

Master of Engineering in Internetworking

Lab # 6

Cisco DevNet

INWK 6312

Copyright© Internetworking Program, Dalhousie University

### About DevNet

DevNet is Cisco's resource and professional community for network engineers and application software developers working with Cisco APIs and platforms. Joining the community is free of charge and carries no requirements or obligations. And what you get is pretty awesome. Like the ability to follow and participate in technical support forums where you can get answers quickly to questions from fellow community members who may be a few steps further along the same path that you are on. Plus, you'll be able to create your own DevNet profile, so you can receive information about events and opportunities specific to your area of interest.

# REST API Basics

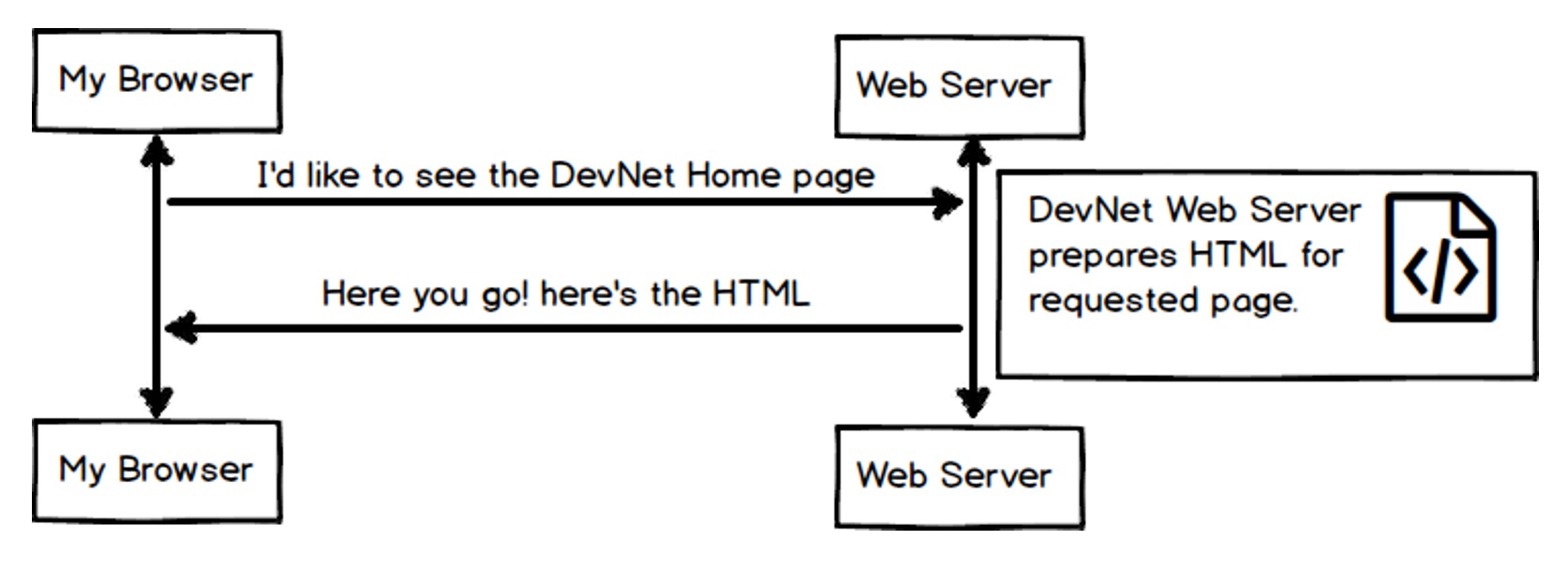
An API (Application Programming Interface) is a way for two pieces of software to talk to each other. Think about the ways you typically interface with software. For example, you might open up a web interface to access your email. You might have a specific workflow to open up messages and file them away for later. Each of these workflows has a specific "interface" or way in which you achieve a certain task.

