TOPIC 2: WEB PROGRAMMING

T2.1) Explanation of Web programming

Web programming refers to the writing, markup and coding involved in Web development, which includes Web content, Web client and server scripting and network security. Or Writing the necessary source code to create a website

Web development refers to building, creating, and an maintaining websites. It includes aspects such as web design, web publishing, web programming, and database management.

T2.2) Approaches to web programming

Web programming can be briefly categorized into client and server coding. The client side needs programming related to accessing data from users and providing information. It also needs to ensure there are enough plug ins to enrich user experience in a graphic user interface, including security measures.

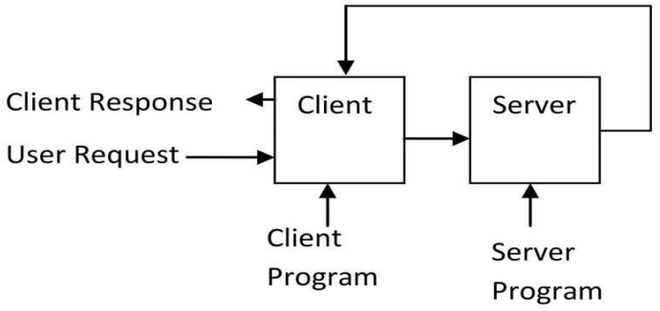
Web development is all about communication. In this case, communication between two (2) parties, over the HTTP protocol:

- The **Server** - This party is responsible for **serving** pages.
- The **Client** - This party *requests* pages from the **Server**, and displays them to the user. In most cases, the client is a **web browser**.
    - The **User** - The user *uses* the **Client** in order to surf the web, fill in forms, watch videos online, etc.

Each side's programming, refers to code which runs at the specific machine, the server's or the client's.

Basic Example

1. The **User** opens his web browser (the **Client**).
2. The **User** browses to http://google.com.
3. The **Client** (on the behalf of the **User**), sends a request to http://google.com (the **Server**), for their home page.
4. The **Server** then acknowledges the request, and replies the client with some meta-data (called *headers*), followed by the page's source.
5. The **Client** then receives the page's source, and *renders* it into a human viewable website.
6. The **User** types `Stack Overflow` into the search bar, and presses `Enter`
7. The **Client** submits that data to the **Server**.
8. The **Server** processes that data, and replies with a page matching the search results.
9. The **Client**, once again, renders that page for the **User** to view.



Server-side Programming

Server-side programming, is the general name for the kinds of programs which are run on the **Server**.

Uses

- Process user input.
- Display pages.
- Structure web applications.
- Interact with permanent storage (SQL, files).

Example Languages

- PHP
- ASP.Net in C#, C++, or Visual Basic.
- Nearly any language (C++, C#, Java). These were not designed specifically for the task, but are now often used for application-level web services.

Client-side programming

Much like the server-side, Client-side programming is the name for all of the programs which are run on the **Client**.

Uses

- Make interactive webpages.
- Make stuff happen dynamically on the web page.
- Interact with temporary storage, and local storage (Cookies, localStorage).
- Send requests to the server, and retrieve data from it.
- Provide a remote service for client-side applications, such as software registration, content delivery, or remote multi-player gaming.

Example languages

- JavaScript (primarily)
- HTML*

- CSS*
- Any language running on a client device that interacts with a remote service is a client-side language.

T2.3) Web programming languages

There are several languages that can be used for server-side programming:

- PHP
- ASP.NET (C# OR Visual Basic)
- C++
- Java and JSP
- Python
- Ruby on Rails and so on.

There are many client-side scripting languages too.

- JavaScript
- VBScript
- HTML (Structure)
- CSS (Designing)
- AJAX
- jQuery etc.

(Some other languages also can be used on the basis of the modeling/designing /graphics/animations and for extra functionalities.)

T2.4) Criteria for choosing a web programming language

How to Choose the Right Programming Language for a Web App

What really motivates people to go with one programming language or another when they start product development? Actually, it's not an easy choice and it's especially difficult for non-technical people who usually have to rely on their partners and advisors in this. This choice might even define the success of your business, so it's definitely worth weighing up all the pros and cons of each option.

