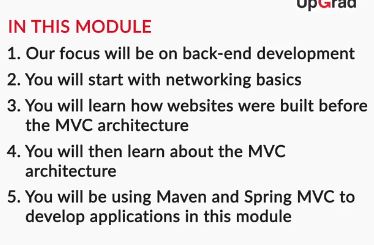
**Module Overview**

Welcome to your first module on application development. This module onwards, you will learn the skills required to work on the back-end development of web applications. Now, let's hear from our SME Thribhuvan Krishnamurthy about what you will be learning through this module and this course.



**Note:** Irrespective of the modules you have covered thus far, this module will involve a more practical approach of working on projects and debugging errors. Most of the errors you may come across are very common with other developers as well and must have already been discussed in a forum, so it is easy to find a resolution by searching for the same on the web. Also, we strongly encourage you to discuss your doubts and errors/debugs on the discussion forum, which will help other students in your batch. We will also be showing you how you can use the debug tool from IntelliJ.

It is very important for you to go through the text and codes along with the videos to completely understand this module.

## In this session

* You will get a basic understanding of certain prerequisites to this module such as HTML, Client-Server architecture, Network terminology. **However, you will learn HTML in detail in course 6.**
* You will also create a simple 'Hello world' website using some approaches less preferred than the MVC architecture.
* You will learn the importance and requirement of MVC architecture, and build a simple ‘Hello world’ website using Spring MVC.

In the upcoming sessions, you will learn to develop a simple blogging web application based on the MVC architecture.

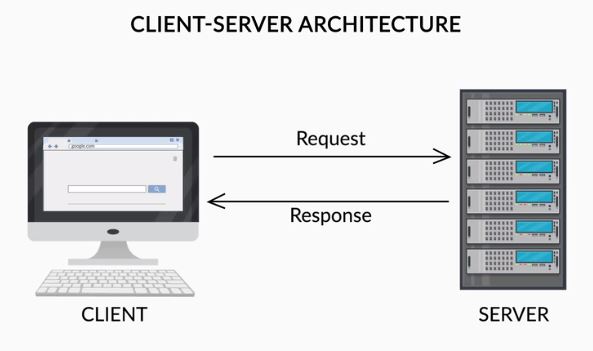
**Client-Server Architecture**

The internet hosts a plethora of services that you can access. You end up using the internet for many different types of activities in a day. You wake up and check your email, greet your friends and relatives using some instant messenger, access your calendar and check your schedule for the day. You may also read some blogs and articles to begin your day or back up your precious pictures from the party the day before on some cloud storage.

In the above-mentioned activities, two common entities always exist for the functionalities to be performed:

* You, trying to access some service via the internet
* Some remote computer, device or organisation providing you with the service you need

In the following video, let's take a deeper look at this setup, which is called the client-server architecture:



Q> **Client-Server Architecture**

Can you think of any real-life scenario that matches the client-server architecture where the services are provided without the use of the internet?

**Suggested Answer**

One of the real-life example that match with the client-server architecture is banking services. Here, **the bank acts as a server and the customers of the bank are clients**. Based on the client request for a specific service, he/she can avail the service at the bank.   
In this way, you may consider many other services which are similar to the client-server architecture, for example, the postal service.

You saw that the client and the server are programs running on machines that interact with each other to provide you facilities on the internet such as rendering a website on a browser.

If you are accessing a website then:

* A software trying to access some service over the internet is called the client. You will type the URL of the website on the browser's search bar. Hence, in this case, the client program will be your browser since clients are programs that request data from a server.
* Similarly, a software providing a service or data over the internet is called a server. In this case, the server will be a web-server that provides you with a website.

With this information, try to answer the following questions:

Q. **Client-Server Architecture**

Imagine that you are uploading pictures to an image hosting website like Instagram. In this case, what would be the client and what would be the server?

**Suggested Answer**

Client: The website that you are using to send the pictures to the image hosting website (the Instagram website).  
Server: The server program that is storing and hosting all the uploaded images.

**Client-Server Architecture**

Can you think of examples where a client (for example, your web browser) is sending requests to a server (a web server or a cloud storage server like Google Drive)?

**Suggested Answer**

When you type a URL in the search bar of your browser, your browser will send a  request to the web-server of that website. The web-server will respond to the request made by the browser and then send the data representing the webpage that you are trying to access back to your browser,  resulting in the webpage finally being displayed in your browser window.

Even when you try to log in to Facebook or search for something on Google, a request is sent from your browser (the client) to the respective Facebook or Google web server, which then validates the request and sends back the corresponding response to your browser.

With this, you must be clear about the fact that these two entities, the client and the server, interact to finally provide you with some functionality/service over the internet.

That said, there might be some questions in your mind.

The client and the server can be located miles apart and connected to the internet, but how do they find the right device to contact? How do they know the specific program to contact on the device they are trying to contact?

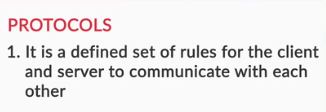
Clearly, there would be multiple processes, entities and programs involved to make this possible. In the next segment, we will briefly introduce you to some network-related concepts that will help you when you try to create a web application of your own.

**Networking Terminologies**

You already learned that the client and the server communicate to provide you with facilities such as rendering the website services on a browser. In order to obtain a basic understanding of how the client and server communicate with each other, you need to have a brief overview of the following concepts:

* Protocols
* Ports
* IP addresses
* DNS servers







**Protocols**

In your own words, explain the need for protocols.

**Suggested Answer**

Since client and server are at the end software programs, there needs to be a defined set of rules and format to be adhered to while communicating and exchanging data. Protocols make sure that the server program understands what the client program is trying to say and vice versa. It makes sure that the millions of clients and servers over the internet communicate in a language that is understood by every other entity over the internet too.

**Protocols**

You already learned that a protocol is defined set of rules to be followed by the client and the server while communicating over the internet. Consider that you have to send a letter via the postal service. Can you list some rules that would have to be followed?

**Suggested Answer**

Here are some rules that would have to be followed:

* All the post offices, logistics managers and customers will have to adhere to the same set of rules.
* You have to get a postcard or write the letter in an acceptable format.
* You have to mention the correct receiver’s address and your own addresses at the appropriate places. You have to make sure that these are understandable by the postal service.
* You have to make sure that the language in which the letter is written is understandable by the receiver.
* You have to make sure that the message is contained within the space provided in the postcard.
* It has to be transported through post offices in a certain defined mechanism
  1. **IP addresses**

Which of the following devices can NOT have an IP address when connected to a network?

**CCTV Camera**

**Feedback :**

*CCTV cameras are often wirelessly connected to a central control system .Each CCTV camera would be a unique  entity in the network.*

**Tablet**

****

**Smartwatch**

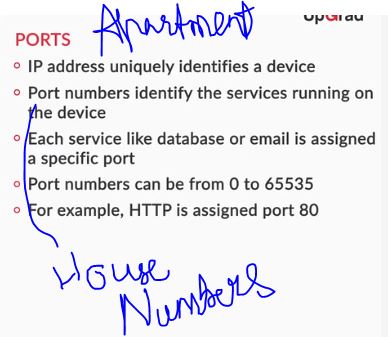
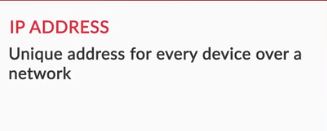
****

**None of these**

**Feedback :**

**All of the listed devices would have IP addresses when connected to a network.**

**Correct**

****

Q. **Ports**

Multiple services can run on the same device like web services or database services. Every device has one unique address called the IP address. Describe in your own words, how does having port numbers help in accessing different services on the same device with a unique IP address.

**Suggested Answer**

Ports uniquely identify the different kinds of services running on a computer.

Consider the following analogy discussed by the SME:

* IP addresses are similar to building addresses. But different people can live in the same building and would need to be identified uniquely
* This is where apartment numbers come into the picture, they identify different people living in the same building. Similarly, port numbers identify different services running at the same IP address.



You were introduced to many new terms in the previous video. Clearly, there is a lot in play to make the internet function. Let's recall the concepts that we used.

We started with learning about **protocols** first. In general, a **protocol** is a defined set of rules to be followed for smooth communication of a process. Similarly, there are a certain set of protocols used by the client and the server to ensure smooth communication of various processes such as sending data, web pages or email. Protocols can also be said to be the communication language over the internet. They define the way in which information has to be communicated over the internet and are uniform for the millions of clients and servers connected to the internet and make sure that they understand communication coming from each other.

There exist different protocols to implement different services. Let's recall the example of sending a letter to someone.

* All the post offices, logistics managers and customers will have to adhere to the same set of rules.
* You have to get a postcard or write a letter in an acceptable format.
* You have to mention the correct receiver’s address and your own addresses in the appropriate places, and make sure that these can be understood by the postal services.
* You have to make sure that the language in which the letter is written is understood by the receiver.
* You have to make sure that the message is contained within the space provided in the postcard.
* The letter has to be transported through post offices in a defined mechanism.

Similarly, when you want to access any website, you would be contacting a web server. The web server and the client have to adhere to the Hypertext Transfer Protocol (HTTP) to communicate in this case. HTTP defines a uniform way or language through which every client and web-server over the internet should communicate. You only need to be concerned with HTTP for now. You will learn in detail about communication through HTTP in the next segment.

