How do applications exchange services and communicate?

Goals

In this case you learn how applications communicate to exchange services through the use of **Application Programming Interfaces**, better known as **APIs**. You will also gain an appreciation of the usefulness and impact of this technology in today's digital world as well as interact with some public Web APIs.

Introduction

Business Context: The insurance market is in a process of transformation. An insurance company wants to expand its business by offering new services to meet the needs of its customers. Your company has hired you as a data scientist to identify new trends in the market and propose technological innovations that generate new business opportunities.

Business Problem: Your manager would like you to answer: **How to take advantange of public insurance data to integrate it with the company's internal processes?**

Analytical Context: In this case, we will learn how to pull information from public web applications.

What are APIs?

We interact with different types of devices in our daily lives. Each of them has its own way of receiving information. Think for a moment about how we interact with the items shown in the following pictures:

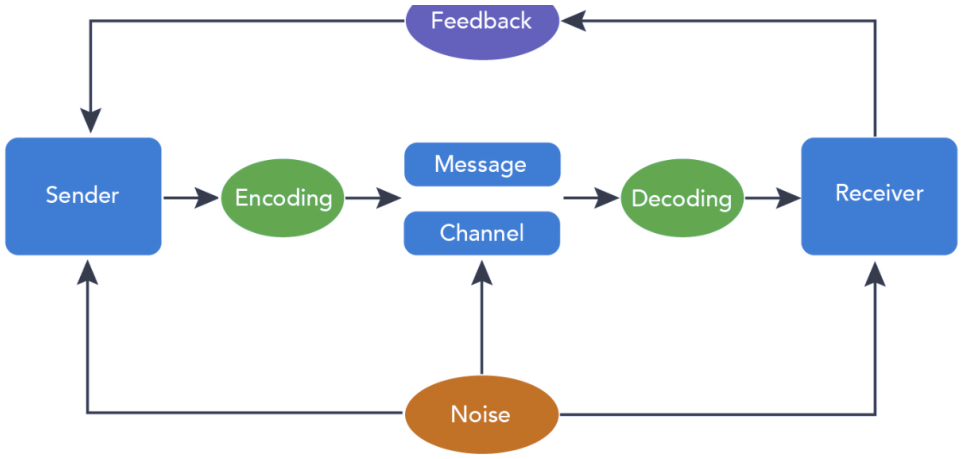
We know clearly what to expect when we interact with a coffee maker - a cup of magic to start the day. Or maybe we need to print out a report for our boss or relax with a videogame after work. To obtain the desired results we need to know how to interact with each device. We need to be able to make the device understand our requests, and then the device has to communicate the results back to us in a clear and effective way. The communication mechanisms that manufacturers put in their products are called **interfaces**. Buttons and screens are examples of common interfaces. As end users, we do not need to understand the inner workings of a device; we just need to know how to communicate with it.

APIs are interfaces, with the difference that they don't necessarily help physical devices communicate with humans, but rather facilitate information transfer between software programs. Imagine how useful it would be for the insurance company to know which places the customer visits most frequently in order to offer them new services. For instance, if a customer visits a car repair often, that can mean that they are in need of auto insurance. By using the Google Maps API, location service developers can provide the consumer with a wide range of applications that can communicate with each other seamlessly to increase sales and improve customer relationship management.

Before getting into deep technical concepts, let's review the main concepts in every communication process.

Communication Process

There are key elements in each communication process: The sender, the receiver and the message that is sent through a channel.



In the context of Web applications, the communication process has two sides. The sender is called the Client and the receiver is the Server.

The message can be encoded in some format and in turn the response must be decoded.

There are several common formats used by applications - JavaScript Object (**JSON),** eXtensive Markup Language (**XML**), HyperText Markup Language (**HTML**) and plain text.

Web APIs

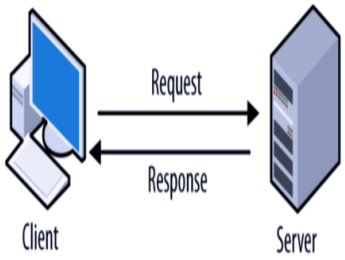
A Web API is a program for exchanging messages between web applications. APIs can be **consumed** (used to obtain information) or **produced** (designed to distribute information, that is, to expose a service). Services can be consumed by different kinds of clients like web browsers, mobile apps, and other programs.