Criteria

First, you should consider the following criteria:

- Efficiency of the language itself;
- Available platforms and frameworks;
- Community support;

- How difficult the language is to learn. Mind you, if the language syntax and rules are complex, you'll get more capable developers as only the best ones can get through. On the other hand, there might be the scarcity of developers, which we see with the languages like Scala, Erlang, Lisp, Prolog, Golang and others.
- Strategic issues: which tech stack your potential acquisition company is using, or which technology stack they would prefer to have.

Factors to consider when choosing a programming language

1. **Popularity**. This is a very important one. A good place to start is the Tiobe index. You are more likely to find people to collaborate with if you use a popular language. You are also more likely to find reference material and other help. Unfortunately, the most popular language globally may not be a good match for your problem domain.
2. **Language-domain match**. Choose one that matches your problem domain. You can do this by looking at what other people in your field are using (after adjusting for popularity, so don't think the match with Java is good simply because a lot of people are using Java) or by looking at some code that solves problems you are likely to have and seeing how natural the mapping is**.**
3. **Availability of libraries**. Some would argue that this is the same as the point above, but I don't think so. If there's a library that solves your problem well, you'll put up with some ugly calling conventions or hassle in the language.
4. **Efficiency**. Languages aren't fast - compilers are efficient. Look at the efficiency of compilers or interpreters for your language. Be aware that interpreted code will run an order of magnitude slower than compiled code as a rule of thumb.
5. **Expressiveness**. The number of lines of code you create per hour is not a strong function of language, so favour languages that are expressive or powerful
6. **Project-size**. Do you want to be programming in the large or programming in the small? Choose a language that supports your use case.
7. **Tool support**. Popularity usually buys tool support (and some languages are easier to write tools for). If you are a tool-oriented user, choose a language with good tool support. Just read this article on tool mavens vs language mavens before you make a choice.

PHP

It's hard to argue against the popularity of PHP, the scripting language which is used by many and has a vibrant developer community. It is used not only for WordPress development but also for complex systems so even Facebook utilizes PHP, which says a lot about its level of credibility.

Since PHP does not use too much of a system's resources in order to run, it operates quite well compared with other scripting languages. Hosting it is also very easy as lots of hosts provide support for PHP.

Pros of PHP:

- It is highly accessible, there are a lot of frameworks written in it like Symfony, Yii, Laravel, CakePHP, Kohana, Zend Framework, PhalconPHP, to name a few.
- It enables fast implementation of complex solutions. And the faster a new application enters the market, the higher is your cost-efficiency and the greater is your competitive advantage.
- Thanks to the popularity of PHP, it offers great flexibility during and after the initial project, as the number of resources is continuously growing.
- It has sound host support.
- PHP can be embedded in HTML.
- PHP7 was released in December 2015 ahead of deadlines and its performance has grown substantially. Totally different architecture sits in its kernel, so it changes the way how the interpreted code is being processed on operation system level, and this actually causes this increase in performance.
- There are performance accelerators available for PHP. For example, Facebook came up with HipHop, a compiler of PHP into C which then turned into executable files. Eventually, HipHop project came up with a virtual machine transforming PHP script into bytecode and doing its own internal performance optimization. There are also such solutions as APC, ionCube, Turck MMCache, Zend Opcache, XCache etc.
- Websites powered by PHP: Facebook, WordPress, Wikipedia, Mailchimp, Flickr, Yahoo!, Tumblr etc.

Cons of PHP

- When someone claims that they know PHP, it's hard to assess their level of development skill, so you'll need to dig deeper. PHP is so easy to learn on the basic level that people without proper computer science background could go and start developing, not knowing much about algorithms, time complexity, application performance optimization, advanced database querying, systems scalability and other important aspects.
- It's not as cross-platform as some of its competitors. The language is compatible with UNIX based OS and Windows OS.

Python

Pros of Python

- It requires less time to develop than, for instance, in Java, as programs are shorter. So faster development and also deeper prototyping would give you a competitive advantage.
- Python has a quite simple syntax and easy-to-use data structures.
- It performs well across different platforms.
- The language has good scalability.
- Apps powered or supported by Python include Instagram, Pinterest, Django, Google, NASA, Yahoo, etc. You can also check out more organizations using Python.

Cons of Python

- Performance could be better. Programs in Python are slower than in Java, for example. This is obviously because it is one of the interpreted languages, which are normally slower than compiled languages.

Ruby

According to Stack Overflow 2015 Developer Survey, Ruby is the technology which pays best in Western Europe, which says a lot about its adoption scale.