Different services have different protocols, such as email services, instant messaging or database services. You can go [here](http://www.pearsonitcertification.com/articles/article.aspx?p=1868080) to view the list of protocols which you may encounter over the internet.

The next concept you learned about was **IP addresses**.

Now the question must have come to your mind that when so many devices are connected over the internet, how do the specific client and server identify and interact with each other? How do they communicate with each other following a specific protocol?

Each device connected over the internet or a network following the TCP/IP protocol is provided with a specific address called the IP (Internet protocol) address. This IP address will help to uniquely identify devices and help those devices communicate with each other.

IP address generally looks like this: **192.168.3.1.**You can go [here](https://whatismyipaddress.com/ip-address) to learn more about IP addresses.

After that, we moved on to the very important and intriguing concept of **Ports**.

Many services can be hosted on the same server (i.e., a web server, an email server, and a file server can be hosted on the same computer).  You might be trying to browse Facebook, send an email, and your device may receive different responses from different servers at the same time.

If a computer only has one IP address, how do we differentiate the services on the computer (i.e., would all the services share the same IP address)?

You already saw the example of every person in the same building having a different address. This is achieved by assigning different apartment numbers.

Similarly, the replies from the server will be addressed to a different door or 'port' in the same client device having a single unique IP address.

If the server is sending an email, it will go through the door assigned to send or receive emails.  
If the server is sending a file, it will go through the door assigned to receive files. These doors are known as ports.

Every different service is assigned a different port. You can go ahead and take a look at the list of ports assigned to different protocols or services [here](https://www.utilizewindows.com/list-of-common-network-port-numbers/).

Requests adhering to or in the HTTP format have to go through the assigned port, which is port 80. Hence, all web services are accessed via port 80 of the web server.

Thus far, you are clear with the following steps:

* Over the internet, different servers provide different facilities, and you use different client programs to access those facilities.
* The client and the server interact via communication rules called protocols.
* The client and the server find each other through assigned addresses called IP addresses.
* Every different type of client program accessing a different service uses a port or a door assigned to that protocol or service.

All this gives rise to a big question. When you go to your browser to access a website, and you type the URL [www.facebook.com](http://www.facebook.com/), how does it find the appropriate server for you?

Clearly, there is some mechanism in play which tells the client the IP address of the server to be contacted. This happens via another server between the client and the server called the **DNS server**.

DNS servers work like phone books. You have the domain name, which is basically the name of a website (like Facebook or Google).

Through the DNS server, you can go and look for the 'phone number' of a website, which is basically its IP address. In simple terms, the DNS server has a table with the IP addresses of different domain names, where the domain name is the actual name of the website that you are trying to access.

A very important thing to note is that domain names are unique. There cannot be another [www.facebook.com](http://www.facebook.com/) over the internet. One has to go and purchase a domain name which is available. Even if a website is not up and running on the internet, but the domain name is already purchased by someone, you cannot use that domain name.

The focus of this module is web services and web application. This means that the protocol we should be concerned the most is HTTP. You will learn some important concepts about HTTP in the upcoming segment.

**Hypertext Transfer Protocol (HTTP)**

You already know that the web server and the client communicate through HTTP. When you build a web application further in this module, you will also be using HTTP to:

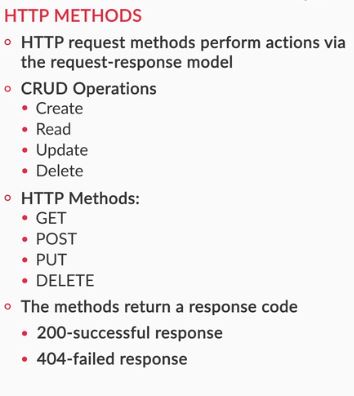
* Send requests from a client to a server
* Send a response from a server to a client

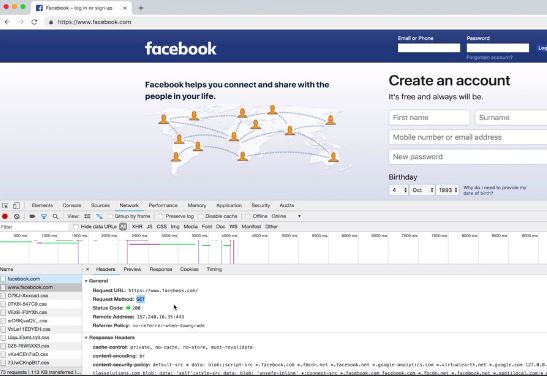
HTTP stands for Hypertext Transfer Protocol. You will learn about hypertext shortly in this segment. HTTP works in a very simple way:

1. The client sends a request to the server.
2. The server processes the client’s request.
3. The server then sends a response to the client as a reply to the client’s request.

There can be different types of client requests based on the required server operations. These client requests are sent using something called 'request methods'.

In this video, our SME will brief you on the different request operations and their corresponding request methods in HTTP.





You have learnt that different HTTP request methods can be used to  perform CRUD operations as follows:

* POST - This HTTP method is used to send a request to create new data on the server.
* GET - This HTTP method is used to read or retrieve some existing data from the server.
* PUT - This HTTP method is used to update some existing data in the server.
* DELETE - This HTTP method is used to delete some existing data from the server.

As you have observed in the video, HTTP is all around you, even when you opened this webpage to read the module, your browser used the HTTP GET method to request this webpage from the UpGrad web server.

You can use **'ctrl+shift+I'** to open the developer console in Mozilla Firefox and Google Chrome. You can open the developers’ console on this webpage right now and see HTTP request methods in play when you try to watch a video or submit an answer to a question

Q. **HTTP Methods**

Consider that you are responsible for creating the UpGrad website that you can see right now. Which of the following methods would you use to send data to the server when a student clicks on the submit button after answering a question?

**POST**

**Feedback :**

*POST is used to send data to the server. When you submit an answer to this question, a POST request is sent.*

**HTTP Protocol**

Which of the following services would use the HTTP protocol for client-server communication?

**A social media website like Facebook.**

**Feedback :**

Sending data over the internet requires HTTP protocol and in this case, a user is accessing a service via a website.

**Correct**



**A cloud storage service like Google Drive accessed via a website.**

**Feedback :**

Sending data over the internet requires HTTP protocol and in this case, a user is accessing a service via a website.

**Correct**



**A video streaming website like Youtube.**

**Feedback :**

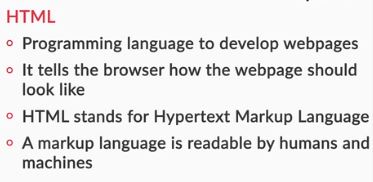
Sending data over the internet requires HTTP protocol and in this case, a user is accessing a service via a website.

# Hypertext Markup Language (HTML)

Like every other action, you have to write a program to tell your computer how a website should be displayed. Every website, including the one you can see right now, is coded in a language called HTML.

HTML is a language that can be used to tell a web browser how the content on the web page is to be displayed. It gives the browser instructions such as some text should be displayed in bold, some text is to be displayed as a link, etc.

Let's now get an understanding of what HTML does and how it creates a web page.



HTML stands for Hypertext Markup Language.   
Hypertext simply stands for text that links to other web pages, as shown in the text below:  
[Click here.](https://www.google.com/)

When you clicked on the text 'Click here', another website opened. Such text is known as hypertext.

Through HTML tags, you "Markup" the elements of a web page. You communicate the details of the webpage or mark them up through HTML tags. Most of these tags have opening and closing tags.

Developing an HTML code is not the focus of this module. Here, we will look at how to develop on server-side Java code. Hence, you need not worry about being proficient at HTML through this course; you will get a much better understanding of the same in upcoming courses.

Let's revise the steps for creating the Hello World web page:  
**Step 1:** Firstly, open a new file of your favourite text editor. (Notepad, Sublime, or any other text editor you prefer)  
   
**Step 2:** Type <!DOCTYPE html> in your text editor. This tag indicates that the file is an HTML file and can be read by a browser.

In the next two lines, type <html></html>. This </html> tag indicates the end of your HTML program. Everything you want to tell your browser about your web page comes within the <html> and </html> tags.

**Step 3:** Within the <html> and </html> tags, add <head></head>, and within the <head></head> tags, add <title> </title>. Write the title of your web page, which will be shown on the title bar of your browser. The head tags indicate that they contain metadata about the web page. Metadata stands for “data about data”. Here, metadata about our website would be the title or the format in which your website text is encoded. This is information that should not be displayed on the main page in the browser window.

**Step 4:**After the </head> tag, type <body></body> as shown below. These tags indicate the beginning of the body of your web page. They contain all the information about how your webpage should include text, images, forms, animation, etc.

**Step 5:** Within the <body></body> tags, add <p></p> as shown below. The <p></p> tags indicate that the text written between them is a paragraph. You can write your Hello message within the <p></p> tags.

<!DOCTYPE html>

<html>

<head><title>Hello World!</title></head>

<body>

<p>Welcome to HTML, Hello World!</p>

</body>

</html>

**Note:** Make sure that you haven’t missed any opening or closing tags.

**Step 6:** Save your file as “filename.html” with any filename. Now, go to the folder in which your file is saved. You may observe the logo of the file as that of your default web browser as you have saved this file with a .html extension.

