SCOM / SRSI
Anatomy of a Web Service

2020-21

Ana Aguiar

DEEC, FEUP

Motivation

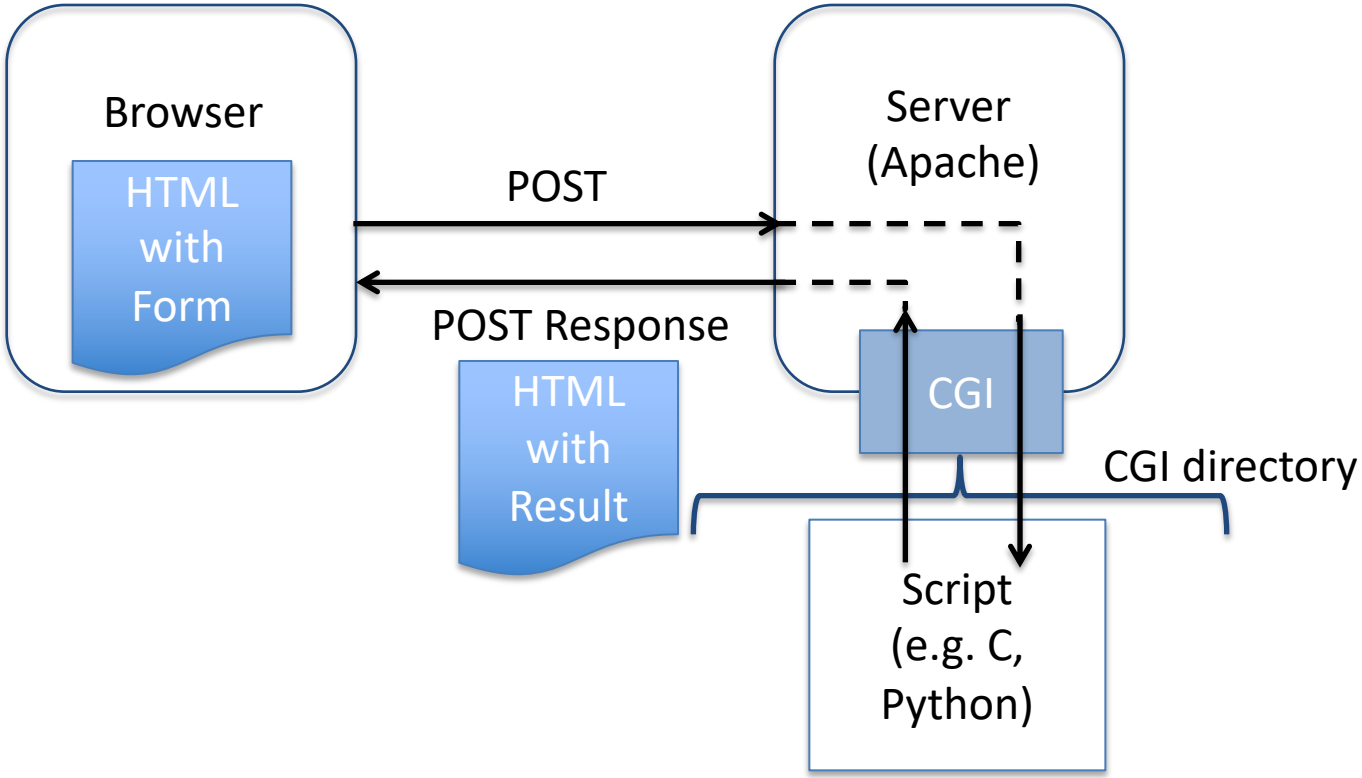
- Service oriented architecture
- Monolithic vs micro-service architecture
- Apart from religious beliefs, almost any application is built on top of more than one service nowadays

- This project shows you the inside of this architecture and its implications

Goals

1. Create a web service that is called by a webpage
   - Take a number n and compute the sum or the product of 1,…,n.
     - Or any other simple computation
   - Write a small program in python or C to do the calculation
   - Use a web page to demonstrate the web service
2. Measure the response time of your service

Understanding what happens



Designing the Web Service

- Which parameters do you need?
- Which technologies are you going to use?
  - In the client
  - For the script on the server
  - For the communication between the two

Designing the Web Service

- Which parameters do you need?

- Which technologies are you going to use?
  - In the client
    - HTML
  - For the script
    - Python or C
  - For the communication between the two
    - Plain HTTP methods

How is the Web Service invoked?

- Create an HTML page to call the script using the HTTP POST method
  - HTTP Form: http://www.w3schools.com/tags/att_form_method.asp
  - Use the loopback interface for development
    - Place your html file in /var/www
    - Access it using http://[yourwebpage.html]
    - Submit the HTML form
- Verify using wireshark
  - Take care to capture on the lo interface, not eth0
  - Observe the GET and POST requests and responses
  - What is the content of the response bodys?
  - Verify that the body of the POST request sends the values you are selecting in the form

Creating the Actual Service

- Create a python script
  - Place it the default location for scripts and configure the server to enable CGI scripts
    - See directions below
  - Start by returning a web page with the parameters received
    - How can you retrieve the parameters inside the script?
  - Then, add the requested service logic
- Configure the web server to run CGI scripts
  - http://httpd.apache.org/docs/current/howto/cgi.html
  - Check the ScriptAlias directive in /etc/apache2/conf-enabled/serve-cgi-bin.conf for the directory where to put your script
  - Enable CGI scripts by running *a2enmod cgi*
  - Add the python extension to the server

Measuring Response Time

- Measure the response time of your service
- Create a client that makes a sequence of N calls to the service
  - N should be a command line parameter
  - Because we want to measure system parameters, the language you choose for the measuring program should have be as close to the kernel as possible
  - Choose well the functions you use to measure time
  - Choose well where you call the time measuring functions

Measurements

- Measure the time to obtain the reply
  - Since you are measuring, you need several samples, and should present results as average and confidence interval
    - https://en.wikipedia.org/wiki/Confidence_interval
    - https://www.khanacademy.org/math/statistics-probability/confidence-intervals-one-sample/estimating-population-mean/v/confidence-interval-1
- Choose the amount of samples in an experiment
  - Keep making new experiments until a confidence interval of 5% is reached
  - If you use another way to decide, justify it
- Anything you do, you should be able to explain and justify your options

Experimental Setup

- You will receive an IP address and access credentials to a virtual machine in a commercial cloud
- Configure apache and your script to work there
- Perform the response time measurements from a local client
  - You should be able to draw a diagram of your experimental setup precisely
- Carry out measurements at different times of the day: 9am, 12pm, 8pm, 12 am
- Compare the results with the RTT obtained with ping

Evaluation

- Evaluation will be an individual 30’ quizz
  - Online
  - In class on Oct 12th
  - Asking about details of how you carried out the assignment and experiments