Pros of Ruby

- Ruby prefers convention over configuration which makes applications easier to develop and understand. Clear syntax makes the programming process much faster and more efficient.
- RoR framework has very simple structure, it's easy for developers.
- Don't Repeat Yourself, or DRY Principle. By not writing the same information over and over again, the code is more maintainable, more extensible, and less buggy.
- You can develop MVP very fast on Ruby, it takes less time than on other languages.
- It has big development community.
- Testing plays a big role in development. Minitest library goes together with Ruby, so when you develop something, it's ready for testing. As a result, you get a neat secure product. There's also a very powerful testing framework for ROR called RSpec.
- Platforms that are supported by Ruby include Hulu, Shopify, Twitter, Scribd, Crunchbase, Groupon, etc.
- There are many libraries, gems for any case, so when you are writing the code, you can use a ready-made solution, so you don't need to write it from scratch. The main repository contains over 100,000 gems. https://rubygems.org/stats

Cons of Ruby

- The performance is not as fast as on PHP or JavaScript.
- As it's the technology that pays really well, development might turn out to be more expensive than in other languages.

Java

Java Pros

- The core value proposition of the Java platform says "Write once, run anywhere". So the most important promise of Java technology is that you only have to write your application once and then you'll be able to run it anywhere. JVM is a universal engine, you can use it for anything, it will work everywhere. It's the most cross-platform language.
- It was arguably built with security in mind, so security features are one of Java's advantages.
- It has a big and dedicated development community.
- Outstanding performance.

Java Cons

- Bigger projects can be difficult to compile and build.

- Development is more expensive than in PHP or Python.

If you are still sitting on the fence with your choice, go through this fun flowchart and find out which language is better for your product And remember, you just need to find a good team with talented developers. Then no matter which programming language you go for, you can be confident in the final result.

T2.5) Common web programming interfaces

Application program interface (API)

An ***a**pplication **p**rogram **i**nterface* (**API**) is a set of routines, protocols, and tools for building software applications. Basically, an API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together.

A **web API** is an application programming interface (API) for either a web server or a web browser. It is a web development concept, usually limited to a web application's client-side (including any web frameworks being used), and thus usually does not include web server or browser implementation details such as SAPIs or web browser engine APIs unless publicly accessible by a remote web application.

A **web framework** (**WF**) or **web application framework** (**WAF**) is a software framework that is designed to support the development of web applications including web services, web resources, and web APIs.
Server Application Programming Interface (**SAPI**) is the direct module interface to web servers such as the Apache HTTP Server, Microsoft IIS, and Oracle iPlanet Web Server.

CGI - Common Gateway Interface

CGI is the abbreviation of ***C**ommon **G**ateway **I**nterface*. It is a specification for transferring information between a World Wide Web server and a CGI program. A CGI program is any program designed to accept and return data that conforms to the CGI specification. The program could be written in any programming language, including C, Perl, Java, or Visual Basic.

What is CGI?

- The Common Gateway Interface, or CGI, is a set of standards that define how information is exchanged between the web server and a custom script.
- The CGI specs are currently maintained by the NCSA and NCSA defines CGI is as follows −
- The Common Gateway Interface, or CGI, is a standard for external gateway programs to interface with information servers such as HTTP servers.
- The current version is CGI/1.1 and CGI/1.2 is under progress.

CGI Programs

CGI programs are the most common way for Web servers to interact dynamically with users. Many HTML pages that contain forms, for example, use a CGI program to process the form's data once it's submitted. Another increasingly common way to provide dynamic feedback for Web users is to include scripts or programs that run on the user's machine rather than the Web server. These programs can be Java applets, Java scripts, or ActiveX controls. These technologies are known collectively as *client-side* solutions, while the use of CGI is a *server-side* solution because the processing occurs on the Web server.

One problem with CGI is that each time a CGI script is executed, a new process is started. For busy websites, this can slow down the server noticeably. A more efficient solution, but one that it is also more difficult to implement, is to use the server's API, such as ISAPI or NSAPI. Another increasingly popular solution is to use Java servlets.

Web Browsing

To understand the concept of CGI, let's see what happens when we click a hyperlink to browse a particular web page or URL.