**Double-click on the file to open it.**  
*Viola! You have created your first web page!*

Here is an additional reading [link](https://developer.mozilla.org/en-US/docs/Web/HTML) you can utilise to learn more about HTML.

<https://developer.mozilla.org/en-US/docs/Web/HTML>

**Q. HTML**

Which of the following statements about HTML is NOT correct?

**HTML code processes client requests received by the server.**

**Feedback :**

*The programs on the server are responsible for processing client requests. The programs can be written in languages such as Java. HTML code only defines how a web page should look. The server processes the client request and sends an appropriate HTML code.*

**Title**

Examine the following statement and state whether it is true or false.  
The title enclosed between the <title> and </title> tags will be displayed at the top of the webpage.

False

**Feedback :**

*The text enclosed between <title> and </title> tags will be displayed on the title bar of the browser and not on the webpage.*

## ACTIVITY:

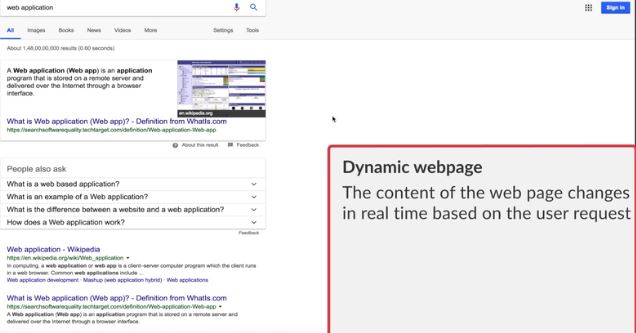
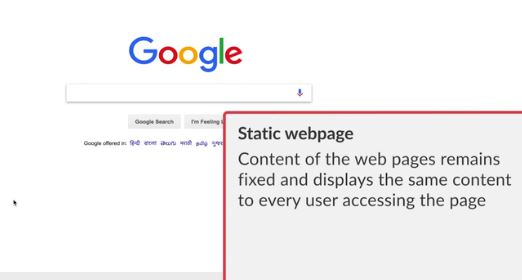
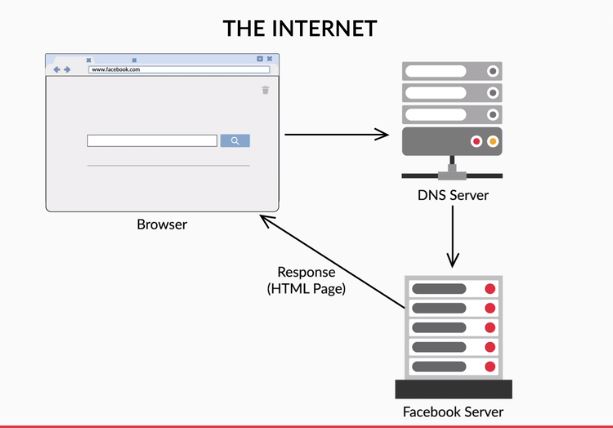
HTML is all around you while browsing the web. To understand how closely you interact with HTML day in and day out, do the following:

* Go to any simple looking website, such as <http://info.cern.ch/> (this is the first website ever created for the internet).
* Right-click on the screen, and select View Page Source.
* You can now see the HTML code of the website.
* You will see that the HTML file has a lot of tags. The HTML tags you have used until now are some of the basic tags and you should be able to find all of them in this HTML source code. For example, you should be able to see the <html> and </html> tags very easily.

Now that you have some tricks up your sleeve, feel free to experiment. Look at what changes take place in the developer console (Ctrl + Shift + i) whenever you do something on the website. You can see a lot of new things there that you will be able to understand fully when you are proficient at both back-end and front-end development.

# Understanding web applications

By now, you have gained an understanding of all the major concepts and entities involved in the process of typing a URL in your browser window. In this video, you will be given a brief overview of the process thus involved. You will further learn about an important classification in web applications.



Once you have understood how you get a web page from a server, another question arises: What if you are requesting a web page that will be created just for you and will change as per the user?

For example, when you want to log in to Facebook, you go to [www.facebook.com](http://www.facebook.com/), where you can see options to log in and register. This webpage is the same for anyone accessing it from anywhere. In other words, **this page is constant or static**.

When you log in, a web page is constructed with your news feed, notifications, suggestions for friends, advertisements, etc.

Clearly, your Facebook homepage is assembled at request and is not a pre-made and saved HTML page. There are programs running on the server that assemble that page for you. In other words, this page can be called **dynamically generated**.

Based on the differences discussed above, websites can be broadly classified as:

1. **Static websites:** These are basic websites where the content of the web pages remains fixed and is displayed to every user accessing the page.
2. **Dynamic websites:** The content of these web pages changes in real time based on the user viewing the website, the location from where it is being accessed, and other differentiating criteria. For example, the Facebook news feed of a user or a Twitter account is dynamically generated for its users based on their viewing history, usage behaviour, and location.

The program that assembles all the information for your web page is called **business logic**.

The program that renders all the assembled information into your Facebook home page is called **presentation logic**.

Q> **Static and Dynamic websites**

Here is a list of URLs for some websites. Open each one, examine them and select the website(s) which can be called a static website.

Top of Form



[**http://info.cern.ch/hypertext/WWW/TheProject.html**](http://info.cern.ch/hypertext/WWW/TheProject.html)

**Feedback :**

This website’s content is constant, it does not change based on any variables like who is accessing the page, at what time is it being accessed etc.

**Correct**



<https://www.reddit.com/> fALSE



[**https://www.atlassian.com/time-wasting-at-work-infographic**](https://www.atlassian.com/time-wasting-at-work-infographic)

**Feedback :**

This website has only constant graphics and text which remain the same always. The content is not dynamic.

**Correct**

**Static and Dynamic websites**

q. You created a webpage in the previous segment with the code as follows:

<!DOCTYPE html>

<html>

<head><title>Hello World</title></head>

<body>

<p>Hello World</p>

</body>

</html>

Is this page a static web-page or a dynamic web-page?

Top of Form



**Static web-page**

**Feedback :**

*How this page would appear to a user does not depend on any variables. This page would always appear the same to everyone. Hence this is a static web-page.  
Consider that if this page were to say "Hello, the current time is:" with the current time shown below. In this case, the page would be called a dynamic webpage.*

**q. Static and Dynamic websites**

Is the current webpage on which you are answering this question static or dynamic?

**Dynamic web-page**

**Feedback :**

*How this page would appear to a user does depends on some variables such as videos id, quiz id etc. These ids are fetched from the database at runtime. Hence this is a dunamic web-page.*

**Introduction to dynamic website creation**

The main aim of this module is to create dynamic websites or web applications. Although you will learn to use the MVC architecture to achieve that, there are certain older approaches that you should be aware of before moving on to that.

There are two traditional ways of creating dynamic websites using Java:

1. Servlet
2. Java Server Page (JSP)

Let's now understand servlets and JSPs in detail through the following video:

You learnt about servlets:

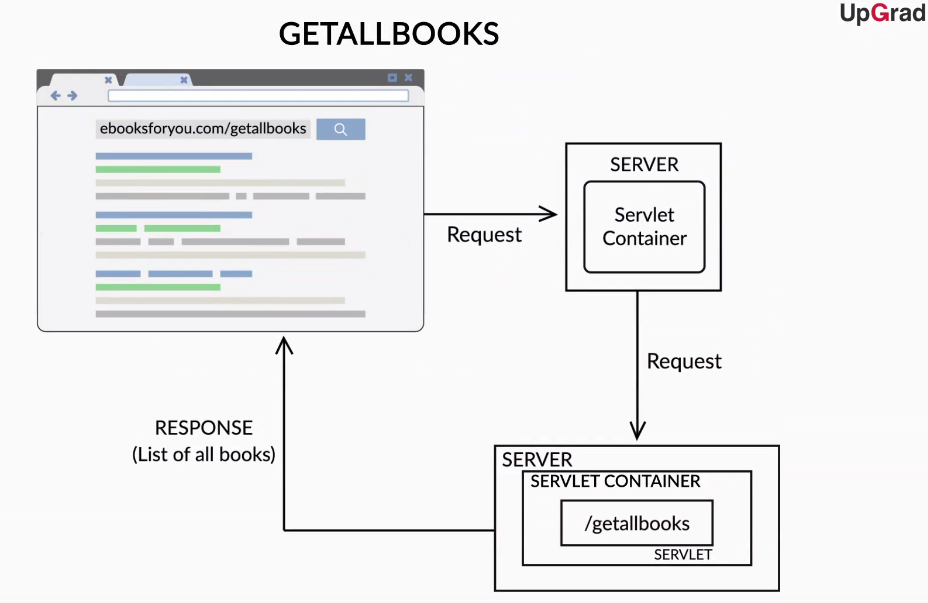
Servlet is a Java class that responds to client requests and can be used to generate dynamic websites. A servlet is a program that:

* Receives a request (for example, a request for your list of suggested friends on Facebook)
* Processes the request and conducts business logic (does all the processing required to find friend suggestions for you), and
* Creates the presentation of the website based on the business logic executed in the step above, i.e., returns the list of friends as HTML.

Different servlet programs can serve different purposes. Consider the example of your Facebook homepage again. A servlet class can be assigned to pick advertisements for you, while another one creates and renders your notifications.

