Application Programming Interfaces (APIs)

Outline

- Data formats
  - JSON
  - XML
  - YAML
- APIs
- REST APIs

Data Formats

Data serialization

- **Data serialization** is the process of translating a data structure or object into a format that can be stored or transmitted and reconstructed later.
- The opposite operation, extracting a data structure from a series of bytes, is **deserialization**

```
JSON.stringify(
  spriteObject = {
     sourceX: 0,
     sourceY: 0,
     sourceWidth: 32,
     sourceHeight: 32,
     width: 32,
     height: 32,
     x: 0,
     y: 0,
     vx: 0,
     vy: 0,
     }
)
```

```
"{"sourceX":0,"sourceY":0,"so
urceWidth":32,"sourceHeight":
32,"width":32,"height":32,"x"
:0,"y":0,"vx":0,"vy":0}"
```

Data formats

- Data formats are simply a way to store and exchange data in a structured format.
- These are some common data formats that are used in many applications:
  - JavaScript Object Notation (JSON)
  - eXtensible Markup Language (XML)
  - YAML Ain't Markup Language (YAML)

Data format rules

- Data formats have rules and structure similar to what we have with programming and written languages.
- Each data format will have specific characteristics:
  - Syntax, which includes the types of brackets used, such as [ ], ( ), { }, the use of white space, or indentation, quotes, commas, and more.
  - How objects are represented, such as strings, lists, and arrays.
  - How key/value pairs are represented.

Compare data formats

```
{
    "message": "success",
    "timestamp": 1560789260,
    "iss_position": {
        "latitude": "25.9990",
        "longitude": "-132.6992"
    }
}
```

JSON Format

```
message: success
timestamp: 1560789260
iss_position:
        latitude: '25.9990'
        longitude: '-132.6992'
```

YAML Format

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
<message>success</message>
<timestamp>1560789260</timestamp>
<iss_position>
   <latitude>25.9990</latitude>
   <longitude>-132.6992</longitude>
</iss_position>
</root>
```

XML Format

JSON data format

- JSON is a standard, human-readable data format used by applications for storing, transferring and reading data.
- JSON is a very popular format used by web services and APIs to provide public data.
- JSON was derived from JavaScript, but it is language-independent and many modern programming languages, including Python, are able to generate and read JSON data.

JSON data format

- JSON has four primitive data types: string, number, Boolean, and null.
  - A **string** is a text value. It is surrounded by double quotes.
  - A **number** is a numeric value. It is NOT surrounded by quotes.
  - A **Boolean** is a data types that has only two possible values which are **true** and **false**.
  - A **null** value represents the intentional absence of any object value.
- JSON also has two 'structured' data types: object and array.

JSON syntax rules

- These are some of the characteristics of JSON:
- It uses a hierarchical structure and contains nested values.
- It uses braces { } to hold objects and square brackets [ ] hold arrays.
- Its data is written as key/value pairs.
- White space is **insignificant**.

JSON syntax rules

▶ An object is one or more key/value pairs enclosed in braces { }. The syntax for a JSON object includes:

- Keys must be strings within double quotation marks " ".
- Values must be a valid JSON data type (string, number, Boolean, null, array, or another object).
- Keys and values are separated by a colon.
- Multiple key/value pairs within an object are separated by commas.

JSON syntax rules

▶ A key may contain more than one value. This is known as an array. Characteristics of arrays in JSON include:

- The key followed by a colon and a list of values enclosed in square brackets [ ].
- The array can contain multiple value types including a string, number, Boolean, object or another array inside the array.
- Each value in the array is separated by a comma.

JSON examples

