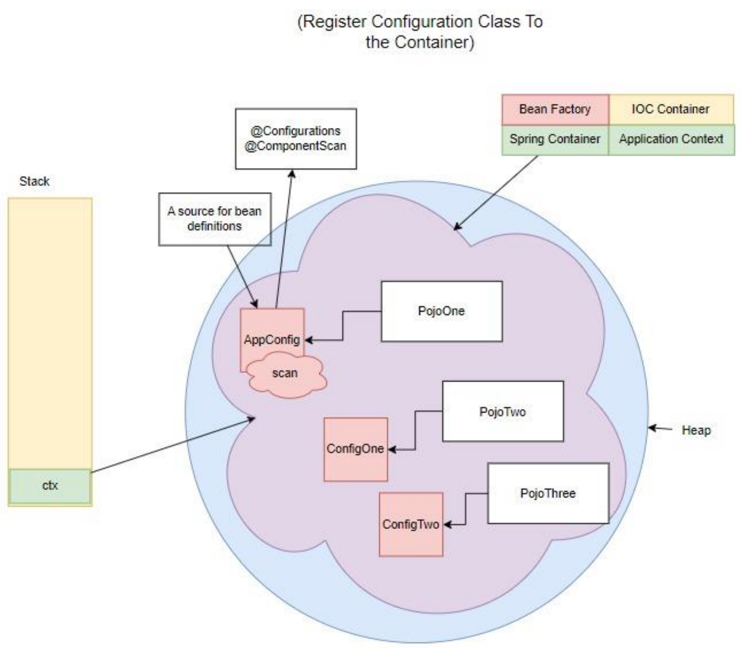
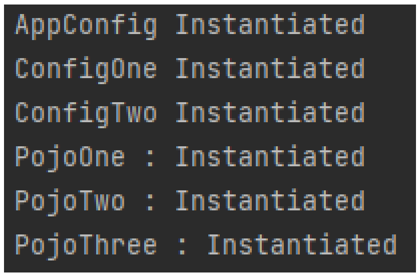
1



2

```
public class AppInitializer {
  public static void main(String[] args) {
      AnnotationConfigApplicationContext ctx = new
            AnnotationConfigApplicationContext();

      ctx.register(AppConfig.class);
      ctx.register(ConfigOne.class);
      ctx.register(ConfigTwo.class);

      ctx.refresh();
      ctx.registerShutdownHook();
  }
}
```

3

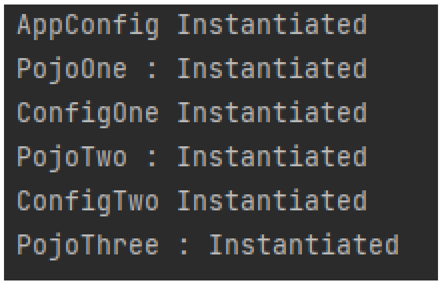


```
AppConfig Instantiated
ConfigOne Instantiated
ConfigTwo Instantiated
PojoOne : Instantiated
PojoTwo : Instantiated
PojoThree : Instantiated
```

4

```
@Configuration
@ComponentScan(basePackages = "lk.ijse.spring.pojo")
@Import({ConfigOne.class,ConfigTwo.class})
public class AppConfig {
  public AppConfig() {
      System.out.println("AppConfig Instantiated");
  }
}
```

5



```
AppConfig Instantiated
PojoOne : Instantiated
ConfigOne Instantiated
PojoTwo : Instantiated
ConfigTwo Instantiated
PojoThree : Instantiated
```

6

```
@Configuration
@ComponentScan(basePackages = "lk.ijse.spring.pojo")
public class AppConfig {
  public AppConfig() {
      System.out.println("AppConfig Instantiated");
  }

  // Bean method එකක් through value එක set කරන්න,
  @Bean
  public String setName() {
      return "John";
  }
}
```

7

```
@Component
public class Customer {
  public Customer(@Value("John") String name) {
     System.out.println("Customer : Instantiated : " + name);
  }
}
```

8

```
@Component
public class Item implements InitializingBean {

   @Value("Test")
   public String name;

   public Item(){
       System.out.println("Item: Instantiated");
       // Instantiate step එකේ ඉන්නෙ, populate properties වෙලා නෑ
       System.out.println(name); // output-> null
   }

   @Override
   public void afterPropertiesSet() throws Exception {
       // Bean එකේ ready වුනාම output එක ගන්න පුලුවන්.
       System.out.println(name); // output-> Test
   }
}
```

9

```
@Component
public class Customer {
   @Autowired
   public Customer(@Value("Tommy") String name) {
       System.out.println("Constructor 1");
   }

   public Customer(@Value("Tommy,John") String names[]) {
       System.out.println("Constructor 2");
   }
}
```

10

```
@Component
public class Customer {
   @Autowired(required = false)
   public Customer(@Value("Tommy") String name) {
       System.out.println("Constructor 1");
   }

   @Autowired(required = false)
   public Customer(@Value("Tommy") String name, @Value("25") int age) {
       System.out.println("Constructor 2");
   }
}
```

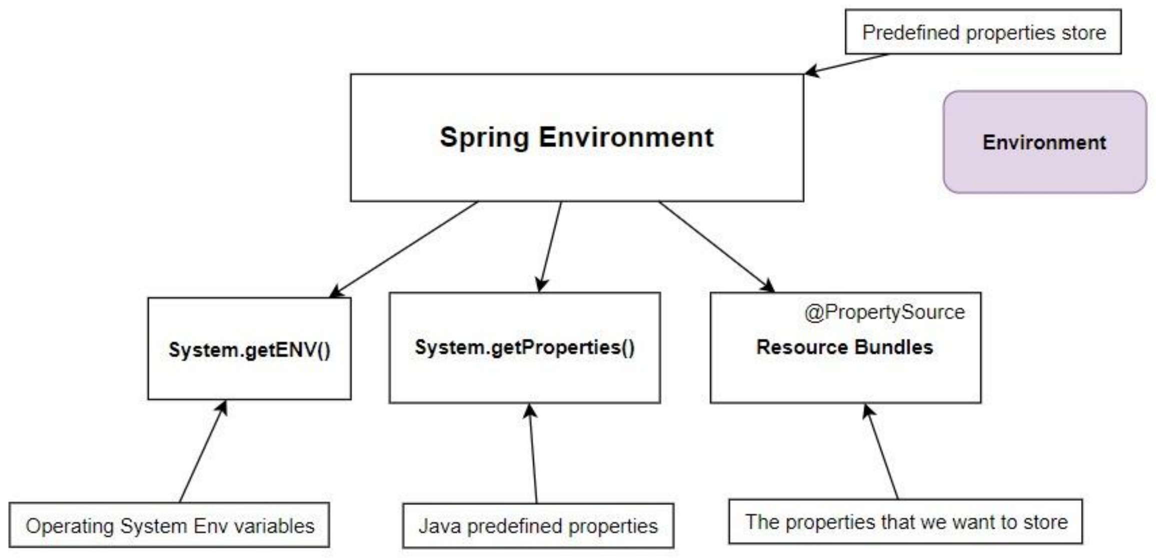
11

```
Map<String, String> getenv = System.getenv();
for (String key : getenv.keySet()) {
   String value = getenv.get(key);
   System.out.println(key+" : "+value);
}
```