A **request** is an API call that uses a direct access **URI** (Uniform Resource Identifier). Client applications send requests to the server that offers a service. There are different ways to send the requests and they are called methods.

A **response** is the server's answer to the request. The server can respond by sending status codes that help the client to know if the request was processed successfully or not. If it wasn't, this code also provides information about the cause of the problem.

There are public and private Web APIs. Some require a key or a token for authentication. The examples below are going to consume public API services that do not require authentication.

There is a widely used standard for creating APIs called **REST** (Representational State Transfer design pattern). It is a set of guidelines and stylistic constraints followed by programmers to write APIs that can reliably send and receive data between clients and servers.



Try it yourself: Consume a public API

Python has a module called **`requests`**) for sending requests and processing responses. It is very easy to use. Let's see an example where we want to consume an API to get random pictures of dogs.

In [1]:
```
import requests
import pandas as pd
import shutil
```

In [2]:
```
# GET request
url = 'https://dog.ceo/api/breeds/image/random'
requests.get(url)
```

Out[2]:
```
<Response [200]>
```

The code above sends a request using the **`GET` method**. There are other ways to send requests to a server such as: **`POST`**, **`PUT`**, and **`DELETE`**. Each of them returns a `Response` object, which can contain a **`status_code`**. This code give us information about what happened with our request.

Exercise 1:

Google the meaning of the 200 status code. What does it mean?

Answer. Code 200 means that the server status was the response "OK", that is, that the request succeeded.

Exercise 2:

What does the code 404 mean?

Answer. It means that the requested resource was not found.

Exercise 3:

A `Response` object contains the server's response to the HTTP request. Save the response of the `dog.ceo` request in a variable called `r`, call the property `status_code`, and print it.

Answer. Shown below.

In [3]:
```
r = requests.get(url)
print(r.status_code)
```

```
200
```

It is very common to receive the response in json format. By calling the `json()` we are able to process the date from our API call and output it to our desired format:

In [4]:
```
r = requests.get(url)
print(r.json())
```

```
{'message': 'https://images.dog.ceo/breeds/puggle/IMG_114654.jpg', 'status': 'success'}
```

Notice that we get a dictionary where the `message` key is the url of a picture of a dog. Now we are going to download the image and save it to a file:

In [5]:
```
# Printing just the URL
dog = r.json()
url_image = dog['message']
print(url_image)
```

```
https://images.dog.ceo/breeds/puggle/IMG_114654.jpg
```

In [6]:
```
# Saving the image to a file
r = requests.get(url_image, stream = True)

with open('dogo.jpg','wb') as f:
        shutil.copyfileobj(r.raw, f)
```

Exercise 4:

Consume an API to get random images of foxes. Use this url: http://randomfox.ca/floof to make your request. What is the link (URL) of the image?

Answer. A possible answer is shown below:

In [7]:
```
# GET request
r = requests.get('http://randomfox.ca/floof')
print(r.status_code)
```

```
200
```

In [8]:
```
# Printing the URL
fox = r.json()
print(fox["image"])
```

```
https://randomfox.ca/images/91.jpg
```

Consume APIs to load datasets

The Socrata Open Data API (**SODA**) is a consumer API that enables us to access thousands of open datasets available online. These datasets come in in a variety of formats, including JSON, which is very convenient for loading data sets as `Dataframe` objects. The following code shows how to get an **API endpoint** (a unique URL to the dataset). If you are looking for data on a specific topic or country, just go and search here and get the corresponding endpoint.

Exercise 5

Suppose we want to find a dataset about insurance in Texas. After searching in SODA, we found this Endpoint, which provides access to data about insurance complaints: https://data.texas.gov/resource/ubdr-4uff.json

Write a `GET` request and use the object response method `json()` to create a `pandas` `DataFrame` object. Then list the first ten rows.

Hint: To create a `DataFrame` object from a JSON file use the `pandas` method `from_dict()`.

