representation of REST API
- There are tools to produce the JSON
- There is a Web UI project
  - Reads the Swagger JSON definition of your API
  - Presents interactive documentation
- Integrates with various languages
Swagger UI demo
Question? Do you start with documentation or do you start with code?
Embedding Docs in Code

- How this works is obviously language specific
- For Java, Swagger tools read JAX-RS annotations and custom Swagger annotations
- Python’s Django Framework uses Swagger
Using a REST API
Let's start with a demo

- The Advance REST Client is a plugin for Chrome to test REST APIs
- cURL is a command Unix command line for accessing web resources
SDK For REST APIs

- You don’t need any special SDKs to consume a REST Server!
  - All modern languages have libraries for HTTP.
  - JSON processing is ubiquitous

- SDKs are a nice to have
  - For strongly type languages like Java, having class definition is nice
  - Swagger provides tools to generate client SDKs
Questions?
Links

- https://jax-rs-spec.java.net/
- http://www.django-rest-framework.org/
- http://swagger.io/
- http://www.w3.org/Protocols/rfc2616/rfc2616.html
- https://jersey.java.net/