**Assignment #2**

* **HTML :**
* HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages.
* Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus the link available on a webpage are called Hypertext.
* As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark up" a text document with tags that tell a Web browser how to structure it to display.
* Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.
* **CSS :**
* **C**ascading **S**tyle **S**heets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.
* CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.
* CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

## Advantages of CSS

* **CSS saves time −** You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
* **Pages load faster −** If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
* **Easy maintenance −** To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
* **Superior styles to HTML −** CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
* **Multiple Device Compatibility −** Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
* **Global web standards −** Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.
* **Offline Browsing −** CSS can store web applications locally with the help of an offline catch Using of this, we can view offline websites. The cache also ensures faster loading and better overall performance of the website.
* **Platform Independence −** The Script offer consistent platform independence and can support latest browsers as well.
* **JavaScript :**
* JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.
* JavaScript is a lightweight, interpreted programming language.
* Designed for creating network-centric applications.
* Complementary to and integrated with Java.
* Complementary to and integrated with HTML.
* Open and cross-platform
* Advantages of JavaScript
* The merits of using JavaScript are −
* Less server interaction − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* Immediate feedback to the visitors − They don't have to wait for a page reload to see if they have forgotten to enter something.
* Increased interactivity − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
* Richer interfaces − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.
* **Bootstrap :**
* Bootstrap is a sleek, intuitive, and powerful, mobile first front-end framework for faster and easier web development. It uses HTML, CSS and JavaScript.
* Mobile first approach − Bootstrap 3, framework consists of Mobile first styles throughout the entire library instead them of in separate files.
* Browser Support − It is supported by all popular browsers.
* Easy to get started − With just the knowledge of HTML and CSS anyone can get started with Bootstrap. Also the Bootstrap official site has a good documentation.
* Responsive design − Bootstrap's responsive CSS adjusts to Desktops, Tablets and Mobiles. More about the responsive design is in the chapter [Bootstrap Responsive Design.](https://www.tutorialspoint.com/bootstrap/bootstrap_responsive_utilities.htm)
* **Node.JS :**
* Node.js is a platform built on [Chrome's JavaScript runtime](https://code.google.com/p/v8/) for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

## Features

* **Asynchronous and Event Driven** − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
* **Very Fast** − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* **Single Threaded but Highly Scalable** − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
* **No Buffering** − Node.js applications never buffer any data. These applications simply output the data in chunks.
* **AngularJS :**
* AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you currently must write. And it all happens within the browser, making it an ideal partner with any server technology.

## Features

* AngularJS is a powerful JavaScript based development framework to create RICH Internet Application(RIA).
* AngularJS provides developers options to write client side application (using JavaScript) in a clean MVC(Model View Controller) way.
* Application written in AngularJS is cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
* AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache License version 2.0.

## Core Features

* **Data-binding** − It is the automatic synchronization of data between model and view components.
* **Scope** − These are objects that refer to the model. They act as a glue between controller and view.
* **Controller** − These are JavaScript functions that are bound to a particular scope.
* **Services** − AngularJS come with several built-in services for example $https: to make a XML Http Requests. These are singleton objects which are instantiated only once in app.
* **Filters** − These select a subset of items from an array and returns a new array.
* **Directives** − Directives are markers on DOM elements (such as elements, attributes, CSS, and more). These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives (NgBind, NgModel...)
* **Templates** − These are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".
* **Routing** − It is concept of switching views.
* **Model View Whatever** − MVC is a design pattern for dividing an application into different parts (called Model, View and Controller), each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-View Model). The Angular JS team refers it humorously as Model View Whatever.
* **Deep Linking** − Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
* **Dependency Injection** − AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.
* **Servlets :**
* Java Servlets are programs that run on a Web or Application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.
* Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
* Java Servlets often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But Servlets offer several advantages in comparison with the CGI.
* Performance is significantly better.
* Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request.
* Servlets are platform-independent because they are written in Java.
* Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. So servlets are trusted.
* The full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already.
* **Spring Framework :**
* Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product.
* Spring is organized in a modular fashion. Even though the number of packages and classes are substantial, you must worry only about the ones you need and ignore the rest.
* Spring does not reinvent the wheel, instead it truly makes use of some of the existing technologies like several ORM frameworks, logging frameworks, JEE, Quartz and JDK timers, and other view technologies.
* Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Furthermore, by using Java Bean style POJOs, it becomes easier to use dependency injection for injecting test data.
* Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks such as Struts or other over-engineered or less popular web frameworks.
* Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.
* Lightweight IoC containers tend to be lightweight, especially when compared to EJB containers, for example. This is beneficial for developing and deploying applications on computers with limited memory and CPU resources.
* Spring provides a consistent transaction management interface that can scale down to a local transaction (using a single database, for example) and scale up to global transactions (using JTA, for example).