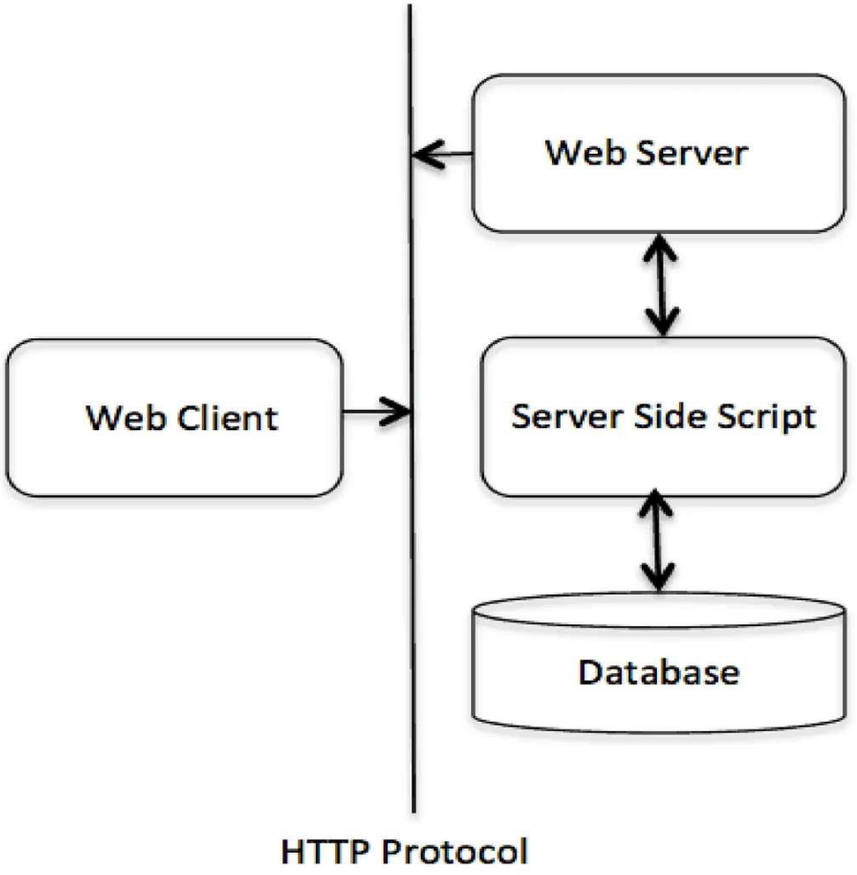
- Your browser contacts the HTTP web server and demand for the URL ie. filename.
- Web Server will parse the URL and will look for the filename. If it finds requested file then web server sends that file back to the browser otherwise sends an error message indicating that you have requested a wrong file.
- Web browser takes response from web server and displays either the received file or error message based on the received response.

However, it is possible to set up the HTTP server in such a way that whenever a file in a certain directory is requested, that file is not sent back; instead it is executed as a program, and produced output from the program is sent back to your browser to display.

The Common Gateway Interface (CGI) is a standard protocol for enabling applications (called CGI programs or CGI scripts) to interact with Web servers and with clients. These CGI programs can be a written in Python, PERL, Shell, C or C++ etc.

CGI Architecture Diagram

The following simple program shows a simple architecture of CGI −



Web Server Configuration

Before you proceed with CGI Programming, make sure that your Web Server supports CGI and it is configured to handle CGI Programs. All the CGI Programs to be executed by the HTTP server are kept in a pre-configured directory. This directory is called CGI directory and by convention it is named as /var/www/cgi-bin. By convention CGI files will have extension as **.cgi**, though they are C++ executable.

By default, Apache Web Server is configured to run CGI programs in /var/www/cgi-bin. If you want to specify any other directory to run your CGI scripts, you can modify the following section in the httpd.conf file −

```
<Directory "/var/www/cgi-bin">
   AllowOverride None
   Options ExecCGI
   Order allow,deny
   Allow from all
</Directory>

<Directory "/var/www/cgi-bin">
   Options All
</Directory>
```

Here, I assume that you have Web Server up and running successfully and you are able to run any other CGI program like Perl or Shell etc.

Common Client Interface (CCI)

The Common Client Interface (CCI) of the Connection Architecture provides a standard interface that allows developers to communicate with any number of Enterprise Information Systems (EISs) through their specific resource adapters, using a generic programming style. The CCI defines a set of interfaces and classes whose methods allow a client to perform typical data access operations.