## Step 2. What is a REST web service?

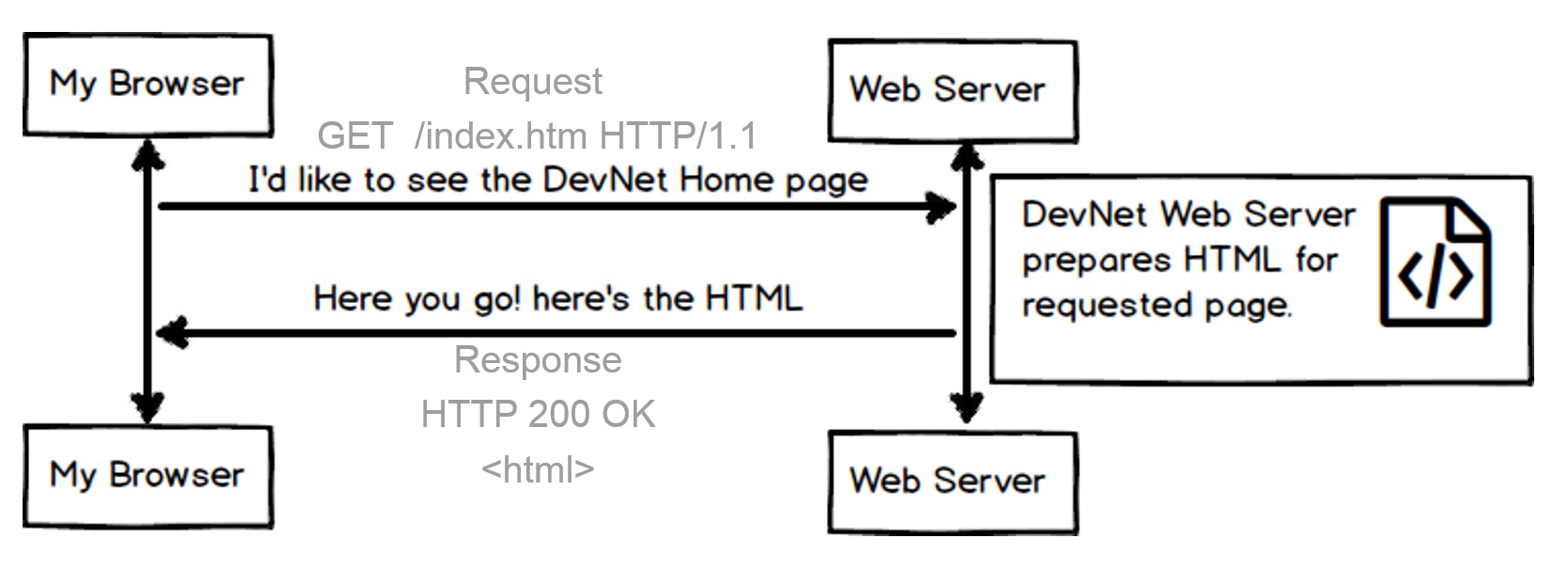
In general, a web service is a way for two systems to communicate through a defined interface. In the past 20 years, there have been two major types of Web Services – **REST** or **SOAP**. In the last 10 years, the REST approach has become increasingly popular.

What is REST? [REST (Representational State Transfer)](https://en.wikipedia.org/wiki/Representational_state_transfer) is an architecture style for designing networked applications. A REST web service is a web service that is as easy to call as making an [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) request.