A request, thus, firsts goes to a **servlet container**, which loads the appropriate servlet and sends the request to it for processing.

You saw the example of ebooksforyou.com, as shown below. You learnt how a server would receive a request, process it via servlets, and return the response to the client.



You also learned about **JSPs**.  
JavaServer Pages (JSPs) are another way to build dynamic websites or web applications. This concept was introduced to ease the life of developers. It decouples the presentation logic from the business logic of generating a dynamic website, thus allowing the front-end developers to work separately from back-end developers. With this, it also makes the presentation logic less dependent on the business logic, and vice versa. This makes it easier for developers to make changes to the business logic without affecting the presentation logic, and vice versa.

You will understand in detail the concepts of servlets and JSP and how JSPs are better when you're building a Hello World website using these two approaches in the upcoming segment

Bottom of Form

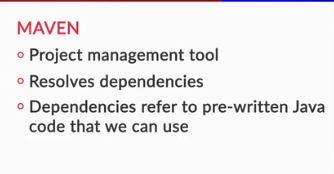
Bottom of Form

q> **ebooksforyou.com**

From the www.ebooksforyou.com example, you saw in the preceding video, describe the process, step-wise from the server receiving the request to the client receiving the response.

**Suggested Answer**

1. Since the server is a web server here, it is listening for client requests at the port 80.
2. When you type www.ebooksforyou.com, the DNS server will give you the IP address for the server hosting the website www.ebooksforyou.com
3. The request is received by the server at the 80 port. The request is received as an HTTP GET request since a user is trying to get the website from the web-server.
4. The servlet container then looks for the servlet class which is mapped to the URL "getallbooks", i.e. the servlet class that is supposed to get a list of all books for you.
5. The request goes to the appropriate servlet class which runs a program to retrieve a list of all available books.
6. The servlet creates an HTML page with this list of books returns the same.
7. The server then responds with the created HTML page which is finally displayed to the user.



You learnt that Maven is a project management tool, which will do the following for you:

* Install the required third-party library for your Java project
* Resolve dependencies. Maven will download all the pre-written Java code, such as third-party libraries, that your application will use. For example, in this case, you would not want to write the code for a servlet container. Instead, you will add a dependency, and Maven will download the code for the servlet container. These dependencies are present as JARs or Java Archives.
* Manage all of your compiled Java files as JARs (Java Archives).
* Give you a project structure. You can observe this structure in the upcoming video while creating a web application.

Maven resolves dependencies via a **POM.xml** file. This is a file where the developer, i.e., you, will mention all the other code and libraries that your program will be dependent upon.   
For example, if you are using SpringBoot, there are some third-party libraries with pre-written code that you can use and install in your project. You just need to refer the same in the POM file such that Maven downloads all these pre-written Java codes in the form of JARs.

Once you have mentioned the required dependencies in the POM file, Maven downloads and stores them in place so that you can use them when needed.

Maven also gives you a project structure, which you will be able to observe when you start writing your application.

SpringBoot here is a framework that will simplify the development process by configuring a lot of parameters for you. You will be able to appreciate the ease of using SpringBoot as you go ahead with this module in developing the web application.

Q> **Maven**

Which of the following would Maven help you with?

Getting a project structure based on pre-defined templates

**Feedback :**

As already mentioned, Maven gives you a project structure which you will be able to observe later in this module and multiple following modules.

**Correct**You missed this!



**Downloading all dependencies mentioned in the POM file as JARs**

**Feedback :**

Maven downloads all Java code that your application needs based on what you define in the POM file.

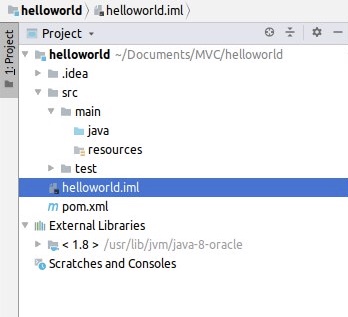
Let's now move on to creating your first web application using servlets.   
In the following video:

* You will create a project using Maven and IntelliJ.
* You will mention the dependencies you require in the POM.xml file.
* You will write the Java classes required to create the web application.

**Note:** You will be writing the Java code to create an application for the first time through this video. Not getting the hang of it at the first go is natural. You will learn about it in depth, along with certain concepts such as SpringBoot, in the upcoming session. Through this segment and the next, you should try keeping up with the SME and implement all the code on your local machine. Soon, you will be comfortable and fluent enough with writing web applications.

As you observed in the video, you need to create a Maven project in the IntelliJ IDE.   
**Group ID** is the team or organisation that you work for. This name would be unique throughout the organisation you work at.  
**Artifact ID** is the name of the project.   
**Version** simply denotes the version of the application. For example, you can also have upgraded applications or change the architecture some time down the line.

You may observe that the created Maven project already had a structure and folders for specific files, as in the image below:



In this folder structure:

* The main directory consists of all the pieces of code related to website development.
* The test folder consists of all the test cases that you can write to test your project.
* The pom.xml file is a configuration file that tells Maven how to integrate all the libraries, frameworks, and templates into your project.

The POM.xml file was already created by Maven, and certain details such as your project’s artifact ID, group ID, etc. were already configured in the POM file. You then added the following dependencies in the POM.xml file:

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.5.RELEASE</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

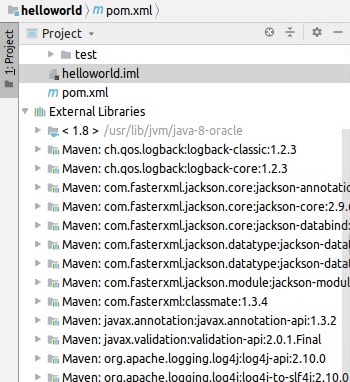
</dependency>

</dependencies>

We added the following to the POM file:

* **Spring-boot-starter-parent:** A parent POM file is another POM file that a child POM can inherit. It has some default configurations required to use Spring Boot to develop any kind of an application. It also controls versions of the dependencies added. So, in order to add any Spring Boot dependencies, you need to add the starter parent to your POM file. You can read more about what the Spring Boot parent POM does in Section 13.2 of the Spring documentation [here.](https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-build-systems.html)
* **Spring-boot-starter-web**: Spring Boot can be used to build various kinds of applications. In our case, we are trying to build a web application, and hence, we added this dependency to download all the JARs required to build a web application using Spring Boot.

As you have observed, Maven imports all the required JARs or pre-written Java code based on the dependencies mentioned in the POM file. You can see all downloaded JARs under external libraries, as shown below:



Q. **POM.xml file**

What are the advantages of using a POM.xml file according to what you observed while coding along with the previous video?

**Suggested Answer**

Here are a few advantages of the POM.xml file.

* You need not download required JARs if you want to run your project on a different machine. When you run your Maven project, all required dependencies mentioned in the POM file will be automatically downloaded and compiled as JAR files for your project.
* You need not search and download all required dependencies from official websites, instead, Maven does the job for you via the POM file.

You can go and read more about the advantages of the POM file [here](https://maven.apache.org/guides/introduction/introduction-to-the-pom.html).

The next step in the project is to create the following main or application class to run the program.   
**Note:** The path to this class should be: main → java → upgrad → HelloWorldApplication.java

package upgrad;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.web.servlet.ServletComponentScan;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

public class HelloWorldApplication {

public static void main(String [] args){

SpringApplication.run(HelloWorldApplication.class, args);

}

}

For now on, all you need to understand is that the application class here does the following:

* It starts up the server and gets your application up and running.
* It runs your application using the Tomcat server on port 8080. This means that all the functionalities that your application provides would be accessible to the user through the 8080 port of the server that the application is running on.

You have also come across something called annotations in this class. For example, the @SpringBootApplication annotation defines that you are using SpringBoot here to create a web application.   
You will be using a lot more of these annotations in the future. You can read more about annotations [here](https://www.geeksforgeeks.org/annotations-in-java/).

<https://www.geeksforgeeks.org/annotations-in-java/>

# Your first web application -II

In the following video, you will create the servlet class to host a Hello World page.

You have learnt to implement the following:

* Create another Java class called the HelloWorldServlet. A servlet is essentially a Java class that responds to HTTP requests. Using servlets spares you the work of writing the code for HTTP requests, packets and worry about the networking aspects of the code. All the worries about adhering to the HTTP protocol and the TCP/IP protocols can be left to the servlet; you can just specify the HTTP request method that you need to use.
* This is where all the magic happens:

package upgrad;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

@WebServlet("/helloworld")

public class HelloWorldServlet extends HttpServlet {

public void service(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {

PrintWriter printWriter = response.getWriter();

printWriter.print(s: “Hello World!”);

printWriter.close();

}

}

This class will process the client request for the Hello World page. Here, @WebServlet is an annotation that denotes that this class is a Java web servlet. The servlet container invokes the service method of the appropriate servlet class based on the request.

* The @ServletComponentScan annotation in the HelloWorldApplication application class: This annotation tells your application class to look for the servlet class you have created annotated with @WebServlet.

