Spring Web MVC module – This module in Spring Framework will help us to create a Spring based web Application. That will interact with any database using any ORM framework.

Library – Will help us to resolve some challenge while developing applications ( jars, bootstrap, jquery, ajax, jstl)

Tools – IDE (Integrated Development Environment, Jenkins, Docker, SonarQube, Post Man) – Used for a particular purpose

Framework – Help us to create and manage complete Application. [To manage, deploy, build and run the entire application we use frameworks. It contains various modules or packages]

Spring, Struts, Spring Boot --- Java Based Frameworks (Back End Frameworks)

Angular, React, Vue, Express --- JavaScript based frameworks (Front End Frameworks)

Spring Has Many Modules

1. Spring Core [ core, beans, context, spel ]
2. Spring Web MVC [servlet, jsp, portlets, web socket]
3. Spring Data [ jdbc, orm, oxm, jms]
4. Spring Transaction
5. Spring Security
6. Spring Test
7. Spring AOP [Aspect Oriented Programming] -- AspectJ (point cuts, advice, joins)

Spring uses two important design patterns (IoC – Inversion of Control, DI – Dependency Injection)

@Autowired [ Inject the bean at runtime ]

Using a xml configuration file (applicationContext.xml) or annotation(@Bean) Beans will be automatically created by Spring Framework.

Bean Class – Is a type of Java Class where it consist of properties , getters, setters & constructors.

@Enity (If this annotation is added to a bean class, then it will be an Entity Bean class]

@Entity – This annotation belongs to JPA (Java Persistence API)

JDBC – Java Database Connectivity API – This will give us standard way of interacting with any Database [ Connection, DriverManager, Statement, ResultSet ]

JDBC is a specification – Most of the jdbc API’s are interfaces (abstract methods) --- Driver Jar file is the implementation of JDBC API for a particular DB.

JPA is a Specification – It simplifies the process of interacting to the Database. Persistence.xml file will have all the details about the DB connection (url, username, password, driver class name)

SessionFactory & Session Object to Manage the Transaction.

With the help of Bean class, we will do the CRUD operation. [No need to write SQL queries]

JPA/Hibernate – JPQL/HQL (DB independent Query – We will be using the Entity Bean class name rather than the db table name)

JPQL – Java Persistence Query Language

HQL – Hibernate Query Language

**In Spring MVC ---- Dispatcher Servlet (Java Class with embedded HTML code) --- Front Controller**

For each JSP file, a corresponding servlet class will be created automatically by the web server (tomcat) index.jsp (index\_jsp.java --- servlet class name)

WebMVC

1. Web.xml (Entry point) – sdnext - inti parameter --- \*.html
2. Index.html (welcome file) – apply sdnext =index
3. Search for a mapping in controller class – index.html – “index”
4. Sdnext-servlet – apply prefix & suffix to the output of controller method
5. Open /WEB-INF/views/index.jsp and process then display the output in browser

Same Web Application using Spring Boot

Spring Boot is the child of Spring Framework.

Why SpringBoot?

* The main challenge with Spring Framework is xml configuration file. (manually restart the server and manage the deployment)
* SpringBoot simplifies the process of creating Spring based Enterprise Application
* Creating SOAP & REST based Web Services using Spring Boot

WebService -- A Service Using Internet (URI) – Executing a method using a URI

URL – Uniform Resource Locator (.jsp/.html/.git/.do/.asp/.php)

URI – Uniform Resource Identifier (/users/id)

Spring Boot App Creation

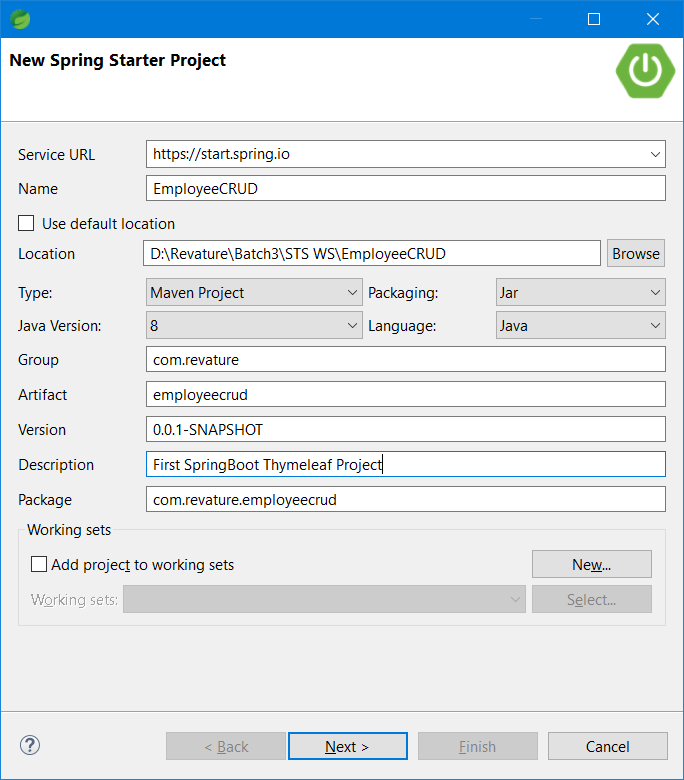
1. Using Spring Initializr (start.spring.io – Web based method) – using the browser you will create spring boot application and download the zip file of the application
2. Using STS (Spring Tool Suite) – [Recommended]
3. Using SpringBoot CLI