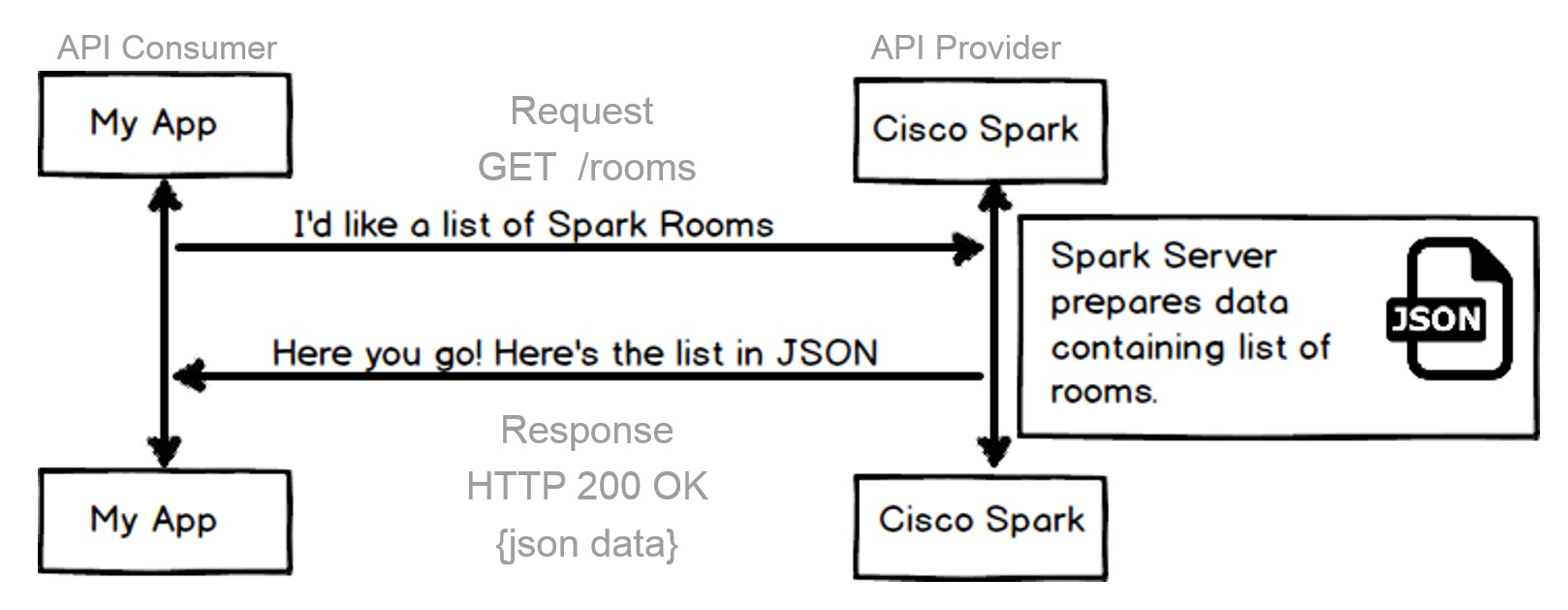
Look at the diagram below. It shows how a browser retrieves web pages. Normally, after a user requests a particular resource in a browser, the appropriate web server responds with the proper HTML to display the page to your client browser.



Behind the scenes, HTTP(S) uses CRUD (Create, Read, Update, Delete) operations on the wire to request data. In the example below, our browser is issuing a GET operation to read the associated web page. The web server returns the associated data and an HTTP response to the client browser.



RESTful interfaces offer these same CRUD (Create, Read, Update, Delete) operations using HTTP(S). Browsers are replaced by software to interface with the RESTful service. The diagram below illustrates the same concept; however, browsers are replaced by software leveraging the REST API.



## Examine the Anatomy of a REST API Query

To construct a request, you need to know the following information for the API that you are calling. You can find this information in the API reference documentation.

* **Method**
  + GET - Retrieve data
  + POST - Create something new
  + PUT - Update data
  + DELETE - Delete data
* **URL**
  + The URL for the endpoint you want to call
  + Example: <http://api.ciscospark.com/v1/rooms>
* **URL Parameters**
  + The parameters that you can pass as part of the URL.
* **Authentication**
  + You need to know the authentication type to use. Basic HTTP, token based, and OAuth are common types.
  + Authentication credentials
* **Custom Headers**
  + Does the API require you to send any HTTP Headers?
  + Example: Content-Type: application/json
* **Request Body**
  + JSON or XML containing data that is needed to complete request can be sent in the body of the request



### A Little About Authentication

There are different types of authentication for REST APIs. Authentication is used to control access and access rights to the REST APIs. For example, some users might have read-only access which means that they can use only the parts of the API that read data. Other users might have both read and write access. This means they can use the API to perform operations that not only read data but also add, edit, and delete data. These access rights are typically based on user assigned roles such as **Administrator** where a user would have full rights to change the data. For example, a plain **User** role might have read-only access rights.