This is what your application class looks like now:

package upgrad;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.web.servlet.ServletComponentScan;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ServletComponentScan

public class HelloWorldApplication {

public static void main(String [] args){

SpringApplication.run(HelloWorldApplication.class, args);

}

}

Now, your application is up and running. This raises the question: **Where are the servlet container and the server program?**

This is taken care of by Spring Boot. Spring Boot comes with an embedded Tomcat server, which provides you with the servlet container. Your web application is hosted on the Tomcat server. You can observe the same written on the console when you try to run your application.

Q. **Maven and Spring Boot**

State true or false:  
Spring Boot provides you with the servlet container which routes requests to appropriate servlets.

False

**Feedback :**

*Tomcat provides you with the servlet container. The servlet container routes requests from the user for a specific URL to appropriate servlet classes. e.g. when the servlet container receives a request for the url "/helloworld", the servlet container will call the helloworld servlet.*

**Maven and Spring Boot**

Which one of the following options is correct?

None of the above

**Feedback :**

*Tomcat only provides you with the servlet container. The servlet container invokes the required servlet class. Whereas Spring Boot helps you with processing the business logic and presentation logic through the application class and the servlet class.*

By now, you must be clear with the roles of the servlet container, the server program, and servlets. You can go [here](https://dzone.com/articles/what-servlet-container) to read more about the roles of specific programs (server, servlet container, and servlets) involved in hosting the website.

Now that your application is up and running, let's observe how the web application can be accessed from the browser. In this video, you will learn how to access the web application and implement an HTML page in a servlet class

<https://dzone.com/articles/what-servlet-container>

**Accessing the website**

Normally, you would need to access the server on which the website is hosted through the IP address of that device. Here, you will be trying to access the website hosted on your own computer from a browser on the same computer. In other terms, you are trying to access the website “locally”. Hence, you need to call the **localhost** through your browser.

Localhost is an alias for the computer that you are working on, as every machine can be referred to with an IP, and you can refer to your own machine with the IP address **127.0.0.1**. This is the default IP assigned to localhost by convention.

Since multiple services could be running on your computer, you need to specify the port on which the service is hosted. As you have observed earlier, the server is hosted on port 8080; so, you need to use port 8080 to access the web application.

Hence, you need to type the following URL in your browser window to access your web application:

**localhost:8080** or **127.0.0.1:8080**

By default, this will look for a class or servlet mapped to the “/” request.  
But you want to access a facility provided by a servlet class mapped to the request “/helloworld”, and hence, you need to convey the same through the URL by adding “/helloworld”.

**localhost:8080/helloworld** or **127.0.0.1:8080/helloworld**

You have also learnt how to implement an HTML code in a servlet class rather than rendering the plain text **"printWriter.print(s: "Hello World!");"**. Now your servlet class returns an HTML page rendered with the title and other HTML page elements.

printWriter.print("<!Doctype html>");

printWriter.print("<html>");

printWriter.print("<head>");

printWriter.print("<title>Hello World Servlet!</title>");

printWriter.print("</head>");

printWriter.print("<body>");

printWriter.print("<p>Hello World!</p>");

printWriter.print("</body>");

printWriter.print("</html>");

**Maven**

Which of the following options is correct?

Top of Form



**The embedded Tomcat server is provided with Spring Boot.**

Bottom of Form

**JAR files**

Explain JAR files in your own words and when are they used?

**Suggested Answer**

A JAR file consists of various components that are required for executing a Java application. The main advantage of using a JAR file is that a single JAR file can store numerous files aggregated into one. This makes it easier for the user to download all the required files from a single compressed JAR file. A JAR file usually consists of compiled Java code, XML files, JSON objects, and many other files required for the successful execution of a Java application

**JAR files**

Can you list down the different tasks involved in implementing the application were taken care of by JAR files or dependencies downloaded as part of your Maven project?

**Suggested Answer**

Some examples of tasks taken care by JAR files in your application are as follows:

* Routing a client request through the servlet container provided by Tomcat.
* Sending and receiving HTTP communication.
* Finding the appropriate servlet class and communicating the client request to it.
* Hosting the website on the HTTP port or the port 8080 of your device.

# Web application using JSP

You already saw how to create a simple application using a servlet. Now, imagine that you need to implement the HTML page of any real-world application. How complicated it would be to write all the code in the servlet class?  
Thus, to make a developer’s life less complex, JSP was introduced.

In simple terms:

* Servlets have HTML code within Java code.
* JSP has Java code within HTML code.

JSP helps you decouple the implementation of the presentation logic (HTML code) and the business logic (Java code). You will be able to observe this difference when you write the same Hello World application using JSP.

In the following video, you will learn how to implement the same Hello World application using JSP and understand the differences in developing the web applications between JSP and servlets.

You need to add the following dependency in the POM.xml file:

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

This dependency is needed to host your JSP on the Tomcat server. Although Spring Boot does have an embedded Tomcat server, the tomcat-embed-jasper dependency is needed to enable JSP support. **JSP pages are compiled dynamically into servlets in the server**. Hence, the processing of JSP is slower than servlets.

You learnt how to create the JSP file in the **webapp folder of the main folder**:

<!Doctype html>

<html>

<head>

<title>Hello World JSP!</title>

</head>

<body>

<p>Hello World. This is your first JSP!</p>

Current time: < %= new java.util.Date() % >

</body>

</html>

You can access your website by going to the following URL:

**localhost:8080/helloworld.jsp**

**Servlet vs JSP**

By now, you must be clear that JSP is a better way of creating dynamic websites. Using JSP as the technology to build a web application, work can be separated between a person responsible for developing the HTML code, or the front-end, and the person developing the dynamic components, or the back-end. That is, it decouples the implementation of the presentation logic (HTML code) and the business logic (Java code), which makes the web application more robust and easier to maintain.

In the JSP code you saw, the business logic was limited to this one Java statement **"new java.util.Date()"**. You can similarly call other classes written in Java too.

This is how the business logic and the presentation logic are decoupled using JSP.

You can read more about the difference between servlets and JSP [here.](https://www.geeksforgeeks.org/difference-between-servlet-and-jsp/)

Think about the large HTML code files that are to be written for real-world web pages. Servlets and JSPs cannot possibly be the only ways to create it. In the next segment, you will learn about the drawbacks of servlets and JSP and the solution to these problems in detail.

**Note:** You need to have an overall understanding of servlets and JSP, but you are not expected to be proficient at implementing them since the focus here is on building a dynamic website using the MVC architecture.

**Q. JSP vs Servlet**

Which of the following statements regarding servlets and JSP is NOT correct?

Top of Form



You cannot create dynamic websites using servlets.

**Feedback :**

Dynamic websites can be created using both servlets and JSP.

Bottom of Form

**Q. Servlets and JSPs**

What do you think are the drawbacks to using servlets and JSP to create dynamic websites?

**Suggested Answer**

One of the major drawbacks of Servlets and JSPs is the separation of concerns. Even with JSPs, a front-end or HTML developer would be worried about adding Java scriptlets to the HTML code. This would mean that an HTML developer would have to deal with Java code often and a Java developer would have to deal with HTML code often.  
Also, JSPs are compiled to create servlets. Due to this process, debugging becomes harder and processing is also slower.

# Introduction to Model-View-Controller architecture

Now that you have learnt how to create web applications using JSP/servlet, in this video, the SME will discuss the drawbacks and the importance of the MVC architecture.

One of the major drawbacks of developing a web application using servlet/JSP is the separation of concerns. MVC or the Model-View-Controller aims to solve exactly this problem. It separates an application into components called model, view, and controller:

* **Model:** This represents the data layer of the application.
* **View:** This is the presentation layer of the application.
* **Controller:** This is the business logic of the application.

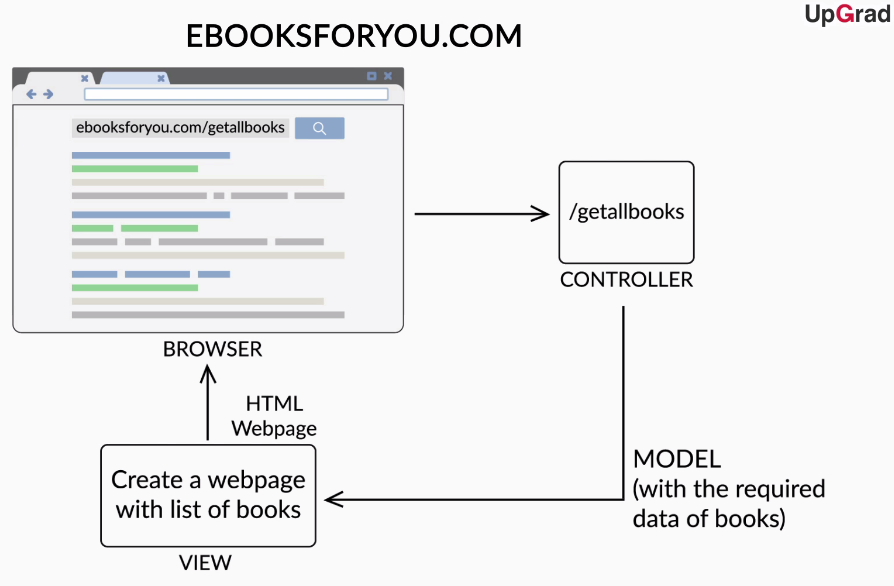
Let's try to understand these components in the context of a web application.