12

```
Properties properties = System.getProperties();
for (Object key : properties.keySet()) {
  Object value = properties.get(key);
  System.out.println(key+" : "+value);
}
```

13


Predefined properties store
Spring Environment
Environment
System.getENV()
System.getProperties()
@PropertySource
Resource Bundles
Operating System Env variables
Java predefined properties
The properties that we want to store

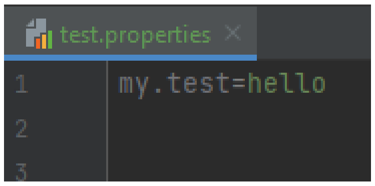
14

```
@Component
public class TestBean implements InitializingBean {
  @Value("${COMPUTERNAME}") // property placeholder
  private String userName;

  @Override
  public void afterPropertiesSet() throws Exception {
     System.out.println(userName); // John
  }
};
```

15

```
@Component
public class TestBean implements InitializingBean {
  @Autowired
  Environment environment;

  @Override
  public void afterPropertiesSet() throws Exception {
     String username = environment.getProperty("COMPUTERNAME");
     System.out.println(username); // John
  }
}
```

16



```
my.test=hello
```

17

```
@Configuration
@ComponentScan(basePackages = "lk.ijse.spring.pojo")
@PropertySource("classpath:test.properties")
public class AppConfig {

}
```

18

```
@Component
public class PojoOne implements InitializingBean {
   @Value("${my.test}") // property placeholder
   private String s;

   @Autowired
   Environment environment;

   @Override
   public void afterPropertiesSet() throws Exception {
       System.out.println(s); // hello

       String property = environment.getProperty("my.test");
       System.out.println(property); // hello
   }
}
```

19

```
@Value("${my.test.wrong}") // property placeholder
private String s;
. . .
@Override
public void afterPropertiesSet() throws Exception {
    System.out.println(s); // ${my.test.wrong}
}
```

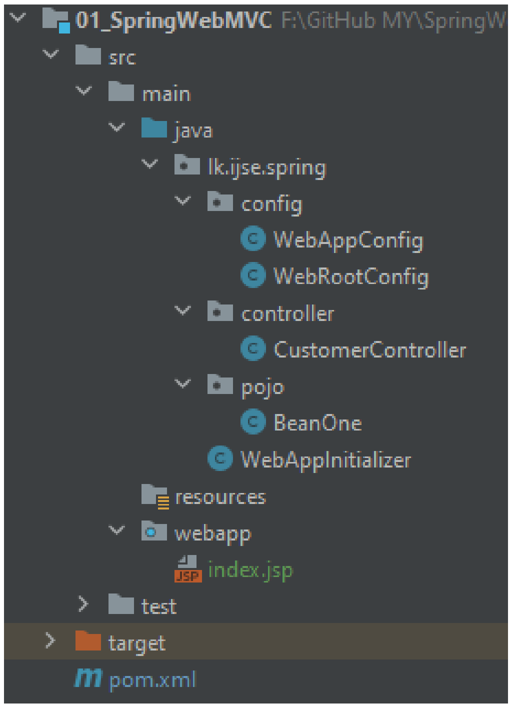
20

```
@Autowired
Environment environment;
. . .
@Override
public void afterPropertiesSet() throws Exception {
    String property = environment.getProperty("my.test.wrong");
    System.out.println(property); // null
}
```

21

```
@Autowired
Environment environment;
. . .
@Override
public void afterPropertiesSet() throws Exception {
    String reqProperty = environment.getRequiredProperty("my.test.wrong");
    System.out.println(property); // throws exception
}
```

22



```
01_SpringWebMVC F:\GitHub MY\SpringW
  src
    main
      java
        lk.ijse.spring
          config
            WebAppConfig
            WebRootConfig
          controller
            CustomerController
          pojo
            BeanOne
          WebAppInitializer
      resources
      webapp
        index.jsp
    test
  target
  pom.xml
```

24

```
public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
   @Override
   protected Class<?>[] getRootConfigClasses() {
       return new Class[]{WebRootConfig.class};
   }

   @Override
   protected Class<?>[] getServletConfigClasses() {
       return new Class[]{WebAppConfig.class};
   }

   @Override
   protected String[] getServletMappings() {
       return new String[]{"/"};
   }
}
```

23

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>lk.ijse.spring</groupId>
    <artifactId>01_SpringWebMVC</artifactId>
    <version>1.0.0</version>
    <packaging>war</packaging>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
    </properties>

    <dependencies>
        <!--Spring Web MVC-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.3.30</version>
        </dependency>

        <!--Tomcat-->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</project>
```

25

```
@Configuration
public class WebRootConfig {
    public WebRootConfig(){
        System.out.println("WebRootConfig : Instantiated");
    }
}
```

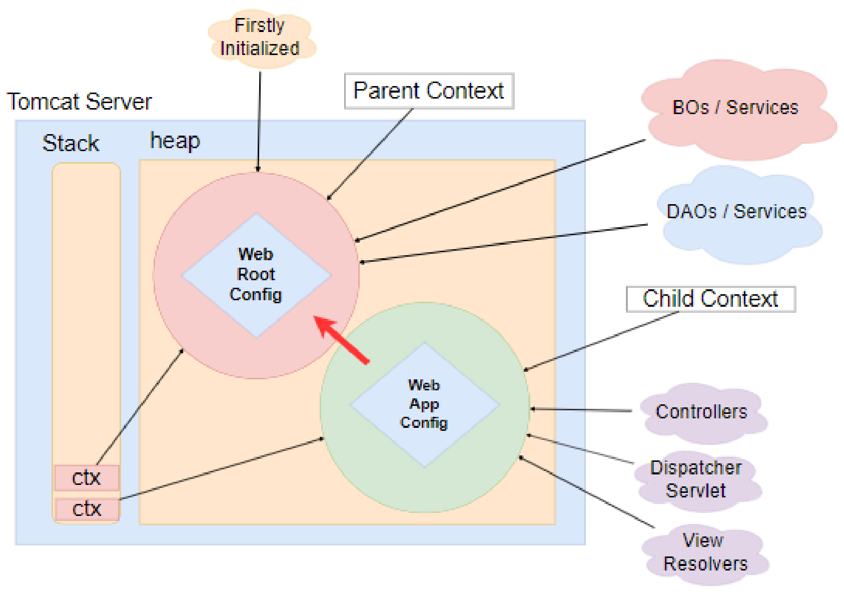
26

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = {"lk.ijse.spring.pojo","lk.ijse.spring.controller"})
public class WebAppConfig {
    public WebAppConfig() {
        System.out.println("WebAppConfig : Instantiated");
    }
}
```