```
{
 "interface": "GigabitEthernet1/1",
 "is_up": true,
 "ipaddress": "192.168.1.1",
 "netmask": "255.255.255.0",
 "speed": 1000
}
```

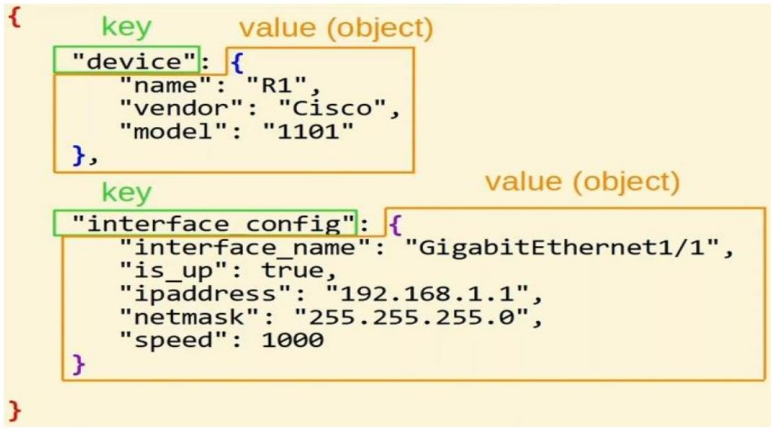
```
{"interface":"GigabitEthernet1/1","is_
up":true,"ipaddress":"192.168.1.1","ne
tmask":"255.255.255.0","speed":1000}
```

JSON examples

```
{
    "device": {
       "name": "R1",
       "vendor": "Cisco",
       "model": "1101"
    },

    "interface_config": {
       "interface_name": "GigabitEthernet1/1",
       "is_up": true,
       "ipaddress": "192.168.1.1",
       "netmask": "255.255.255.0",
       "speed": 1000
    }
}
```

JSON examples



- Objects within objects are called nested objects.

JSON examples

```
{
    "interfaces": [
        "GigabitEthernet1/1",
        "GigabitEthernet1/2",
        "GigabitEthernet1/3"
    ],

    "random_values": [
        "Hi",
        5
    ]
}
```

XML data format

- XML was developed as a markup language, but is now used as a general data serialization language.
- HTML is the standardized markup language for creating web pages.
- Markup languages (i.e., HTML) are used to format text (font, size, color, headings, etc.).
- Whitespace is **insignificant**.

XML data format

- XML is self-descriptive. It encloses data within a related set of tags: <tag>data</tag>
- Unlike HTML, XML uses no predefined tags or document structure.
- XML objects are one or more key/value pairs, with the beginning tag used as the name of the key: <key>value</key>

XML example

```
<?xml version="1.0" encoding="UTF-8" ?>
<ietf-interfaces:interface>
   <name>GigabitEthernet2</name>
   <description>Wide Area Network</description>
   <enabled>true</enabled>
   <ietf-ip:ipv4>
      <address>
         <ip>172.16.0.2</ip>
         <netmask>255.255.255.0</netmask>
      </address>
      <address>
         <ip>172.16.0.3</ip>
         <netmask>255.255.255.0</netmask>
      </address>
      <address>
         <ip>172.16.0.4</ip>
         <netmask>255.255.255.0</netmask>
      </address>
   </ietf-ip:ipv4>
</ietf-interfaces:interface>
```

YAML data format

- YAML originally meant Yet Another Markup Language.
- To distinguish its purpose as a data-serialization language, it was repurposed to YAML Ain't Markup Language.
- Whitespace is **significant** (unlike JSON and XML).
- Indentation is very important.
- YAML files start with ---.
- - is used to indicate a list.
- Key value pairs are represented as key:value.

YAML example

```
{
   "ietf-interfaces:interface": {
      "name": "GigabitEthernet2",
      "description": "Wide Area Network",
      "enabled": true,
      "ietf-ip:ipv4": {
         "address": [
            {
               "ip": "172.16.0.2",
               "netmask": "255.255.255.0"
            },
            {
               "ip": "172.16.0.3",
               "netmask": "255.255.255.0"
            },
            {
               "ip": "172.16.0.4",
               "netmask": "255.255.255.0"
            }
         ]
      }
   }
}
```

