ReST

- REST stands for **RE**presentational **S**tate **T**ransfer

- It simply means that you are requesting to transfer the state of a resource (more or less complex)

ReST

It relies on a 5 points constraint (the most important):

1. client-server

2. statelessness

Client-Server

- It is an architecture model, today is the most used
- It relies on the basic principle that the server has the resources, thus the client is the requester of one or more resource, so

  the client asks and the server responds

- Hence, the client does not need to know anything about the server implementation

Statelessness

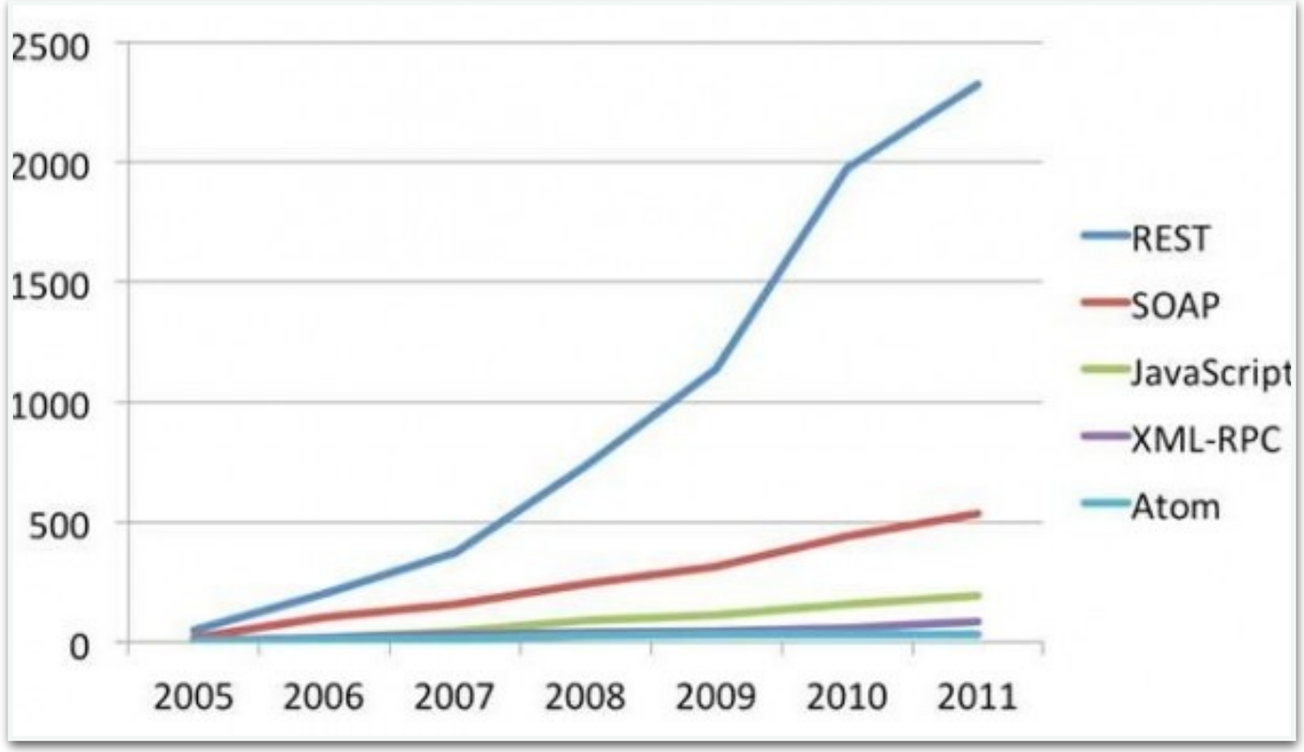
- It means that every request from the client to the server must contain all the needed information to be correctly processed and must be completely agnostic about the server context

- Every request must not rely on any preceding request

  Every request needs to know anything but itself

An alternative to ReST

ReST is not the only one architecture we can use to transfer data, the most popular is **SOAP**



SOAP with XML

- The most important disadvantage of SOAP is that it works with XML

- XML is a markup language used to represents entities but also transfer objects among the web:

```
<xml version="1.0" encoding="UTF-8">
<utenti>
 <utente>
<nome>Luca</nome>
<cognome>Cicci</cognome>
<indirizzo>Milano</indirizzo>
</utente>
<utente>
<nome>Max</nome>
<cognome>Rossi</cognome>
<indirizzo>Roma</indirizzo>
</utente>
</utenti>
```

SOAP with XML

- As you might have seen from the example it is quite clean and expressive, but I hope you have at least found a con of it :)

SOAP with XML

- What happens if we need to model a lot more complex entity than a man/women?

- Consider you need to model a department or an office with all its customers, employees, etc, etc : ?

SOAP with XML

- This is the greatest problem with XML, it has a lot of expressiveness but it is **too verbose** to be used to model complex entities with relationships among them

SOAP with XML

- Moreover you are not modelling your entities to keep them for yourself but to share it with clients requesting it!!!

- It means a lot of bandwidth to transfer all that kind of information from a server to the client/s

Is there an alternative?

Obviously, there is always an alternative

JSON

- What is JSON?

- Some example

JSON

- JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent.

JSON

```
{
                "employees":[
    {
      "firstName":"John",
      "lastName":"Doe"
    },
    {
      "firstName":"Anna",
      "lastName":"Smith"
    },
    {
      "firstName":"Peter",
      "lastName":"Jones"
    }
  ]
}
```

JSON vs XML

- Same example with XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<employees>
  <employee>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName>
    <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName>
    <lastName>Jones</lastName>
  </employee>
</employees>
```

HTTP + ReST + JSON = magic

- ReST architecture is put on top of HTTP
- Being on top of HTTP it can use all its verbs:

1. POST
2. GET
3. PUT
4. DELETE

HTTP + ReST + JSON = magic

- To those basic verbs you can link basic db actions, in fact:

1. POST -> update or create a resource
2. GET -> get resource/s (list of students)
3. PUT -> update or create a resource
4. DELETE -> I hope you don't need anything for this :)

HTTP + ReST + JSON = magic

- ReST is on top of HTTP so will you have url to access resources? Of course

1. POST -> POST /users/
2. GET -> GET /users/<id> or simply /users/
3. PUT -> PUT /users/<id>
4. DELETE -> DELETE /users/<id>

HTTP + ReST + JSON = magic

- What is the difference between PUT and POST?

Django ReST Framework

- In the previous lesson you learnt (I hope), the very basic principles of Django. You are now able to configure and run a simple web app

- Let's say you now want to add some new feature, for example you want to develop a mobile app (Android or whatever you like) to interact with the web app

Django ReST Framework

- How can you allow
- Let's say you now want to add some new feature, for example you want to develop a mobile app (Android or whatever you like) to interact with the web app

Django ReST Framework

- It sounds like we have to plug in a ReST service to handle the basic ReST/HTTP method

Django ReST Framework

```
pip install djangorestframework
pip install markdown        # Markdown support for the browsable API.
pip install django-filter   # Filtering support
```