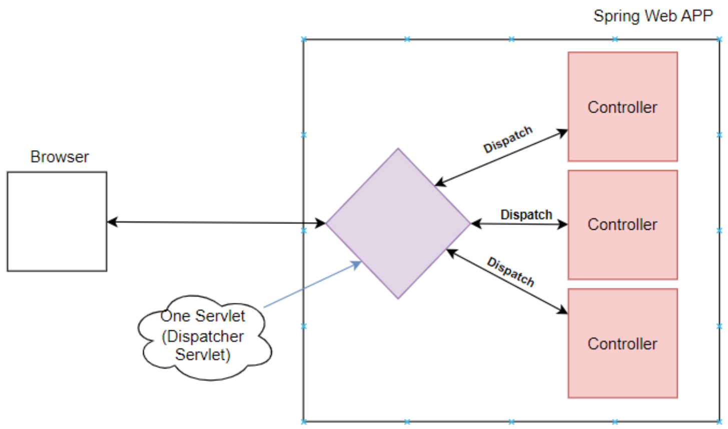
27

```
@RestController
@RequestMapping("/customer")
public class CustomerController {
  @GetMapping
  public String helloSpring() {
     return "Hello Spring";
  }
}
```

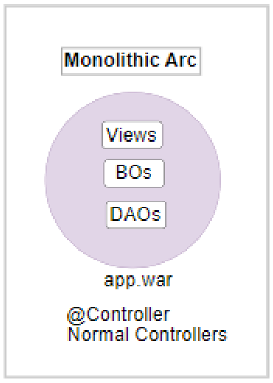
28



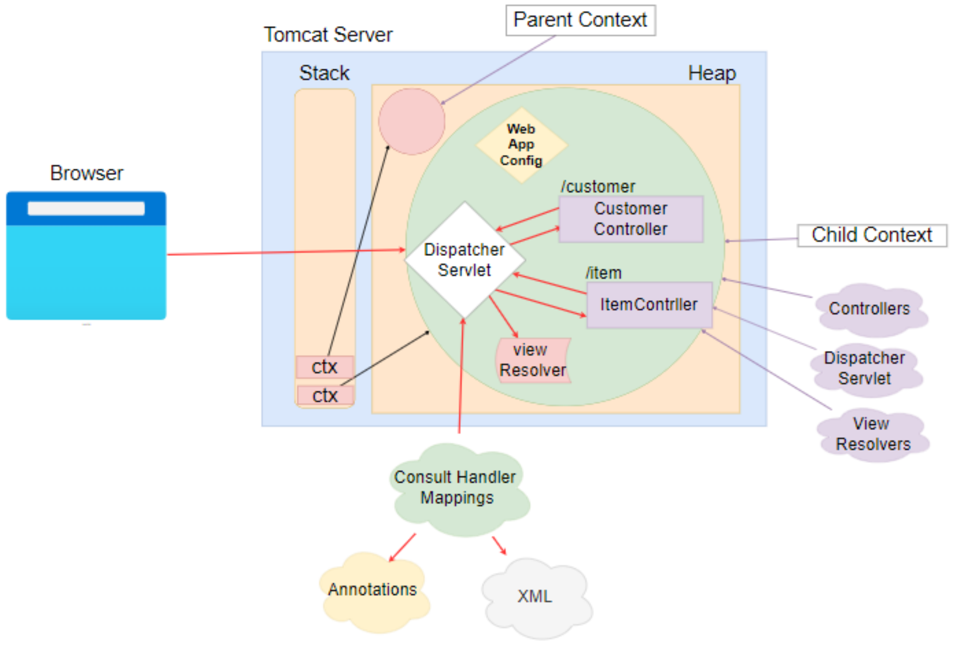
29



30



31



32

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "lk.ijse.spring.controller")
public class WebAppConfig {
   // create view resolver
   @Bean
   public InternalResourceViewResolver viewResolver() {
       InternalResourceViewResolver vr = new InternalResourceViewResolver();
       vr.setPrefix("/");
       vr.setSuffix(".jsp");
       return vr;
   }
}
```

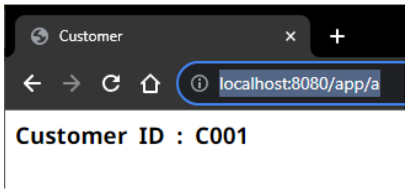
33

```
@Controller
@RequestMapping("/a")
public class CustomerController {
  @GetMapping
  public ModelAndView test(){
    ModelAndView mv = new ModelAndView("/customer");//.jsp name in webapps

    mv.addObject("Id","C001");
    return mv;
  }
}
```

34

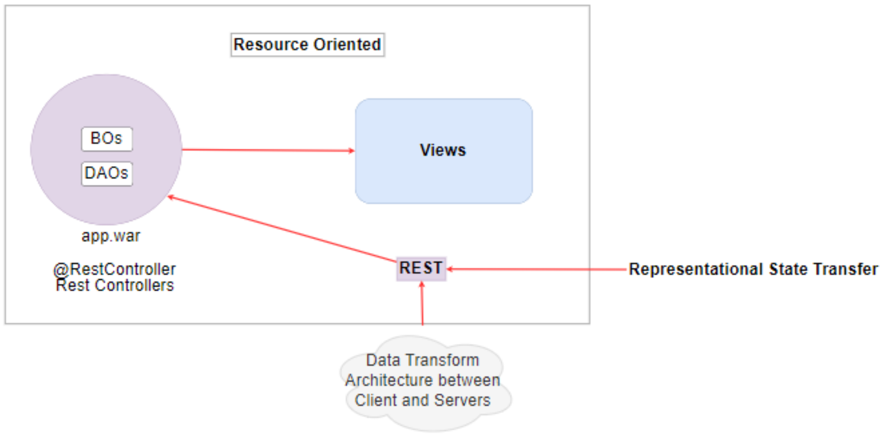
```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
  <title>Customer</title>
</head>
<body>
<h3>Customer ID : ${Id} </h3>
</body>
</html>
```

35



```
Customer ID : C001
```

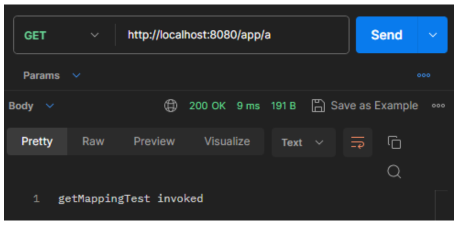
36


Resource Oriented
BOs
DAOs
app.war
@RestController
Rest Controllers
Views
REST
Representational State Transfer
Data Transform Architecture between Client and Servers

37

```
@RestController
@RequestMapping("/a")
public class CustomerController {
  @GetMapping
  public String getMappingTest(){
      return "getMappingTest invoked";
  }

  @PostMapping
  public String postMappingTest(){
      return "postMappingTest invoked";
  }

  @PutMapping
  public String putMappingTest(){
      return "putMappingTest invoked";
  }

  @DeleteMapping
  public String deleteMappingTest(){
      return "deleteMappingTest invoked";
  }
}
```

38


GET http://localhost:8080/app/a

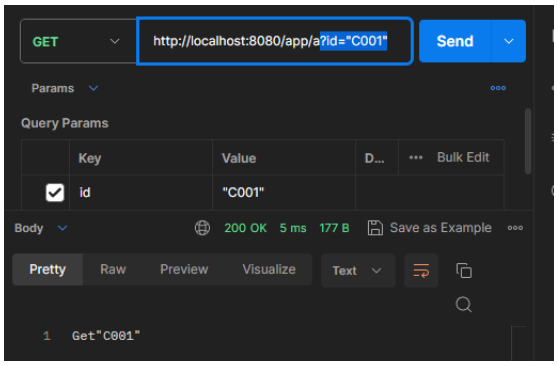
200 OK 9 ms 191 B