A **model** represents the data of a web application. The data is stored in the form of a model object and passed to the view by the controller.   
For example, your friend suggestions, notifications, etc. will be stored as a model object and sent to the view by the controller class.

A **controller** involves classes and methods that process the business logic for you. It receives a user request, processes it, and uses the model and view components to send the appropriate response to the user.  
For example, in the Facebook newsfeed of a particular user, the controller would process all the relevant posts to display on the news feed, friend suggestions, notifications, etc. It would then return all this information to the view.

The **view** component of the web application takes information from the controller (in the form of a model) and renders it on the web page to be displayed to the user.   
For example, the view would ensure that your notifications are visible at the top-right corner of the webpage and in a particular fashion (the font, colour, etc.). The actual content of the notifications will be sent to the view by the controller.

You also saw the same "ebooksforyou" example and what happens when we implement the same functionality using the MVC architecture. Let's take a look at the examples again:



You can clearly see now that the separation of concerns is being handled well here. Unlike servlets, the front-end (the HTML code), the back-end (the Java code), and the data are being handled by different components. This means that the view part handles the front end, the controllers handle the back end, and the model component is used to exchange data between the view and controller components.

We highly recommend that you go through [this](https://www.upgrad.com/blog/mvc-architecture-in-java/) blog post about the MVC architecture before moving on to the next segment.

Q. **MVC Architecture**

Consider that you are building a web application whose view part needs to be modified very often. The view part(HTML page) is also very complicated with intricacies.  
Changes like a change in text colour or site of the website are very frequent. Also, functionalities of the website are modified often too. For example: initially, a user could subscribe to an email list which is now replaced with an option to follow the company on various social media platforms.  
Which method would you prefer to build your web application?

**MVC architecture**

**Feedback :**

If the application is implemented via MVC architecture, an HTML developer will be able to work on the presentation logic separately and make the changes in the view part. The Java developer would be able to work separately to make changes to remove the function to add an email to an emailing list.

Q. **MVC Architecture**

Which component of the MVC architecture is responsible for the presentation logic, i.e. for generating appropriate HTML code to be sent to the client?

view

**Feedback :**

*View takes the appropriate data from the controller as model and builds the HTML code around it to be shown to the client as a webpage.*

Q> **MVC Architecture**

Consider that you have built a website to show a student’s exam scores. Here are the specifics of the website:

1. The website is accessed via the URL 192.168.2.1:8080/scores.
2. Assume that when a person is accessing the website, the name of the person is coming to the server as input.
3. The student data is stored as models with each student’s name and scores.
4. The scores of the person are to be displayed to the user at the screen as “ Hello<student name>, your score is <the score>.”

Assuming that the web application is implemented using the MVC architecture, explain the procedure of the application responding to the client request from the server receiving the request for the URL with the student name till the score being displayed to the user.

**Suggested Answer**

1. The request will first come to the controller mapped to the URL "/scores"
2. The controller will take the student’s name as input.
3. The controller will retrieve the score of the student based on the student’s name as a model (the model will have the name of the student and score of the student).
4. The controller will return this model and send it to the view component.
5. The view component will take the model. It will take the student name and score data out of the model and create an HTML page with it with. The view will already have a template of the HTML page with fixed components like the title or the sentence template mentioned in the problem statement. It will only insert the name and score.
6. Assume that the name is "Ramesh" and the score is "23". The webpage will display "Hello Ramesh, your score is 23."

Now that you have gained an understanding of the theory related to the MVC architecture, let's move on to the implementation of the same using Java. There are frameworks that provide us with pre-written code as JARs and guidelines to implement the MVC architecture, using Java to create a web application. One of such frameworks is the **Spring MVC**.

In the next segment, you will learn how Spring MVC implements the MVC architecture and how are we going to use it to create a web application.

# Spring MVC and Springboot

MVC is an architecture that can be implemented to create standalone applications or web applications. SpringMVC provides us with a popular Java implementation of the MVC architecture to help develop web applications using MVC architecture. It comes with a pre-written MVC framework that implements the underlying details of creating a website or web application, which leaves the developer with only the task of writing the models, views, and controllers.

Let's understand what SpringMVC is and how it implements the MVC architecture.

**Spring MVC**

Which component of the MVC architecture is responsible for mapping particular requests to their specific controllers?

**Handler mapping**

**Feedback :**

*In the Spring MVC architecture, the handler mapping component is responsible for mapping incoming user requests to their corresponding handlers. For a user request, the handler mapping component responds to the dispatcher servlet with the corresponding controller information.*

**Spring MVC**

Which of the following components is responsible for adding presentation logic to the model data and finally returning a web-page to be shown to the user?

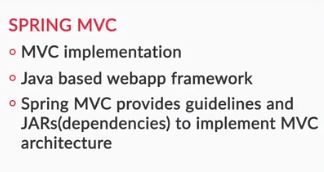
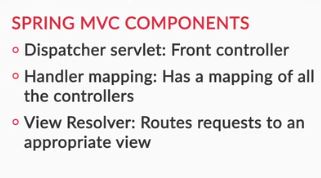
Top of Form

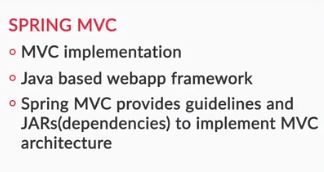
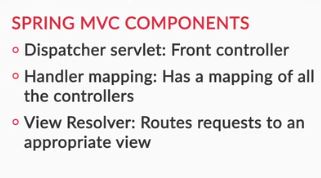


**View**

**Feedback :**

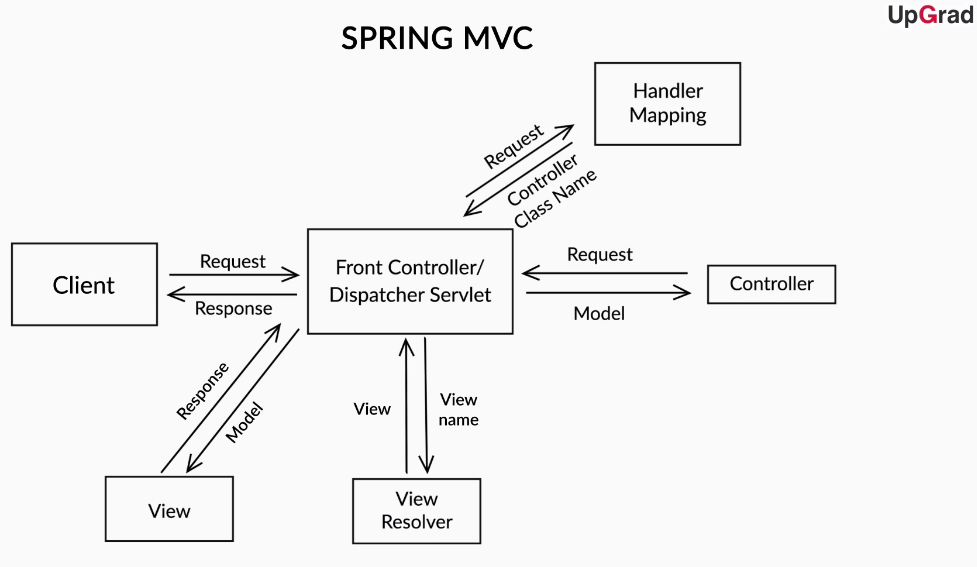
*The view receives the model and stitches together the HTML and the data and renders the web page finally.*

Bottom of Form



You saw that Spring MVC implements the MVC architecture through the **dispatcher servlet, handler mapping**, and **view resolver**.  
Overall, the following components are involved in creating a web application using Spring MVC:

* **Dispatcher servlet:** This acts as a front controller. This means that a request coming to the server is first intercepted and responded to by the dispatcher servlet. The dispatcher servlet acts as a point of communication between all the other components.
* **Handler mapping:** When the dispatcher servlet receives a request, it contacts the handler mapping to get the name of the appropriate controller to which the request should be routed. Handler mapping finds the right controller to be contacted and sends that information to the dispatcher servlet. As you already learnt, there can be different controllers for different purposes, such as getting posts for your news feed and getting friend suggestions for you.
* **Controller:** A controller is contacted by the dispatcher servlet to process a request. It processes the client request, retrieves the required data or the model, and sends this model along with the view name that the model has to be rendered to the dispatcher servlet.
* **Model:** The model represents the data layer of the website. All the required data is stored in the database and sent from one component to another in the form of models. Models are basically Java classes with specific attributes. For example, a model class storing data about a student in a school would have attributes such as name, ID number, date of birth, etc.  The model layer also implements functionalities for the controller to retrieve and manipulate data in the database. You will be learning more about this in the next module.
* **View resolver:** A view resolver receives the view name from the controller via the dispatcher servlet. It has the mapping of view names and views. It finds the appropriate view and sends it to the dispatcher servlet. For example, the view name could be helloworld and the view could be helloworld.jsp. The view resolver would receive helloworld and return the view helloworld.jsp to the dispatcher servlet.
* **View:** The view receives the business logic and/or the model data from the controller via the dispatcher servlet and builds the HTML around the data that it receives. It builds the final web page that a client will be able to see. It sends the HTML code back to the dispatcher servlet, which then sends the same to the client as a response.



We are going to use Spring Boot here to create our Spring MVC application. You have already used Spring Boot earlier to develop your Hello World page using servlet/JSP.