```
ietf-interfaces:interface:
   name: GigabitEthernet2
   description: Wide Area Network
   enabled: true
   ietf-ip:ipv4:
      address:
         - ip: 172.16.0.2
           netmask: 255.255.255.0
         - ip: 172.16.0.3
           netmask: 255.255.255.0
         - ip: 172.16.0.4
           netmask: 255.255.255.0
```

Quiz

- Which of the following describes a key/value pair?

a) A key is a string and a value is a list.

b) A key/value pair is another name for an array.

c) A key describes the data and the value is the data itself.

d) A key/value pair is only used in JSON.

X

- TRUE or FALSE: White space in JSON format is significant and must be correctly formatted.

Quiz

- This data format is a minimalist format that is very easy to read.
  a) HTML.
  b) XML.
  c) JSON.
  d) YAML.

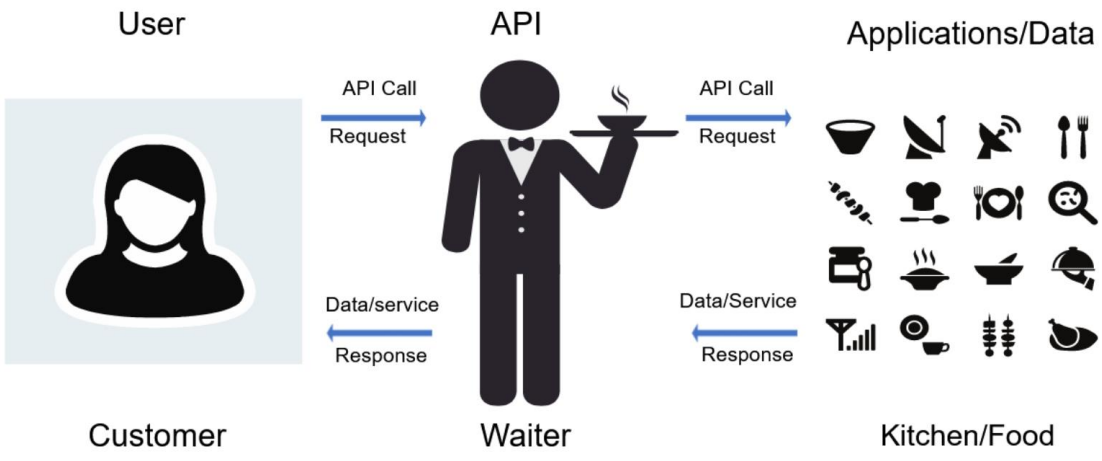
- This data format is self-descriptive through the use of the <tag>data</tag> structure.
  a) HTML.
  b) XML.
  c) JSON.
  d) YAML.

APIs

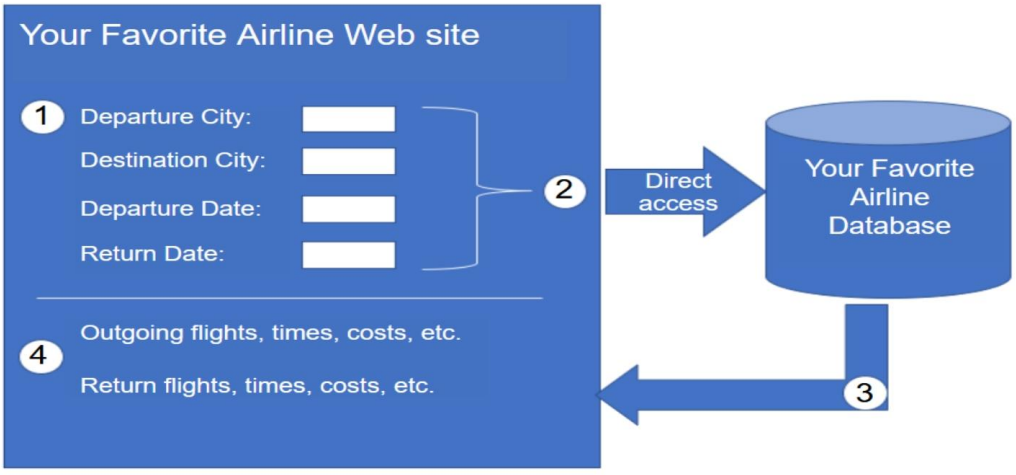
The API concept

- An API is software that allows other applications to access its data or services.
- It is a set of rules describing how one application can interact with another, and the instructions to allow the interaction to occur.
- The user sends an API request to a server asking for specific information and receives an API response in return from the server along with the requested information.

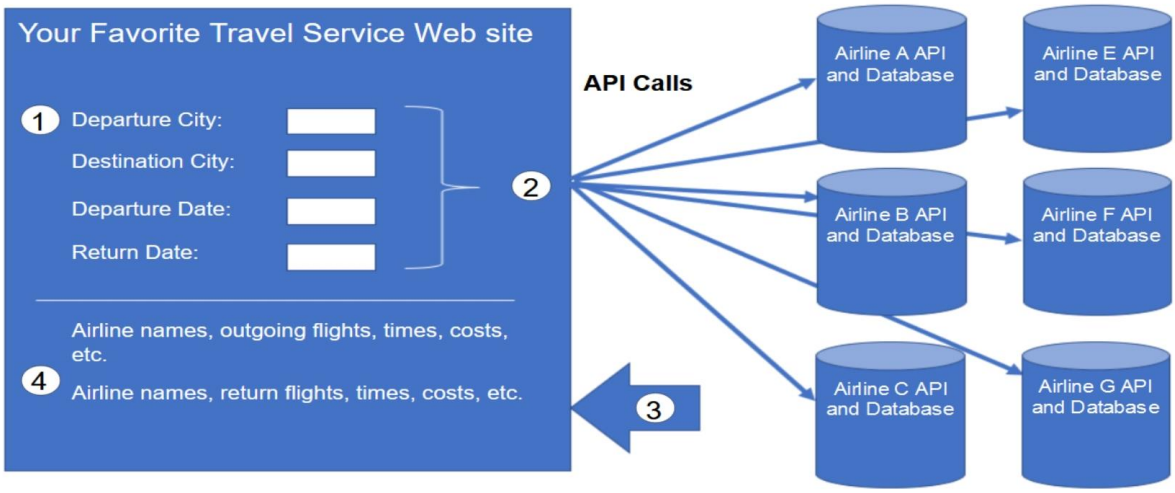
The API concept



An API example



An API example



The API concept

- The API acts as a kind of messenger between the requesting application and the application on the server that provides the data or service.
- The message from the requesting application to the server where the data resides is known as an API call.

Open, Internal, and Partner APIs

Open/Public APIs		APIs that are publicly available
Internal/Private APIs		APIs used within an organization
Partner APIs		APIs between a company and its business partners

Open, Internal, and Partner APIs

- An important consideration when developing an API is the distinction between open, internal, and partner APIs:
- **Open APIs or Public APIs** - These APIs are publicly available and can be used with no restrictions. Because these APIs are public, many API providers, such as Google Maps, require the user to get a free key, or token, prior to using the API. This is to help control the number of API requests they receive and process.
- **Internal or Private APIs** - These are APIs that are used by an organization or company to access data and services for internal use only. An example of an internal API is allowing authorized salespeople access to internal sales data on their mobile devices.