```
getMappingTest invoked
```

39

```
@GetMapping
public String getMapping(HttpServletRequest req, HttpServletResponse rsp) {
  String id = req.getParameter("id");
  System.out.println(id);
  return "Get" + id;
}
```

40


GET http://localhost:8080/app/a?id="C001"

Query Params

Key	Value	D...
id	"C001"	

200 OK 5 ms 177 B

```
Get"C001"
```

41

```
@GetMapping
public String getMapping1() {
  return "Get1";
}

@GetMapping
public String getMapping2() {
  return "Get2";
}
```

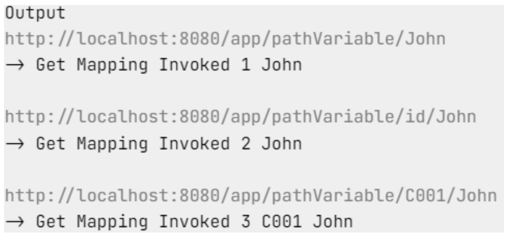
42

```
@RestController
@RequestMapping("/segment")
public class B_Path_Segments_Controller {
  //request narrow downing using path segments
  @GetMapping(path = "/two")
  public String getMapping2(){
     return "Get Mapping Invoked 2";
  }

  @GetMapping(path = "/three")
  public String getMapping3(){
     return "Get Mapping Invoked 3";
  }

  @GetMapping(path = "/three/four")
  public String getMapping4(){
     return "Get Mapping Invoked 3/4";
  }
}
```

43

```
@RestController
@RequestMapping("/pathVariable")
public class C_Path_Variables_Controller {
  @GetMapping(path = "/{name}")
  public String getMapping1(@PathVariable String name){
     return "Get Mapping Invoked 1 "+name;
  }

  @GetMapping(path = "/id/{name}")
  public String getMapping2(@PathVariable String name){
     return "Get Mapping Invoked 2 "+name;
  }

  @GetMapping(path = "/{id}/{name}")
  public String getMapping3(@PathVariable String id,@PathVariable String name){
     return "Get Mapping Invoked 3 "+id+" "+name;
  }
}
```

44

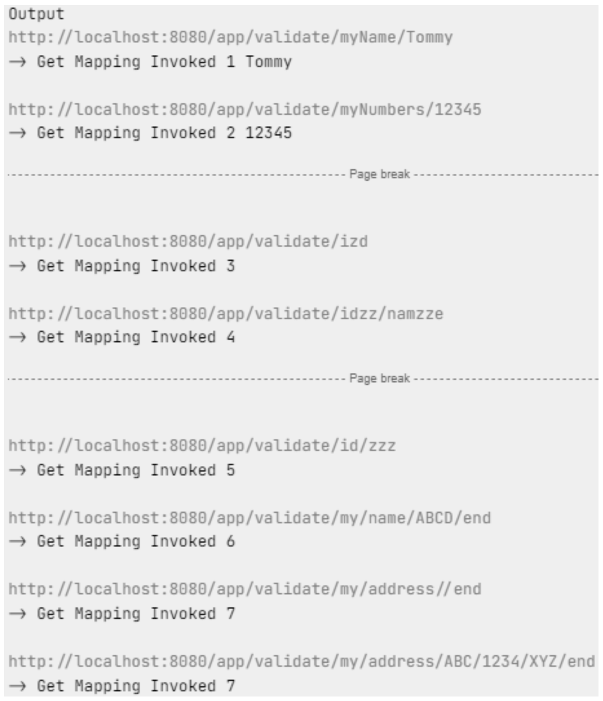


```
Output
http://localhost:8080/app/pathVariable/John
→ Get Mapping Invoked 1 John

http://localhost:8080/app/pathVariable/id/John
→ Get Mapping Invoked 2 John

http://localhost:8080/app/pathVariable/C001/John
→ Get Mapping Invoked 3 C001 John
```

45

```
@GetMapping(path = "/{id}")
public String getMapping(@PathVariable("id") String ids){ //alias
   return "Get Mapping Invoked "+ids;
}
```

47



```
Output
http://localhost:8080/app/validate/myName/Tommy
→ Get Mapping Invoked 1 Tommy

http://localhost:8080/app/validate/myNumbers/12345
→ Get Mapping Invoked 2 12345
------------------------------------------------ Page break ---------------------------


http://localhost:8080/app/validate/izd
→ Get Mapping Invoked 3

http://localhost:8080/app/validate/idzz/namzze
→ Get Mapping Invoked 4
------------------------------------------------ Page break ---------------------------


http://localhost:8080/app/validate/id/zzz
→ Get Mapping Invoked 5

http://localhost:8080/app/validate/my/name/ABCD/end
→ Get Mapping Invoked 6

http://localhost:8080/app/validate/my/address//end
→ Get Mapping Invoked 7

http://localhost:8080/app/validate/my/address/ABC/1234/XYZ/end
→ Get Mapping Invoked 7
```

48

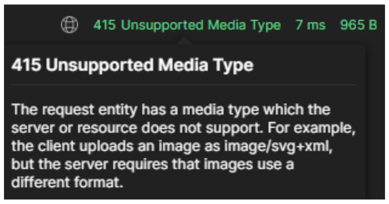
```
@RestController
@RequestMapping("/query")
public class E_Query_Sting_Parameters_Controller {
   @GetMapping(params = {"id", "name", "address"})
   public String getMapping1(@RequestParam String id, String name, String address) {
       return "Get Mapping Invoked 1 " + id + " " + name + " " + address;
   }
}
```

```
@RestController
@RequestMapping("/validate")
public class D_Validate_Path_Variables_Controller {
  //01) RegEx Validation path
  @GetMapping(path = "/myName/{name:[A-Z]{1}[a-z]{4}}")
  public String getMapping1(@PathVariable String name) {
    return "Get Mapping Invoked 1 " + name;
  }

  @GetMapping(path = "/myNumbers/{numbers:[0-9]{5}}")
  public String getMapping2(@PathVariable String numbers) {
    return "Get Mapping Invoked 2 " + numbers;
  }

  //02) Single Character Validations
  // ? තියෙන තැනට කැමති character එකක් දාන්න පුලුවන්, දන්නෙ නැති character
    path වලට use කරන්න පුලුවන්
  @GetMapping(path = "/i?d")
  public String getMapping3() {
    return "Get Mapping Invoked 3";
  }

  @GetMapping(path = "/id??/nam??e")
  public String getMapping4() {
    return "Get Mapping Invoked 4";
  }

  @GetMapping(path = "/id/???")
  public String getMapping5() {
    return "Get Mapping Invoked 5";
  }

  //03) Single Wild Card Validator (wildcard mapping)
  //* <- one or more characters inside a segment
  @GetMapping(path = "/my/name/*/end")
  public String getMapping6() {
    return "Get Mapping Invoked 6";
  }