To give you a perspective on how Spring Boot will ease your life, here are a few of the many things you would have to do to use Spring MVC without Spring Boot:

* You would have to configure the dispatcher servlet using an XML file and mention names and mappings of controllers, etc.
* Spring Boot comes packaged with the Tomcat Server. This means that you don’t have to download the Tomcat server program, configure it, and integrate it separately with your application.
* Spring Boot uses annotations such as @SpringBootApplication, which you saw earlier, and bypasses a lot of XML configurations that you may have to write to integrate different programs with your application such as database services

Q. **Spring MVC**

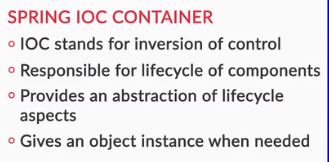
In your own words, explain how Spring MVC overcomes the disadvantage of simple servlets or JSPs?

**Suggested Answer**

MVC architecture provides the separation of concerns via the three components:model, view and controller.  
The front-end(presentation layer) and the back-end(business logic) are somewhat separated if you use JSPs in the technology stack. Although, MVC provides with another layer of abstraction called the model which separates the data layer of the application too.  
In Spring MVC architecture, the dispatcher servlet interacts with the specific controller to retrieve the required response data and separately interacts with the view resolver to obtain the specific view. Then, it sends the whole information as a response to the client/user’s request.  
This way, the Spring MVC architecture implements the presentation logic in the view template and the business logic in the controller. It also provides us with a lot of default configurations required to implement the MVC architecture. Hence, creating a web application using SpringMVC is a much better approach than doing the same via servlets/JSPs

As you go through this course, you will get more and more clarity on why we are using Maven or Spring Boot and you will be able to appreciate their utility more.

An important core concept required to understand Spring MVC is the **Spring IOC container,** which plays a vital role in managing all the components of your web application. Although you will be able to appreciate the ingeniousness of the Spring IOC container more when you are done building a web application with multiple components, here is an overview for you to understand the functions of the IOC container.



The IOC container “registers” or keeps track of all the classes in your application, which are components of Spring MVC, such as the controller. It creates an object of each of these classes to be used when required. You will understand in the upcoming session how this object creation and management help us.

Now that you have gained an understanding of all the theory related to MVC and all the tools we are using, it is time to put all this knowledge to use. In the next segment, you will be building a simple Hello World page using Spring MVC and Spring Boot.

# Hello World using Spring MVC

Let's now see how you can build a simple Spring MVC application using Spring Boot.

In the following video, you will learn how to create a controller that processes the request for your web page and returns "Hello World" similar to the servlet class implementation earlier.

You have learnt how to create a simple controller class annotated with @Controller, as follows:

**package** **upgrad**;

**import** **org.springframework.stereotype.Controller**;

**import** **org.springframework.web.bind.annotation.RequestMapping**;

**import** **org.springframework.web.bind.annotation.ResponseBody**;

**@Controller**

**public** **class** **HelloWorldController** {

**@RequestMapping**("/hellospring")

**@ResponseBody**

**public** String **helloSpring**() {

**return** "Hello Spring!";

}

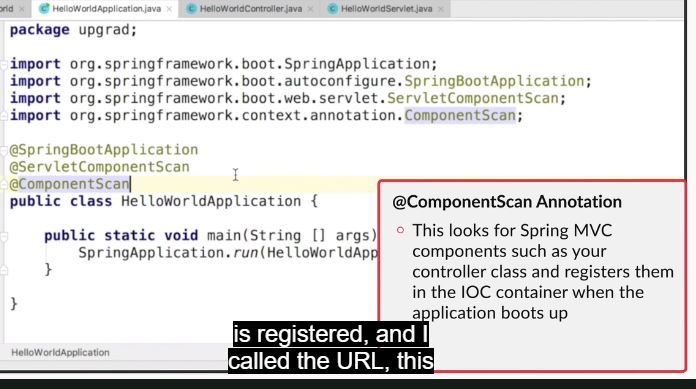
}

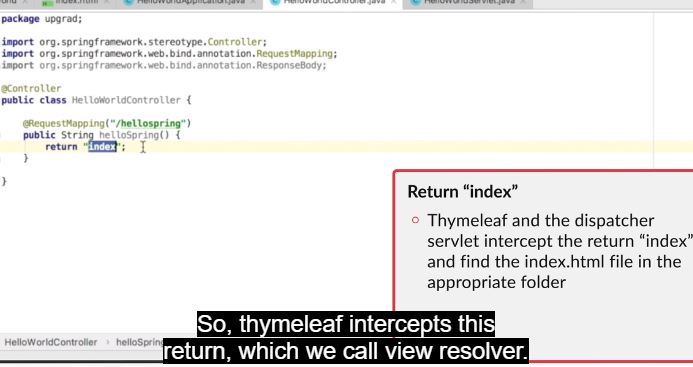
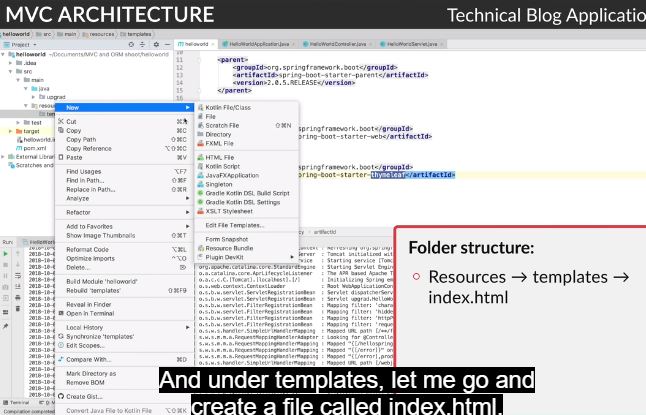
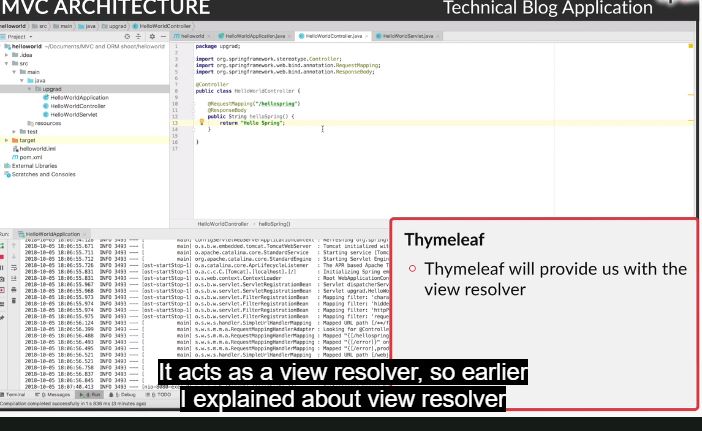
You have come across three annotations here:

* @Controller: This annotation tells Spring that the class is a controller class.
* @RequestMapping: This annotation is used to tell the Spring (dispatcher servlet and handler mapping classes) that the controller method is mapped to a specific URL. In this case, it is mapped to "/hellospring" (similar to how you mapped the servlet to a specific request using the @WebServlet annotation, i.e., @WebServlet("/helloworld")).
* The @ResponseBody annotation denotes that the object returned by this method is to be returned to the user as an HTTP response.

You already have an application class ready. So, when you run this application and hit the URL **"localhost:8080/hellospring"**, you should be able to observe "Hello Spring!" written on your browser screen.

So far, you have learnt how to return the simple plain text from the controller class. In the following video, you will learn how to create an HTML page to the client using another dependency called Thymeleaf.





The following Thymeleaf dependency is added to the POM.xml file. Thymeleaf is a templating engine that allows you to write the view using HTML and also provides you with a view resolver :

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

In the video above, a simple HTML page in the resources → templates folder named index.html is implemented and Thymeleaf takes the view name and returns the appropriate view to the dispatcher servlet. That is, it takes “index” as the input, which is the view name, and returns index.html, which is the view.

The @ResponseBody annotation is removed since the view name is returned by the controller method and should be intercepted by the view resolver.   
This is how your controller class should look now:

**package** **upgrad**;

**import** **org.springframework.stereotype.Controller**;

**import** **org.springframework.web.bind.annotation.RequestMapping**;

**import** **org.springframework.web.bind.annotation.ResponseBody**;

**@Controller**

**public** **class** **HelloWorldController** {

**@RequestMapping**("/hellospring")

**public** String **helloSpring**() {

**return** "index";

}

}

**Activity 1:**

If the @ResponseBody annotation is still present in the controller class, you will be able to see “index” written on the web page. Try adding the annotation and running the application to observe the results.

Currently, the models are not created but simply hosting a static web page through Spring MVC. In the upcoming session, you will also see how Thymeleaf helps in creating dynamic websites and adding model data returned by the controller to the view template.

**Activity 2:**

You learnt that Spring MVC implements the MVC architecture using dispatcher servlet, handler mapping, view resolver, controller, and view. But you have never implemented the classes for the aforementioned Spring MVC components; these classes are pre-written Java codes and Spring Boot dependencies help you download them as JAR files. When you run the application, these classes run in the background of the application to implement the required functionality.