Answer. Shown below:

In [9]:
```
# Defining the URL
url= 'https://data.texas.gov/resource/ubdr-4uff.json'

# Making the request
r = requests.get(url)
print("Status code:", r.status_code)

# Obtaining the JSON and converting it to a DataFrame
rjson = r.json()
df = pd.DataFrame.from_dict(rjson)
df.head(10)
```

```
Status code: 200
```

Out[9]:

	complaint_number	respondent_name	complainant_role	reason	complaint_confirmed_code	disposition	received_date	clo
0	1	METROPOLITAN LIFE INSURANCE COMPANY	Relative	Customer Service	No	Other	2012-06-12T00:00:00.000	2012-07-25T00:(
1	2	AETNA LIFE INSURANCE COMPANY	Provider	Delays (Claims Handling)	No	Information Furnished	2012-06-21T00:00:00.000	2012-08-01T00:(
2	3	BLUE CROSS AND BLUE SHIELD OF TEXAS, A DIVISIO...	Provider	Denial Of Claim	No	Other	2012-06-11T00:00:00.000	2012-07-30T00:(
3	4	BLUE CROSS AND BLUE SHIELD OF TEXAS, A DIVISIO...	Provider	Denial Of Claim	No	Other	2012-06-28T00:00:00.000	2012-07-30T00:(
4	5	CHARTER OAK FIRE INSURANCE COMPANY, THE	Insured	Unsatisfactory Settle/Offer	No	Contract Language/Legal Issue; Question of Fact	2012-06-13T00:00:00.000	2012-07-17T00:(
5	6	REASSURE AMERICA LIFE INSURANCE	Insured	Cash Value; Misrepresentation	No	No Jurisdiction	2012-06-18T00:00:00.000	2012-07-30T00:(

		COMPANY						
6	7	USAA GENERAL INDEMNITY COMPANY	Third Party	Unsatisfactory Settle/Offer	No	Claim Settled; Question of Fact	2012-02-27T00:00:00.000	2012-07-02T00:0
7	8	BLUE CROSS AND BLUE SHIELD OF TEXAS, A DIVISIO...	Insured	Denial Of Claim	No	Company Position Upheld	2012-06-19T00:00:00.000	2012-08-06T00:0
8	9	PROPERTY AND CASUALTY INSURANCE COMPANY OF HAR...	Insured	Unsatisfactory Settle/Offer	No	Complainant Retained Attorney; Question of Fac...	2011-10-03T00:00:00.000	2012-07-20T00:0
9	9	AMERICAS SERVICING CORPORATION	Insured	Unsatisfactory Settle/Offer	Yes	Failure to Timely Respond	2011-10-03T00:00:00.000	2012-07-20T00:0

Filters can be applied to endpoints directly using parameters. The following code illustrates how to filter by the `reason` column to obtain only `Denial Of Claim` reasons. Notice the use of ? and the parameter with its respective value:

In [10]:
```
url = 'https://data.texas.gov/resource/ubdr-4uff.json?reason=Denial Of Claim'
r = requests.get(url)
rjson = r.json()
filtered_data = pd.DataFrame.from_dict(rjson)
filtered_data
```

Out[10]:

	complaint_number	respondent_name	complainant_role	reason	complaint_confirmed_code	disposition	received_date	closed_dat
0	3	BLUE CROSS AND BLUE SHIELD OF TEXAS, A DIVISIO...	Provider	Denial Of Claim	No	Other	2012-06-11T00:00:00.000	2012-07-30T00:00:00.00
1	4	BLUE CROSS AND BLUE SHIELD OF TEXAS, A DIVISIO...	Provider	Denial Of Claim	No	Other	2012-06-28T00:00:00.000	2012-07-30T00:00:00.00
2	8	BLUE CROSS AND BLUE SHIELD OF TEXAS, A DIVISIO...	Insured	Denial Of Claim	No	Company Position Upheld	2012-06-19T00:00:00.000	2012-08-06T00:00:00.00
3	12	HEALTHSPRING LIFE & HEALTH INSURANCE COMPANY, ...	Relative	Denial Of Claim	No	Contract Language/Legal Issue	2012-06-18T00:00:00.000	2012-07-19T00:00:00.00
4	15	ACCEPTANCE CASUALTY INSURANCE COMPANY	Insured	Denial Of Claim	No	Question of Fact; Information Furnished; Addit...	2012-06-05T00:00:00.000	2012-07-27T00:00:00.00
...	...	...	...	...	...	...	...	...
995	7604	BLUE CROSS AND BLUE SHIELD OF TEXAS, A DIVISIO...	Provider	Denial Of Claim	No	No Jurisdiction	2012-09-11T00:00:00.000	2012-11-13T00:00:00.00
996	7611	TEXAS FARM BUREAU MUTUAL INSURANCE COMPANY	Insured	Denial Of Claim	No	Information Furnished; Question of Fact	2012-09-04T00:00:00.000	2012-11-05T00:00:00.00
997	7612	ACCC INSURANCE COMPANY	Third Party	Denial Of Claim	No	Contract Language/Legal Issue; Information Fur...	2012-09-04T00:00:00.000	2012-11-05T00:00:00.00
998	7615	NAUTILUS INSURANCE COMPANY	Insured	Denial Of Claim	No	NaN	2012-09-04T00:00:00.000	2013-01-18T00:00:00.00
999	7632	ALLSTATE FIRE AND CASUALTY INSURANCE COMPANY	Insured	Denial Of Claim	No	Information Furnished	2012-09-04T00:00:00.000	2012-11-06T00:00:00.00