  //04) Multiple Wild Card Validator (Dual wildcard mapping)
  //** <- zero or more segments with unlimited characters
  @GetMapping(path = "/my/address/**/end")
  public String getMapping7() {
    return "Get Mapping Invoked 7";
  }
}
```

49

```
@RestController
@RequestMapping("/headers")
public class F_Request_Headers_Controller {
  @GetMapping(consumes = "application/json")
  public String getMapping1() {
     return "Get Mapping Invoked 1";
  }

  @GetMapping(consumes = "text/html")
  public String getMapping2() {
     return "Get Mapping Invoked 2";
  }
}
```

50



415 Unsupported Media Type 7 ms 965 B

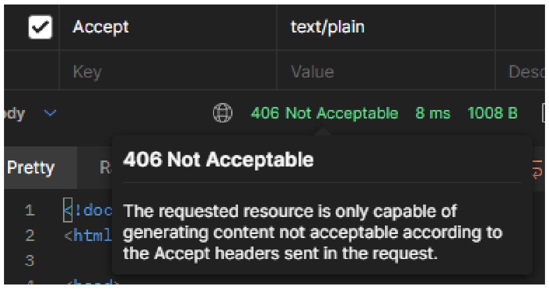
415 Unsupported Media Type

The request entity has a media type which the server or resource does not support. For example, the client uploads an image as image/svg+xml, but the server requires that images use a different format.

51

```
@RestController
@RequestMapping("/headers")
public class F_Request_Headers_Controller {
  @GetMapping(produces = "text/html")
  public String getMapping3() {
     return "Get Mapping Invoked 3";
  }

  @GetMapping(produces = "application/json")
  public String getMapping4() {
     return "Get Mapping Invoked 4";
  }
}
```

52



✓	Accept	text/plain
	Key	Value

406 Not Acceptable 8 ms 1008 B

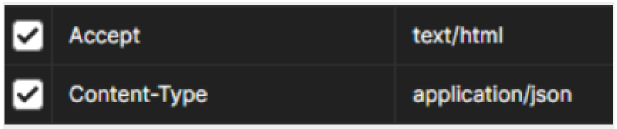
406 Not Acceptable

The requested resource is only capable of generating content not acceptable according to the Accept headers sent in the request.

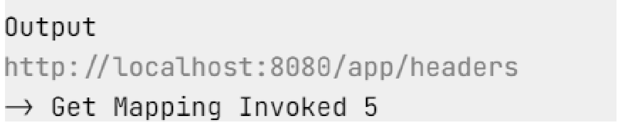
53

```
@GetMapping(headers = {"Content-Type=application/json","Accept=text/html"})
public String getMapping5() {
  return "Get Mapping Invoked 5";
}
```

54



✓	Accept	text/html
✓	Content-Type	application/json

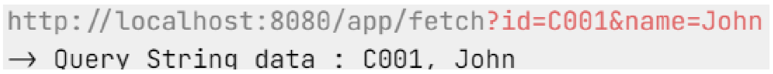


```
Output
http://localhost:8080/app/headers
→ Get Mapping Invoked 5
```

55

```
@RestController
@RequestMapping("/fetch")
public class A_Data_Fetch_Controller {
  @GetMapping(params = {"id","name"})
  public String receiveDataWithQueryString(String id,@RequestParam String name){
     return "Query String data : "+id+", "+name;
  }
}
```

56

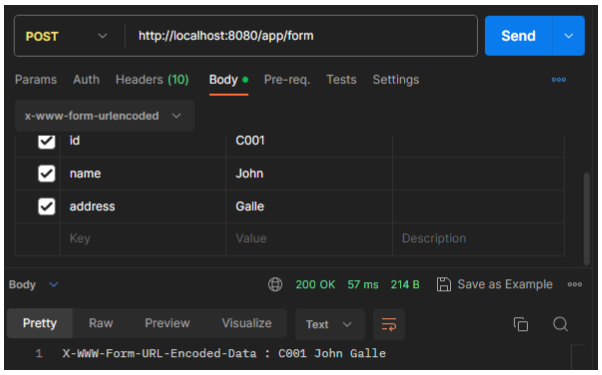


```
http://localhost:8080/app/fetch?id=C001&name=John
→ Query String data : C001, John
```

57

```
@RestController
@RequestMapping("/form")
public class B_X_WWW_Url_Encoded_Controller {
  @PostMapping
  public String receiveDataWithFormData(String id, String name, String address) {
     return "X-WWW-Form-URL-Encoded-Data : " + id + " " + name + " " + address;
  }
}
```

58



POST http://localhost:8080/app/form

Body: x-www-form-urlencoded

	Key	Value	Description
✓	id	C001	
✓	name	John	
✓	address	Galle	

200 OK 57 ms 214 B

```
X-WWW-Form-URL-Encoded-Data : C001 John Galle
```

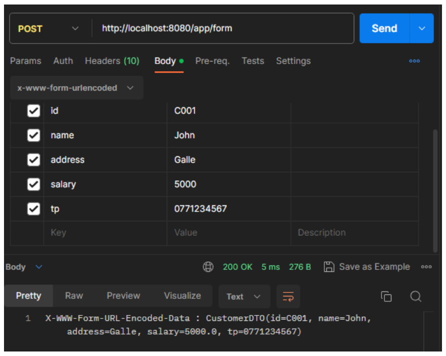
59

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CustomerDTO {
   private String id;
   private String name;
   private String address;
   private double salary;
   private String tp;
}
```

60

```
@RestController
@RequestMapping("/form")
public class B_X_WWW_Url_Encoded_Controller {
   @PostMapping
   public String receiveData(@ModelAttribute CustomerDTO dto) {
       return "X-WWW-Form-URL-Encoded-Data : " +dto;
   }
}
```

61



POST http://localhost:8080/app/form

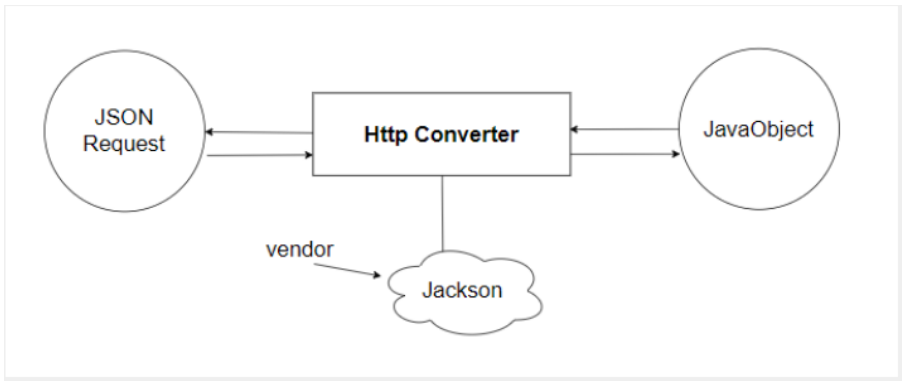
x-www-form-urlencoded

	Key	Value	Description
✓	id	C001	
✓	name	John	
✓	address	Galle	
✓	salary	5000	
✓	tp	0771234567	

200 OK 5 ms 276 B

```
X-WWW-Form-URL-Encoded-Data : CustomerDTO(id=C001, name=John,
    address=Galle, salary=5000.0, tp=0771234567)
```

62



Http Converter - request/response body data, different MIME types වලට convert කරයි.

