**DAY-2**

**Web development Application**

* java core (coding) – 10k line of code
* Technologies (Servlet/JSP) – 6k line code
* Spring
  + Readymade code

Developer works on functional logic.

Website : <http://spring.io/>

Springboot – one project build on the top of the spring project.

* Rapid or fast development.

Spring and springboot wrote code to make any application to web application , and provided that code into jar files.

**Difference Between Spring and Spring Boot:**

| **Feature** | **Spring** | **Spring Boot** |
| --- | --- | --- |
| **Configuration** | Requires manual setup | Auto-configured |
| **Server** | Needs external server (Tomcat, etc.) | Comes with embedded servers |
| **Dependency Management** | Requires managing dependencies manually | Uses pre-defined starter dependencies |
| **Development Speed** | Slower due to setup | Faster due to less boilerplate code |
| **Microservices Support** | Requires additional setup | Designed for microservices |

**Dependency management:**

* It may possible that jar file will depend on other jar files to manage this we introduced.
* **Build Tools** : Maven , Gradle
* **Starters** :
  + Starter is a pre-configured set of dependencies that help developers quickly add functionality to their applications without manually configuring multiple libraries.
  + Include essential libraries for common use cases like web applications, database connectivity, security, etc.
* **Auto Configuration**:

**Spring MVC (Model-View-Controller)**

* Separate Frontend and Backend Code.
* Provides a **flexible** and **powerful** way to develop web applications by separating concerns into different components.

**Key Components of Spring MVC:**

1. **Model (M)** – Represents the business logic and data handling.
2. **View (V)** – Handles UI representation (JSP, Thymeleaf, etc.).
3. **Controller (C)** – Manages requests, processes data, and returns a response.

**Spring MVC Workflow:**

1️ **Client Request** → Sent to **DispatcherServlet**  
2️ **DispatcherServlet** → Passes request to the **Controller**  
3️ **Controller** → Processes request and interacts with the **Model**  
4️ **Model** → Retrieves or processes data and sends it back  
5️ **Controller** → Selects the appropriate **View**  
6️ **View (JSP, Thymeleaf, etc.)** → Generates response  
7️ **Response Sent** → Back to the client

**API** : (**Application Programming Interface)**

* **Set of rules and protocols** that allows two software applications to communicate with each other.
* **Ex.** To communicates between frontend and backend.

**Key Components of an API:**

|  |  |
| --- | --- |
| **Endpoint** | * URL where the API is accessed **(/users/1).** |
| **Request** | * Sent by the client to ask for data **(GET /users).** |
| **Response** | * Data returned by the server **(JSON/XML format).** |
| **HTTP Methods** | * **GET** (fetch), **POST** (create), **PUT** (update), **DELETE** (remove). |
| **Authentication** | * API keys, OAuth, JWT to secure API access. |

**Web API’s** :

* Allows communication between applications over the internet using HTTP requests.
* It enables web, mobile, or desktop applications to exchange data with servers.
* **Ex :** Like when click on login button, it will handle by backend code and this intimation will go to backend by API.

**Web services** : It enables to communicate with one service to another service(from frontend to backend).

* It also done by invoking Web API’s.

**Standards to develop API:**

1. **REST (Representational State Transfer API**
   * Most common, uses HTTP methods like GET, POST, PUT, and DELETE.
2. **SOAP API (Simple Object Access Protocol)**
   * Uses XML-based messaging for strict communication.
3. **GraphQL API**

**Spring Tool Suite (STS) Overview:**

* **IDE (Integrated Development Environment)** designed specifically for developing **Spring applications**.
* Built on **Eclipse** and provides tools for **Spring Boot, Spring MVC, Spring Data, and microservices development**.

**Making project using spring.io**

**1.** making project from start.spring.io

**Group ID –** unique identifier for group of project like com.example

* Company name

**Artifact ID** – Name of the specific project or module

* My-spring-app

**2.**  unzip folder

**3.** open it into IDE

**4.**  Explore pom.xml file

**5.** Add Controller for mapping /hello

**6.**  Run using maven commnds**:**

* **clean** → Deletes old compiled files
* **compile** → Compiles source code
* **package** → Creates .jar or .war file
* **install** → Installs package to local repo
* **test** → Runs unit tests
* **spring-boot:run** → Runs Spring Boot app

**7.** Running application

* we need server to run web application.
* **Tomcat** is provided embedded server with **Spring web** dependency.
* tomcat executes and run application on 8080.

**8.** Access the application services

* Using url, other can access your web api.
* url **syntax**:
  + http://<machine-ip>:<port>/<path>
  + <http://localhost:8080/hello>
  + W
  + <path>?<param-name>=<param-value>&>?<param-name>=<param-value>
  + <http://8080//add?a=5&b=9>

9. Trying Post method from Browser

* Debugging in browser
  + Inspect -> network tab
* Hard reload
  + Inspect -> right click on refresh -> empty cache and hard reload
* Browser passes data as Get .. for post it will fail.
* How you will all these api’s?
* Postman

10. Adding Dependency

* Devtools – Auto reload on changes occurring.

**DNS (Domain Name System)**

* Translates human-readable domain names (e.g., www.google.com) into IP addresses (e.g., 142.250.182.4) that computers use to communicate.

DNS Provider – GoDaddy, namecheap, Cloudflare

<http://studio.youtube.com/> 🡪 http://<IP>

* command to see attached IP to DNS
  + vlookup google.com

default assigned port:

http – 80

https - 443