You can check out the implementation of these classes in the IDE using the shortcut '**Ctrl + N**' and type any of the components names, such as 'DispatcherServlet'. However, you need not get into the details of the class implementation, which will be taken care by the Spring Boot. This is how the Spring Boot provides the base framework with all the classes and configurations required to develop the web application.

q> **Controller**

What would happen if @RequestMapping is added to the hellospring() method but the @Controller annotation is not added to the controller class?

**Suggested Answer**

Spring needs to find the relevant class for which requests should be mapped to. It searches for classes with the @Controller annotation, and according to this annotation, it will map the relevant methods to the specified URL requests. Hence, if the @Controller annotation is not present, it will lead to an error saying 'application has no explicit mapping from /hellospring'.

q> **@ResponseBody**

What would happen if the controller method is annotated with @ResponseBody and returns the string “index”?

**Suggested Answer**

The view name “index” would not go to the view resolver. Instead, it would be returned as a response to the user and the user will observe “index” on the browser when he/she requests localhost:8080/hellospring.

q> **Website**

Now, try to add some content to the basic website using the HTML file. The content would be as follows:  
**Title:** 'Technical Blog'  
**Heading:** 'First post'  
**Description:** "Computer software, or simply software, is a part of a computer system that consists of data or computer instructions, in contrast to the physical hardware from which the system is built. In computer science and software engineering, computer software is defined as the information processed by computer systems, programs, and data. Computer software includes computer programs, libraries, and related non-executable data, such as online documentation or digital media. Computer hardware and software require each other, and neither can be realistically used on its own."

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Technical Blog</title>

</head>

<body>

<h2>First post</h2>

<p> Computer software, or simply software, is a part of a computer system that consists of data or computer instructions, in contrast to the physical hardware from which the system is built. In computer science and software engineering, computer software is defined as the information processed by computer systems, programs, and data. Computer software includes computer programs, libraries, and related non-executable data, such as online documentation or digital media.

Computer hardware and software require each other, and neither can be realistically used on its own.</p>

q> **Website**

Now, delete the index.html file from your project and make the required changes to the Java code to display the following string 'Welcome to Technical Blog' in the browser

**Suggested Answer**

In order to display the string 'Welcome to Technical Blog' in the browser without using an HTML file, you need to annotate the hellospring() method with '@ResponseBody'. Then, the dispatcher servlet would not look out for a corresponding HTML file and display the String in the browser.

**@Controller**

**public** **class** **HelloWorldController** {

**@RequestMapping**("/hellospring")

**@ResponseBody**

**public** String **hellospring**(){

**return** "Welcome to Technical Blog";

}

}

q> **MVC Architecture**

Which of the following components of Spring MVC architecture is responsible for business logic such as calling the necessary helper functions, consolidating data, and responding with a model to the presentation layer?

**Controller**

**Feedback :**

*The controller dictates the entire workflow of the application. Whenever a user sends a request to the application, dispatcher servlet invokes the corresponding controller. The controller is responsible for the business logic of any user request, invokes the required helper functions and upon processing data, it responds with the corresponding model data and the view template name.*

**Q >MVC Architecture**

Assume that a web application is being created using a Maven project and Springboot.  
A controller class was written to return a static HTML page named index.html stored at resources → templates in a Maven project.  
Following is the code for the controller class:

**@Controller**

**public** **class** **TestController** {

**@RequestMapping**("/showpage")

**@ResponseBody**

**public** String **webpage**() {

**return** "index";

}

}

Assuming all dependencies required are already added in the POM file(all dependencies used by you to create your hello world application using Spring MVC). When you run the application and request localhost:8080/showpage through a browser, you would see “index” written in the browser window. Why would you see “index” written in the browser window?

**ANS: The method is annotated with @ResponseBody**

**Feedback :**

*Because of the @ResponseBody annotation, the string returned by the controller method is not being intercepted by the view resolver. Instead, it is being returned to the client as plain text.*

**Dependencies**

Which of the following was not taken care of by dependencies in your Spring MVC hello world application?

Top of Form



Given a view name, returning the appropriate view.



Hosting the website on 8080 port of the server.



Routing a client request to the appropriate controller class.



**Routing a controller to the appropriate business/data logic.**

**Feedback :**

*For routing a controller to the appropriate business/data logic you specifically need to mention where you want to go to within the code itself. It is not automatically handled by maven dependencies.*

**Correct**

Bottom of Form

# Session Overview

Welcome to the session on MVC architecture.

## In this session

You will learn how to develop a technical blog website from scratch using the MVC architecture pattern. In the technical blog application, you will implement different features such as displaying all posts with the date of creation, creating a new blog post, deciding on the content of the post, setting the login feature and the registration feature, etc.  
   
It is advisable that you practice the project along with the videos and develop the technical blog website on your computer/laptop. This will improve your grasp over the code. Also, keep in mind that while working on the website project, you may come across different errors, which you are expected to debug on your own. However, you may use the **Discussion Forum** to post your doubts as they arise. Also, you may search on 'google.com' or 'stackoverflow.com' for discussions/resolutions related to similar errors. As a software engineer, it is important for you to improve your error debugging skills.

**You can find the GitHub commits of the project**[**here.**](https://github.com/upgrad-edu/TechnicalBlogApplication)

**Note:** In this module, you will learn about the development of the application until the third commit (third milestone). We recommend that you refer to this only when you really need it, and until then, just use the code provided, as part of the text in the segments and the videos, to create your application.

It is always a good practice for a developer to be comfortable with reading official documentation. Here are the links to the official documentation that you will need:

* [Spring Web MVC](https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html)
* [Maven](https://maven.apache.org/guides/)
* [Spring Boot](https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/html/)

# MVC Architecture - II

# Session Overview

Welcome to the session on MVC architecture.

## In this session

You will learn how to develop a technical blog website from scratch using the MVC architecture pattern. In the technical blog application, you will implement different features such as displaying all posts with the date of creation, creating a new blog post, deciding on the content of the post, setting the login feature and the registration feature, etc.  
   
It is advisable that you practice the project along with the videos and develop the technical blog website on your computer/laptop. This will improve your grasp over the code. Also, keep in mind that while working on the website project, you may come across different errors, which you are expected to debug on your own. However, you may use the **Discussion Forum** to post your doubts as they arise. Also, you may search on 'google.com' or 'stackoverflow.com' for discussions/resolutions related to similar errors. As a software engineer, it is important for you to improve your error debugging skills.

**You can find the GitHub commits of the project**[**here.**](https://github.com/upgrad-edu/TechnicalBlogApplication)

<https://github.com/upgrad-edu/TechnicalBlogApplication>

**Note:** In this module, you will learn about the development of the application until the third commit (third milestone). We recommend that you refer to this only when you really need it, and until then, just use the code provided, as part of the text in the segments and the videos, to create your application.

It is always a good practice for a developer to be comfortable with reading official documentation. Here are the links to the official documentation that you will need:

* [Spring Web MVC](https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html)
* [Maven](https://maven.apache.org/guides/)
* [Spring Boot](https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/html/)

# Technical Blog Application

In this session, you will learn how to develop a basic blogging website using the MVC architecture and Spring Boot framework. Let's have a quick look at all the features you will learn to develop as part of the web application in this module.

So, the features you will learn to develop in this module are:

1. The login feature
2. The registration feature
3. The create a post feature

**NOTE:** For now, you will not be integrating any database with your application. All data storage and retrieval operations for the application will be taken care of in the next module, i.e., Databases and ORM.

Now, let's start working on developing the technical blog web application. Before you start working on this, you should create a project and keep it ready, with basic configurations such as the POM file and the application class, as explained in the Hello World program.

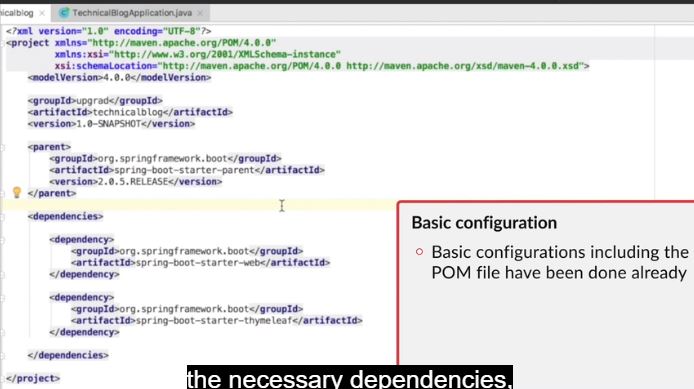
You can also download the following project file with basic configurations to start with:

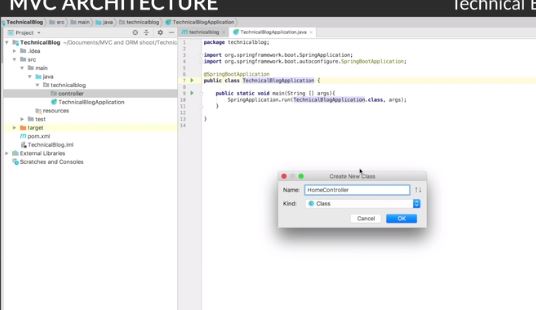
//PROJECT

In the next video, you will go through the following steps to develop the web application:

* Create a controller package and a controller within it.
* Create a model class to store a technical blog post.







Create a package controller and a class in it