(MIME (Multipurpose Internet Mail Extensions) is a standard for identifying the type of data in files. Example: text/html for HTML web pages, application/json for JSON data, and image/jpeg for JPEG images. It helps browsers and servers handle data correctly.)

63

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class CustomerDTO {
  private String id;
  private String name;
  private String address;
  private double salary;
  private String tp;
  private ArrayList<ItemDTO> items;
}
```

64

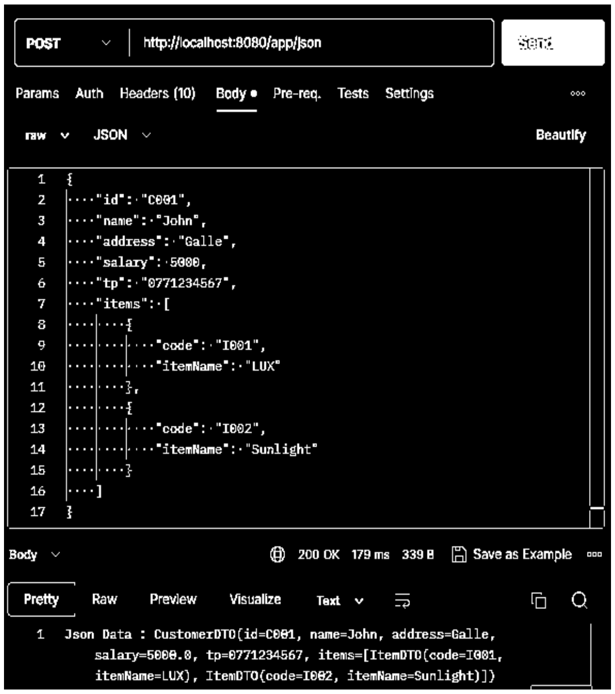
```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class ItemDTO {
  private String code;
  private String itemName;
}
```

65

POST

```
@RestController
@RequestMapping("/json")
public class C_JSON_Controller {
  @PostMapping
  public String receiveDataWithJson(@RequestBody CustomerDTO dto){
     return "Json Data : "+dto.toString();
  }
}
```

66



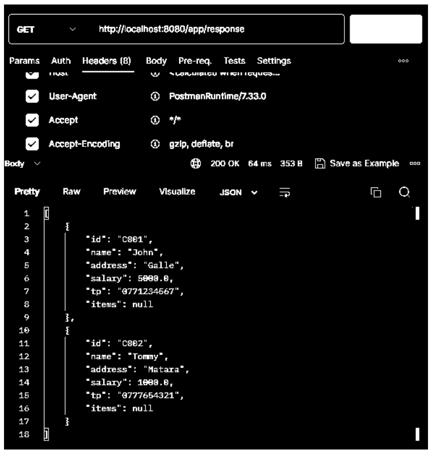
```
{
    "id": "C001",
    "name": "John",
    "address": "Galle",
    "salary": 5000,
    "tp": "0771234567",
    "items": [
        {
            "code": "I001",
            "itemName": "LUX"
        },
        {
            "code": "I002",
            "itemName": "Sunlight"
        }
    ]
}
```

```
Json Data : CustomerDTO(id=C001, name=John, address=Galle,
    salary=5000.0, tp=0771234567, items=[ItemDTO(code=I001,
    itemName=LUX), ItemDTO(code=I002, itemName=Sunlight)])
```

67

```
@RestController
@RequestMapping("/response")
public class D_Response_Controller {
  @GetMapping
  public ArrayList<CustomerDTO> sendJsonData(){
      ArrayList<CustomerDTO> allCustomers = new ArrayList<>();
      allCustomers.add(new CustomerDTO("C001","John","Galle",5000,"0771234567",null));
      allCustomers.add(new CustomerDTO("C002","Tommy","Matara",1000,"0777654321",null));
      return allCustomers;
  }
}
```

68



GET http://localhost:8080/app/response

Params Auth Headers (8) Body Pre-req. Tests Settings

```
User-Agent        PostmanRuntime/7.33.0
Accept            */*
Accept-Encoding   gzip, deflate, br
```

Body 200 OK 64 ms 353 B Save as Example

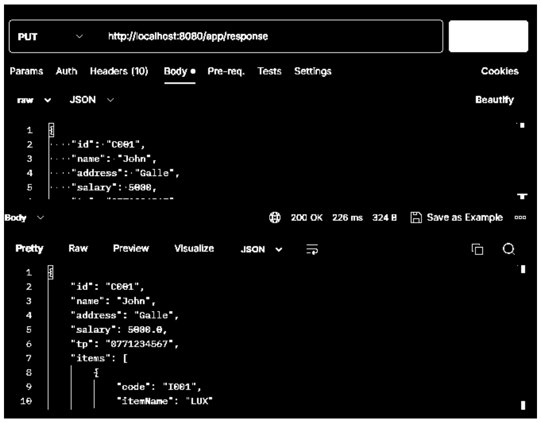
Pretty Raw Preview Visualize JSON

```
[
    {
        "id": "C001",
        "name": "John",
        "address": "Galle",
        "salary": 5000.0,
        "tp": "0771234567",
        "items": null
    },
    {
        "id": "C002",
        "name": "Tommy",
        "address": "Matara",
        "salary": 1000.0,
        "tp": "0777654321",
        "items": null
    }
]
```

69

```
@RestController
@RequestMapping("/response")
public class D_Response_Controller {
  @PutMapping
  public CustomerDTO receiveDataWithJson(@RequestBody CustomerDTO dto) {
      return dto;
  }
}
```

70



PUT http://localhost:8080/app/response

Params Auth Headers (10) Body Pre-req. Tests Settings Cookies

raw JSON Beautify

```
{
    "id": "C001",
    "name": "John",
    "address": "Galle",
    "salary": 5000,
```

Body 200 OK 226 ms 324 B Save as Example

Pretty Raw Preview Visualize JSON

```
{
    "id": "C001",
    "name": "John",
    "address": "Galle",
    "salary": 5000.0,
    "tp": "0771234567",
    "items": [
        {
            "code": "I001",
            "itemName": "LUX"
```

71

```
@ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
@RestControllerAdvice
public class AppWideExceptionHandler {
   @ExceptionHandler({RuntimeException.class})
   public ResponseUtil handleAllRuntimeException(RuntimeException e){
       return new ResponseUtil("Error", e.getMessage(), null);
   }
}
```

72

```
@ExceptionHandler({Exception.class})
public ResponseUtil handleAllExceptions(Exception e){
   return new ResponseUtil("Error", e.getMessage(), null);
}
```

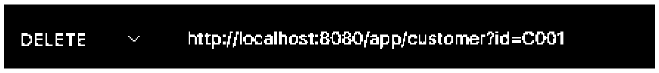
73