Adv SpringBoot

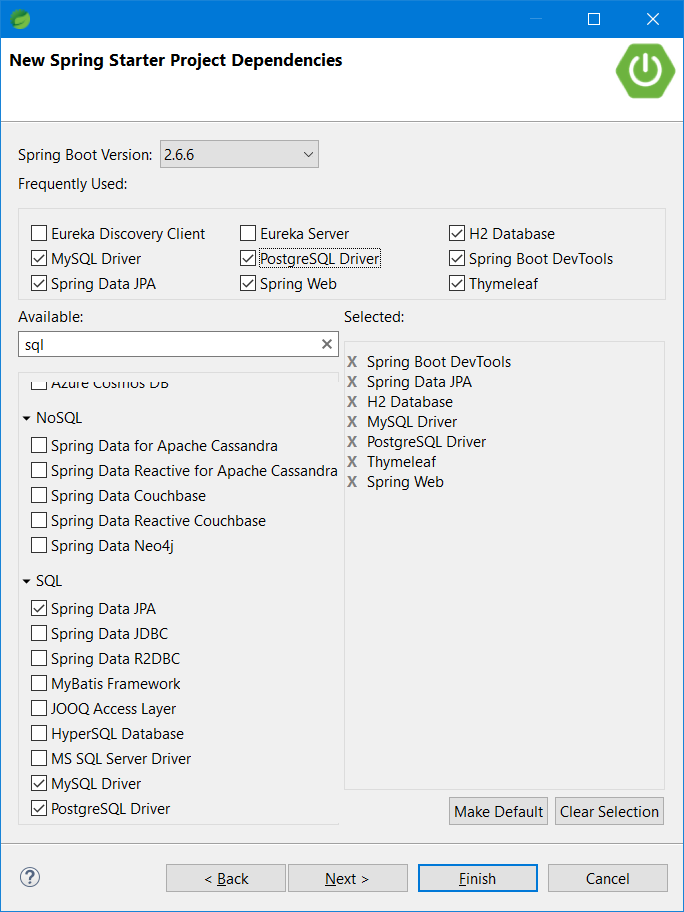
1. Is a Opinionated framework [Add Annotations or property to configure SpringBoot App]
2. Choice of build tool [maven/graddle]
3. Language [Java/Kotlin/Groovy]
4. Packaging [Jar/War]
5. It contains embedded Tomcat server to deploy Web based applications
6. @SpringBootApplication – (@SpringBootApplication) -@[SpringBootConfiguration](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot%5C/2.6.6%5C/spring-boot-2.6.6.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.boot=/=/maven.artifactId=/spring-boot=/=/maven.version=/2.6.6=/=/maven.scope=/compile=/%3Corg.springframework.boot(SpringBootConfiguration.class%E2%98%83SpringBootConfiguration)  
   @[EnableAutoConfiguration](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/2.6.6%5C/spring-boot-autoconfigure-2.6.6.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.boot=/=/maven.artifactId=/spring-boot-autoconfigure=/=/maven.version=/2.6.6=/=/maven.scope=/compile=/%3Corg.springframework.boot.autoconfigure(EnableAutoConfiguration.class%E2%98%83EnableAutoConfiguration)  
   @[ComponentScan](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(ComponentScan.class%E2%98%83ComponentScan)([excludeFilters](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(ComponentScan.class%E2%98%83ComponentScan~excludeFilters)={@[Filter](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter)([type](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter~type)=[CUSTOM](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(FilterType.class%E2%98%83FilterType%5ECUSTOM), [classes](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter~classes)={[TypeExcludeFilter](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot%5C/2.6.6%5C/spring-boot-2.6.6.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.boot=/=/maven.artifactId=/spring-boot=/=/maven.version=/2.6.6=/=/maven.scope=/compile=/%3Corg.springframework.boot.context(TypeExcludeFilter.class%E2%98%83TypeExcludeFilter).class}), @[Filter](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter)([type](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter~type)=[CUSTOM](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(FilterType.class%E2%98%83FilterType%5ECUSTOM), [classes](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/spring-context%5C/5.3.18%5C/spring-context-5.3.18.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework=/=/maven.artifactId=/spring-context=/=/maven.version=/5.3.18=/=/maven.scope=/compile=/%3Corg.springframework.context.annotation(ComponentScan$Filter.class%E2%98%83Filter~classes)={[AutoConfigurationExcludeFilter](eclipse-javadoc:%E2%98%82=employeecrud/C:%5C/Users%5C/Siva%5C/.m2%5C/repository%5C/org%5C/springframework%5C/boot%5C/spring-boot-autoconfigure%5C/2.6.6%5C/spring-boot-autoconfigure-2.6.6.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.boot=/=/maven.artifactId=/spring-boot-autoconfigure=/=/maven.version=/2.6.6=/=/maven.scope=/compile=/%3Corg.springframework.boot.autoconfigure(AutoConfigurationExcludeFilter.class%E2%98%83AutoConfigurationExcludeFilter).class})})