- **Partner APIs** - These are APIs that are used between a company and its business partners or contractors to facilitate business between them. The business partner must have a license or other form of permission to use the API. A travel service using an airline's API is an example of a partner API.

Quiz

- True or False: An API is a set of rules describing how one application can interact with another, and the instructions to allow the interaction to occur.
- Which of the following APIs would be used exclusively between Google and Cisco?

a) Open API

b) Internal API

c) Partner API

REST APIs

REST and RESTful APIs

- APIs are used to facilitate data exchanges between programs.
- **REST** (representational state transfer) **APIs** (**REST-based APIs** or **RESTful APIs**).
- REST is a **framework** for APIs. It describes a set of rules about how the API should work.
- The **six constraints**, or rules, of RESTful architecture are:
  - Client-server
  - Stateless
  - Cacheable
  - Uniform interface
  - Layered system
  - Code-on-demand

REST APIs

- For applications to communicate over a network, networking protocols must be used to facilitate those communications. For REST APIs, HTTP(S) is the most common choice.
- Simply stated, a REST API is an API that works on top of the HTTP protocol.

CRUD

- **CRUD** (Create, Read, Update, Delete) refers to the operations (actions) that can be performed using REST APIs.
  - **Create** operations are used to create new variables and set their initial values.
  - **Read** operations are used to retrieve the value of a variable.
  - **Update** operations are used to change the value of a variable.
  - **Delete** operations are used to delete variables.
- HTTP uses 'verbs' ('methods') that map to these CRUD operations.

CRUD and HTTP methods

REST Operation	HTTP Method (Verb)
Create	POST
Read	GET
Update	PUT
Delete	DELETE

RESTful constraints

- REST APIs use a **client-server** architecture.
- The client uses API calls (HTTP requests) to access resources on the server.
- The separation between the client and server means that they can both change and evolve independently of each other.
  - When the client application changes or the server application changes, the interface between them must not break.

RESTful constraints

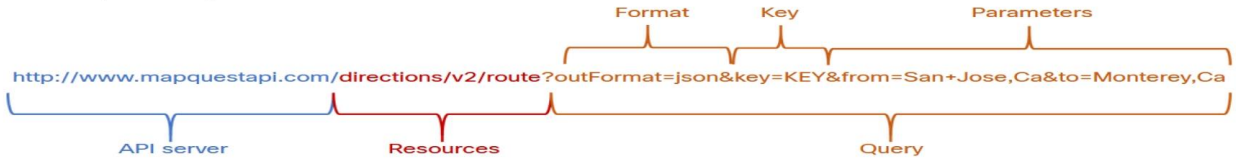
- REST APIs exchanges a **stateless**.
- This means that each API exchange is a separate event, independent of all past exchanges between the client and server.