```
@EnableWebMvc
@Configuration
@ComponentScan(basePackages = {"lk.ijse.spring.controller","lk.ijse.spring.adviser"})
public class WebAppConfig {
}
```

74

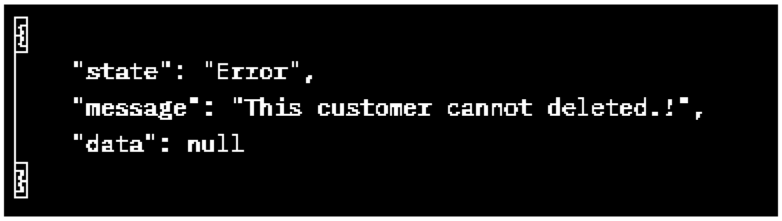
```
@RestController
@RequestMapping("/customer")
public class CustomerController {
   @DeleteMapping(params = "id")
   public ResponseUtil deleteCustomer(@RequestParam String id) {
       if (id.endsWith("C001"))
           throw new RuntimeException("This customer cannot deleted.!");
       return new ResponseUtil("Ok", "Successfully Delete", id);
   }
}
```

75



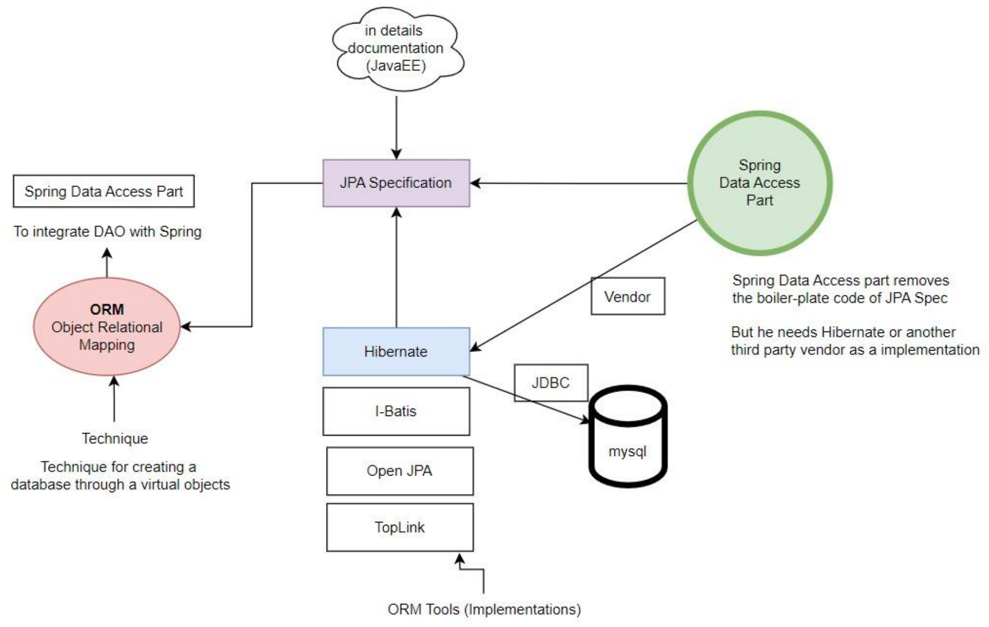
```
DELETE    http://localhost:8080/app/customer?id=C001
```

76



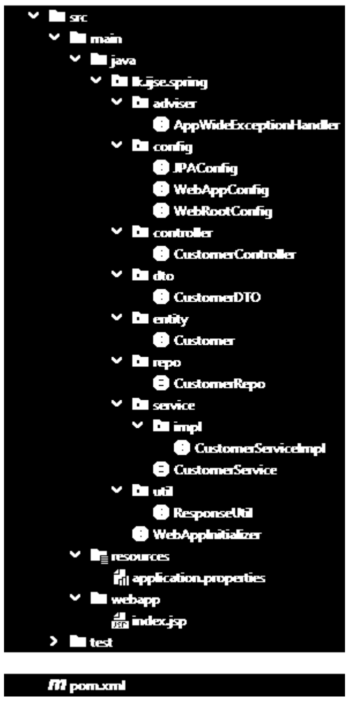
```
{
    "state": "Error",
    "message": "This customer cannot deleted.!",
    "data": null
}
```

77



in details documentation (JavaEE)
JPA Specification
Spring Data Access Part
Spring Data Access Part
To integrate DAO with Spring
ORM
Object Relational Mapping
Technique
Technique for creating a database through a virtual objects
Hibernate
Vendor
JDBC
mysql
I-Batis
Open JPA
TopLink
ORM Tools (Implementations)
Spring Data Access part removes the boiler-plate code of JPA Spec
But he needs Hibernate or another third party vendor as a implementation

78


```
src
  main
    java
      lk.ijse.spring
        adviser
          AppWideExceptionHandler
        config
          JPAConfig
          WebAppConfig
          WebRootConfig
        controller
          CustomerController
        dto
          CustomerDTO
        entity
          Customer
        repo
          CustomerRepo
        service
          impl
            CustomerServiceImpl
          CustomerService
        util
          ResponseUtil
        WebAppInitializer
    resources
      application.properties
    webapp
      index.jsp
  test

pom.xml
```

79

```
@Configuration
@EnableJpaRepositories(basePackages = "lk.ijse.spring.repo")
@EnableTransactionManagement
public class JPAConfig {
  @Bean
  public LocalContainerEntityManagerFactoryBean entityManagerFactory(DataSource ds, JpaVendorAdapter vad) {
     LocalContainerEntityManagerFactoryBean factory = new LocalContainerEntityManagerFactoryBean();
     factory.setDataSource(ds);
     factory.setJpaVendorAdapter(vad);
     factory.setPackagesToScan("lk.ijse.spring.entity"); // set entity locations
     return factory;
  }

  @Bean
  public DataSource dataSource() {
     DriverManagerDataSource ds = new DriverManagerDataSource();
     ds.setUsername("root");
     ds.setPassword("1234");
     ds.setDriverClassName("com.mysql.jdbc.Driver");    ds.setUrl("jdbc:mysql://localhost:3306/customerDB?createDatabaseIfNotEx
ist=true");
     return ds;
  }

  @Bean
  public JpaVendorAdapter jpaVendorAdapter() {
     HibernateJpaVendorAdapter va = new HibernateJpaVendorAdapter();
     // Set the database platform to MySQL 8 Dialect
     va.setDatabasePlatform("org.hibernate.dialect.MySQL8Dialect");
     va.setDatabase(Database.MYSQL); // Set the database type to MySQL
     va.setGenerateDdl(true); // Generate Data Definition Language (DDL) queries
     va.setShowSql(true); // Show SQL queries in the logs
     return va;
  }

  @Bean
  public PlatformTransactionManager transactionManager(EntityManagerFactory factory) {
     return new JpaTransactionManager(factory);
  }
}
```

80

```
@Configuration
@Import(JPAConfig.class) // Import the JPA configuration class
public class WebRootConfig {
}
```

81

```java
@CrossOrigin // Enable Cross-Origin Resource Sharing (CORS) for this controller
@RestController
@RequestMapping("/customer")
public class CustomerController {
  @Autowired
  private CustomerService service;