**Dependency Injection (DI)**

* The technology that Spring is most identified with is the Dependency Injection (DI) flavor of Inversion of Control. The Inversion of Control (IoC) is a general concept, and it can be expressed in many ways. Dependency Injection is merely one concrete example of Inversion of Control.
* When writing a complex Java application, application classes should be as independent as possible of other Java classes to increase the possibility to reuse these classes and to test them independently of other classes while unit testing. Dependency Injection helps in gluing these classes together and at the same time keeping them independent.
* What is dependency injection exactly? Let's look at these two words separately. Here the dependency part translates into an association between two classes. For example, class A is dependent of class B. Now, let's look at the second part, injection. All this means is, class B will get injected into class A by the IoC.
* Dependency injection can happen in the way of passing parameters to the constructor or by post-construction using setter methods. As Dependency Injection is the heart of Spring Framework, we will explain this concept in a separate chapter with relevant example.

**Aspect Oriented Programming (AOP)**

* One of the key components of Spring is the Aspect Oriented Programming (AOP) framework. The functions that span multiple points of an application are called cross-cutting concerns and these cross-cutting concerns are conceptually separate from the application's business logic. There are various common good examples of aspects including logging, declarative transactions, security, caching, etc.
* The key unit of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. DI helps you decouple your application objects from each other, while AOP helps you decouple cross-cutting concerns from the objects that they affect.
* The AOP module of Spring Framework provides an aspect-oriented programming implementation allowing you to define method-interceptors and point cuts to cleanly decouple code that implements functionality that should be separated. We will discuss more about Spring AOP concepts in a separate chapter.
* **Hibernate :**
* Hibernate is an Object-Relational Mapping(ORM) solution for JAVA
* Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.
* Hibernate sits between traditional Java objects and database server to handle all the work in persisting those objects based on the appropriate O/R mechanisms and patterns.

Hibernate Advantages

* Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code.
* Provides simple APIs for storing and retrieving Java objects directly to and from the database.
* If there is change in Database or in any table then the only need to change XML file properties.
* Abstract away the unfamiliar SQL types and provide us to work around familiar Java Objects.
* Hibernate does not require an application server to operate.
* Manipulates Complex associations of objects of your database.
* Minimize database access with smart fetching strategies.
* Provides Simple querying of data.
* **Rest :**
* REST stands for **RE**presentational **S**tate **T**ransfer. REST is a web standards based architecture and uses HTTP Protocol for data communication. It revolves around resources where every component is a resource and a resource is accessed by a common interface using HTTP standard methods.
* In REST architecture, a REST Server simply provides access to resources and the REST client accesses and presents the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource like Text, JSON and XML. JSON is now the most popular format being used in Web Services.
* HTTP Methods
* The following HTTP methods are most commonly used in a REST based architecture.
* GET − Provides a read only access to a resource.
* PUT − Used to create a new resource.
* DELETE − Used to remove a resource.
* POST − Used to update an existing resource or create a new resource.
* OPTIONS − Used to get the supported operations on a resource.