1000 rows × 17 columns

Exercise 6:

Copy the previous code cell and repeat the request, but this time include only those records where `coverage_type` is `Automobile`.

In [11]:
```
url = 'https://data.texas.gov/resource/ubdr-4uff.json?coverage_type=Automobile'
r = requests.get(url)
rjson = r.json()
filtered_data = pd.DataFrame.from_dict(rjson)
filtered_data
```

Out[11]:

	complaint_number	respondent_name	complainant_role	reason	complaint_confirmed_code	disposition	received_date	clos
		CHARTER OAK				Contract		

0	5	FIRE INSURANCE COMPANY, THE	Insured	Unsatisfactory Settle/Offer	No	Language/Legal Issue; Question of Fact	2012-06-13T00:00:00.000	2012-07-17T00:0
1	7	USAA GENERAL INDEMNITY COMPANY	Third Party	Unsatisfactory Settle/Offer	No	Claim Settled; Question of Fact	2012-02-27T00:00:00.000	2012-07-02T00:0
2	10	ALLSTATE FIRE AND CASUALTY INSURANCE COMPANY	Attorney	Non-Disclosure Of Coverage	Yes	Information Furnished	2012-06-15T00:00:00.000	2012-09-26T00:0
3	13	OLD AMERICAN COUNTY MUTUAL FIRE INSURANCE COMPANY	Insured	Cancellation	No	Company Position Upheld; Policy Not In Force	2012-04-13T00:00:00.000	2012-07-20T00:0
4	13	ROMERO INSURANCE AGENCY, LLC	Insured	Cancellation	No	Information Furnished	2012-04-13T00:00:00.000	2012-07-20T00:0
...	...	...	...	...	...	...	...	...
995	3493	STATE FARM MUTUAL AUTOMOBILE INSURANCE COMPANY	Third Party	Delays (Claims Handling)	No	Complainant Retained Attorney; Information Fur...	2012-07-30T00:00:00.000	2012-09-20T00:0
996	3494	STATE FARM MUTUAL AUTOMOBILE INSURANCE COMPANY	Third Party	Delays (Claims Handling)	No	Claim Settled; Information Furnished	2012-07-30T00:00:00.000	2012-10-18T00:0
997	3496	OLD AMERICAN COUNTY MUTUAL FIRE INSURANCE COMPANY	Third Party	Delays (Claims Handling)	No	Claim Settled	2012-07-30T00:00:00.000	2012-10-10T00:0
998	3500	DRIVER'S INSURANCE COMPANY	Attorney	Delays (Claims Handling)	No	Claim Settled; Company Position Upheld	2012-07-27T00:00:00.000	2012-10-04T00:0
999	3502	OLD AMERICAN COUNTY MUTUAL FIRE INSURANCE COMPANY	Consumer (Non-Insured)	Delays (Claims Handling); Unsatisfactory Settl...	No	Contract Language/Legal Issue	2012-07-30T00:00:00.000	2012-10-24T00:0

1000 rows × 17 columns

Takeaways and Conclusion

In this case we learned about APIs as programs that allow for communication between other programs, and focused on Web APIs, which we can interact with by using the `requests` package. We also learned about status codes and how to retrieve information from public APIs.

Attribution

"Melitta Thermal coffee maker", Alan Levine, 28 August 2013, Creative Commons Attribution 2.0 Generic license, https://commons.wikimedia.org/wiki/File:Melitta_Thermal_coffee_maker.jpg

"The Process of Communication", Freedom Learning Group - Lumen Learning. Creative Commons Attribution,https://courses.lumenlearning.com/wm-organizationalbehavior/chapter/the-process-of-communication/

"REST-API", Seobility, Creative Commons Attribution Share-Alike, https://www.seobility.net/en/wiki/images/f/f1/Rest-API.png