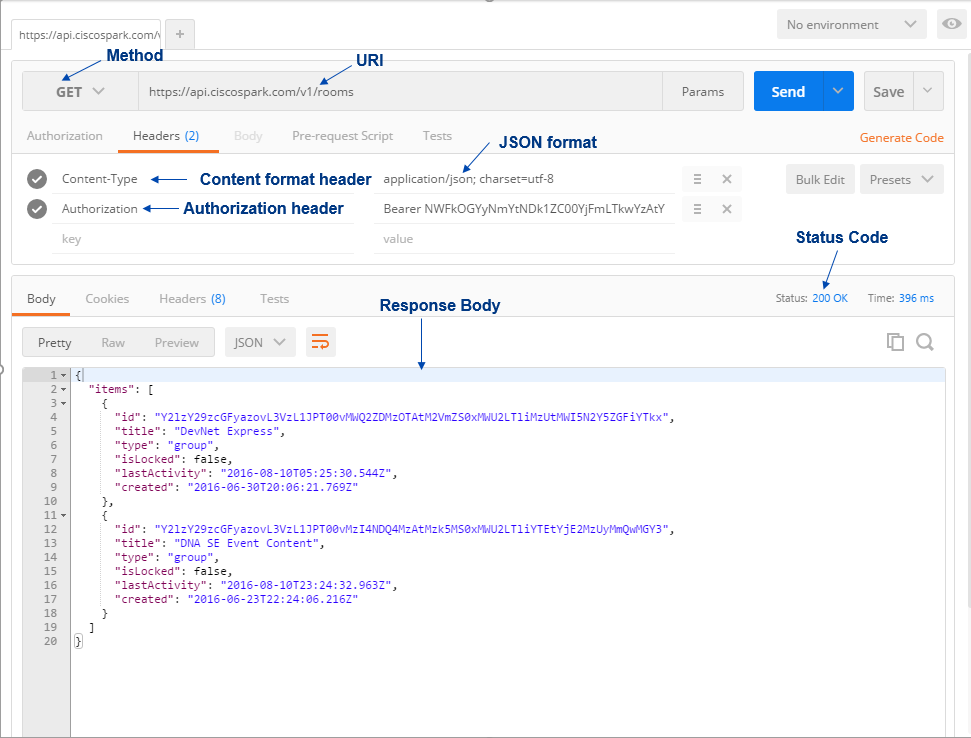
#### Types of Authentication Controls

* **None**: the Web API resource is public, anybody can place call. Generally the case for GET methods, rarely for POST, PUT, DELETE.
* **Basic HTTP**: The username and password are passed to the server in an encoded string.
  + Authorization: Basic ENCODEDSTRING
  + See [Basic Authentication](https://en.wikipedia.org/wiki/Basic_access_authentication) for more information.
* **Token**: A secret key generally retrieved from the Web API developer portal.
  + The keyword may change from one Web API to another: Bearer, token..
  + Passed with each API call.
  + See [Token Based Authentication](https://scotch.io/tutorials/the-ins-and-outs-of-token-based-authentication) for more information.
* **OAuth**: A sequence flow is initiated to retrieve an access token from an Identity Provider. The token is then passed with each API call.
  + Open standard. User rights are associated with the token (OAuth scope).
  + The token expires. It can be revoked. It can also be re-issued via a refresh token.
  + See [OAuth](https://en.wikipedia.org/wiki/OAuth) for more information.

### What is in the Response?

The API Reference Guide includes information about the attributes to be sent and returned. The returned data is defined in the Response portion which includes the HTTP status codes along with the data format and attributes.

* HTTP Status Codes
  + HTTP status codes are used to return success, error, or other statuses. <http://www.w3.org/Protocols/HTTP/HTRESP.html>
  + Some common examples are:
    - 200 OK
    - 202 Accepted/Processing
    - 401 Not Authorized
* Content
  + Often returned in different formats based upon the request. Common formats are: JSON, XML and Text.
    - JSON (most commonly used)
    - {
    - "data": [{
    - "company": "Cisco Systems",
    - "event": "DevNet Express",
    - "location": "Las Vegas, NV, USA"
    - }, {
    - "company": "Cisco Systems",
    - "event": "Cisco Live America",
    - "location": "San Jose, CA, USA"
    - }]
    - }

Here is an example that shows a REST API request and response all in the same window:  
  


Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

What is Git? •

Git is a distributed version control system, developed by Linus Torvalds (of Linux kernel fame).

– Version Control System: Something that keeps track of the changes made to files (commit)

– Distributed: Everyone has a copy (clone) of the git repository

– So, a Distributed Version Control System allows you to have a copy of the complete development history of the project on your local workstation (i.e., all changes ever made in the history of the project are in on your computer!!).

• This means that at any moment you can go back to the state of the project as it existed at any other point in time. (reset)

• Because of the distributed nature of git (i.e., everyone has a repository clone on their computer), they can develop locally and, when ready, send (push) those changes to the global repository (i.e., to the server).

**What is GitHub?**

**GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.**

**GitHub Tutorial**

[**https://guides.github.com/activities/hello-world/**](https://guides.github.com/activities/hello-world/)

To complete this tutorial, you need a [GitHub.com account](http://github.com/) and Internet access.