  - The server does not store information about previous requests to determine how it should respond to new requests.
- TCP is an example of a stateful protocol.
- UDP is an example of a stateless protocol.

RESTful constraints

- REST APIs must support **caching** of data.
- This means storing data for future use.
  - For example, your computer might cache many elements of a web page so that it doesn't have to retrieve the entire page every time you visit it.
  - This improves performance for the client and reduces the load on the server.
- Not all resources have to be cacheable, but cacheable resources must be declared as cacheable.

RESTful API request

- These are the different parts of the API request:
- **API Server** - This is the URL for the server that answers REST requests. In this example it is the MapQuest API server.
- **Resources** - Specifies the API that is being requested. In this example it is the MapQuest directions API.
- **Query** - Specifies the data format and information the client is requesting from the API service. Queries can include:



RESTful API request

- **Query** - Specifies the data format and information the client is requesting from the API service. Queries can include:
  - **Format** – This is usually JSON but can be YAML or XML. In this example JSON is requested.
  - **Key** - The key is for authorization, if required. MapQuest requires a key for their directions API. In the above URI, you would need to replace "KEY" with a valid key to submit a valid request.
  - **Parameters** - Parameters are used to send information pertaining to the request. In this example, the query parameters include information about the directions that the API needs so it knows what directions to return: "from=San+Jose,Ca" and "to=Monterey,Ca".

RESTful API request

- Many RESTful APIs, including public APIs, require a key. The key is used to identify the source of the request. Here are some reasons why an API provider may require a key:
- To authenticate the source to make sure they are authorized to use the API.
- To limit the number of people using the API.
- To limit the number of requests per user.

How to execute a REST API request?

- Web browser – HTTP
  - Some RESTful API requests can be made by typing in the URI from within a web browser.
- Command Line - CURL
- Application - Postman
- Programming Language - Python, Javascript, and more

Quiz

- An API in considered RESTful if it has which of the following features?

a) Stateful
b) Stateless
c) Cacheless
d) Client-server

- Which of the following make up the query portion of a RESTful request?

a) API Sever
b) Resouces
c) Key
d) Format
e) Parameters

Thank you