Creating a SpringBoot Application

* Open STS
* Create a new Spring Starter Project [File 🡪 New 🡪 Spring Starter Project]



* Fill all the fields with relevant values and click Next



* Select the required dependencies and click Finish
* Open application.properties file and add the following

spring.thymeleaf.cache=false

spring.thymeleaf.suffix: .html

server.port=8090

# h2 database configuration properties

#spring.datasource.url=jdbc:h2:mem:testdb

#spring.datasource.driver-class-name=org.h2.Driver

#spring.datasource.username=sa

#spring.datasource.password=

#spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

#

#spring.h2.console.enabled=true

# MySQL Database Configuration

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/revature1

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

#Postgres database configuration

#spring.datasource.url= jdbc:postgresql://localhost:5432/testdb

#spring.datasource.username= postgres

#spring.datasource.password= 123

#

#spring.jpa.properties.hibernate.jdbc.lob.non\_contextual\_creation= true

#spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.PostgreSQLDialect

#

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

## Hibernate ddl auto (create, create-drop, validate, update)

#spring.jpa.hibernate.ddl-auto= update

#

#logging.level.org.springframework=info

#logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output

#logging.level.org.hibernate.SQL=trace

#logging.level.org.hibernate.type.descriptor.sql=trace

# Log pattern

#logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

* Add index.html & student-data.html under resources/templates folder

<!DOCTYPE html>

<html lang=*"en"*>

<head>

<title>Index Page</title>

</head>

<body>

<form action=*"save"* method=*"post"*>

<table>

<tr>

<td><label for=*"id"*>Student ID</label></td>

<td><input type=*"text"* name=*"id"*></input></td>

</tr>

<tr>

<td><label for=*"name"*>Student Name</label></td>

<td><input type=*"text"* name=*"name"*></input></td>

</tr>

<tr>

<td></td>

<td><input type=*"submit"* value=*"Submit"*></input></td>

</tr>

</table>

</form>

</body>

</html>

<html xmlns:th=*"https://thymeleaf.org"*>

<table>

<tr>

<td><h4>Student ID:</h4></td>

<td><h4 th:text=*"${student.id}"*></h4></td>

</tr>

<tr>

<td><h4>Student Name:</h4></td>

<td><h4 th:text=*"${student.name}"*></h4></td>

</tr>

</table>

</html>

* StudentController.java

**package** com.revature.employeecrud.controller;

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

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

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

**import** org.springframework.web.servlet.ModelAndView;

**import** com.revature.employeecrud.model.Student;

@Controller

**public** **class** StudentController {

@RequestMapping("/")

**public** String index() {

**return** "index";

}

@RequestMapping(value = "/save", method = RequestMethod.***POST***)

**public** ModelAndView save(@ModelAttribute Student student) {

ModelAndView modelAndView = **new** ModelAndView();

modelAndView.setViewName("student-data");

modelAndView.addObject("student", student);

**return** modelAndView;

}

}

* Student.java

**package** com.revature.employeecrud.model;

**public** **class** Student {

**private** **int** id;

**private** String name;

**public** Student(**int** id, String name) {

**super**();

**this**.id = id;

**this**.name = name;

}

**public** Student() {

**super**();

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

StudentService.java

**package** com.revature.employeecrud.service;

**import** java.util.List;

**import** com.revature.employeecrud.model.Student;

**public** **interface** StudentService {

**public** List<Student> findAll();

**public** Student findById(**int** id);

**public** **void** add(Student student);

**public** **void** update(**int** id, Student student);

**public** **void** deleteById(**int** id);

}

StudentServiceImpl.java

**package** com.revature.employeecrud.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.revature.employeecrud.model.Student;

**import** com.revature.employeecrud.repo.StudentRepository;

@Service

**public** **class** StudentServiceImpl **implements** StudentService {

@Autowired

StudentRepository studentRepository;

@Override

**public** List<Student> findAll() {

// **TODO** Auto-generated method stub

**return** studentRepository.findAll();

}

@Override

**public** Student findById(**int** id) {

// **TODO** Auto-generated method stub

**return** studentRepository.getById(id);

}

@Override

**public** **void** add(Student student) {

// **TODO** Auto-generated method stub

studentRepository.save(student);

}

@Override

**public** **void** update(**int** id, Student student) {

// **TODO** Auto-generated method stub

studentRepository.save(student);

}

@Override

**public** **void** deleteById(**int** id) {

// **TODO** Auto-generated method stub

studentRepository.deleteById(id);

}

}

StudentRepo.java

**package** com.revature.employeecrud.repo;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.revature.employeecrud.model.Student;

**public** **interface** StudentRepository **extends** JpaRepository<Student, Integer> {

}