  @ResponseStatus(HttpStatus.CREATED)
  // get all
  @GetMapping
  public ResponseUtil getAllCustomers() {
     return new ResponseUtil("Ok", "Successfully Loaded", service.getAllCustomers());
  }

  // find
  @GetMapping(params = {"id"})
  public ResponseUtil findCustomer(String id) {
     return new ResponseUtil("Ok", "Successfully Searched", service.searchCustomer(id));
  }

  // add
  @PostMapping
  public ResponseUtil saveCustomer(@ModelAttribute CustomerDTO dto) {
     service.saveCustomer(dto);
     return new ResponseUtil("Ok", "Successfully Added", dto);
  }

  // update
  @PutMapping
  public ResponseUtil updateCustomer(@RequestBody CustomerDTO dto) {
     service.updateCustomer(dto);
     return new ResponseUtil("Ok", "Successfully Updated", dto);
  }

  // delete
  @DeleteMapping(params = {"id"})
  public ResponseUtil deleteCustomer(String id) {
     service.deleteCustomer(id);
     return new ResponseUtil("Ok", "Successfully Deleted", id);
  }
}
```

82

```java
public interface CustomerService {
  ArrayList<CustomerDTO> getAllCustomers();
  CustomerDTO searchCustomer(String id);
  void saveCustomer(CustomerDTO dto);
  void updateCustomer(CustomerDTO dto);
  void deleteCustomer(String id);
}
```

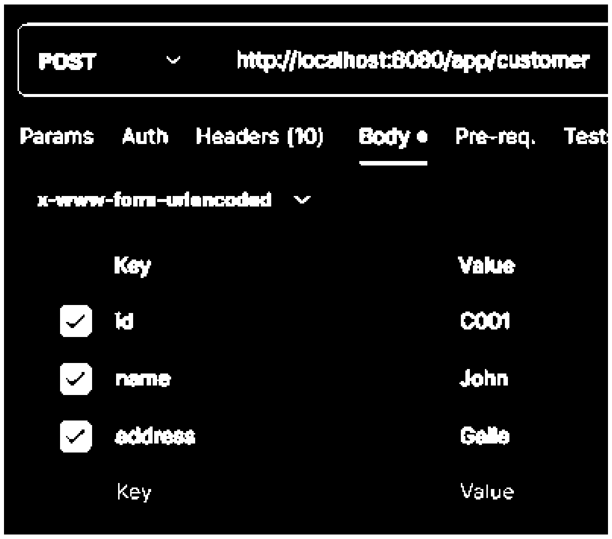
83

```
@Service
@Transactional
public class CustomerServiceImpl implements CustomerService {
  @Autowired
  CustomerRepo repo;

  @Autowired
  ModelMapper mapper;

  @Override
  public ArrayList<CustomerDTO> getAllCustomers() {
    List<Customer> all = repo.findAll();
    return mapper.map(all, new TypeToken<ArrayList<CustomerDTO>>() {}.getType());
  }

  @Override
  public CustomerDTO searchCustomer(String id) {
    if (!repo.existsById(id)) throw new RuntimeException("Id not exists !");
    return mapper.map(repo.findById(id).get(), CustomerDTO.class);
  }

  @Override
  public void saveCustomer(CustomerDTO dto) {
    if (repo.existsById(dto.getId())) throw new RuntimeException("Error, Already added!");
    repo.save(mapper.map(dto, Customer.class));
  }

  @Override
  public void updateCustomer(CustomerDTO dto) {
    if (!repo.existsById(dto.getId())) throw new RuntimeException("Id not exists !");
    repo.save(mapper.map(dto, Customer.class));
  }

  @Override
  public void deleteCustomer(String id) {
    if (!repo.existsById(id)) throw new RuntimeException("Id not exists !");
    repo.deleteById(id);
  }
}
```

84

```
public interface CustomerRepo extends JpaRepository<Customer, String> {
}
```

85

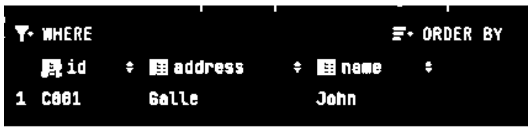

POST http://localhost:8080/app/customer

Params Auth Headers (10) Body ● Pre-req. Test

x-www-form-urlencoded

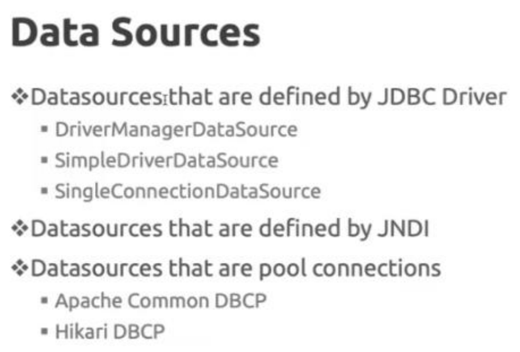
	Key	Value
✓	id	C001
✓	name	John
✓	address	Galle
	Key	Value

86


WHERE ORDER BY

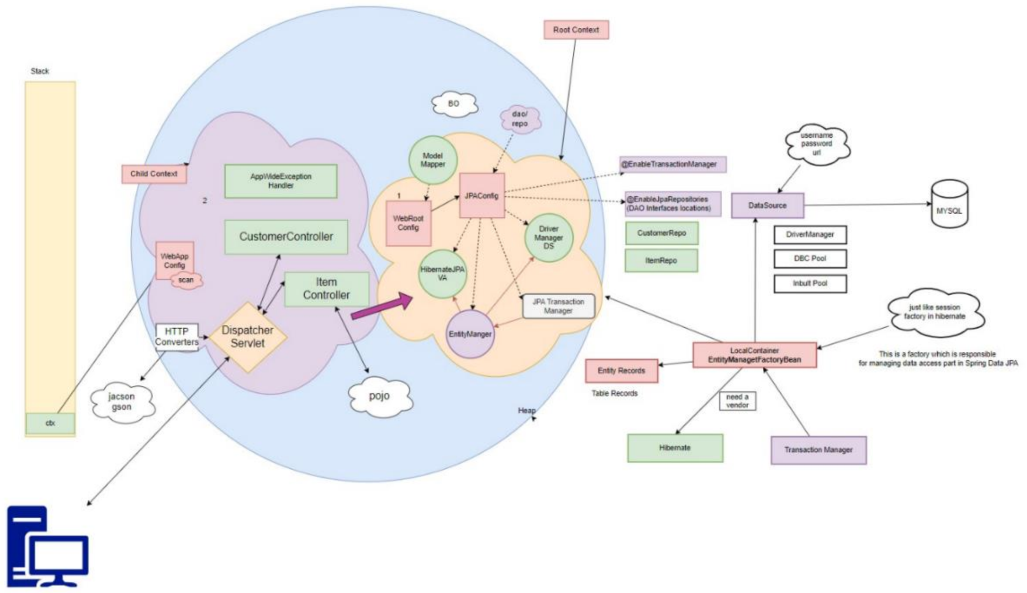
```
   id      address      name
1  C001    Galle        John
```

87

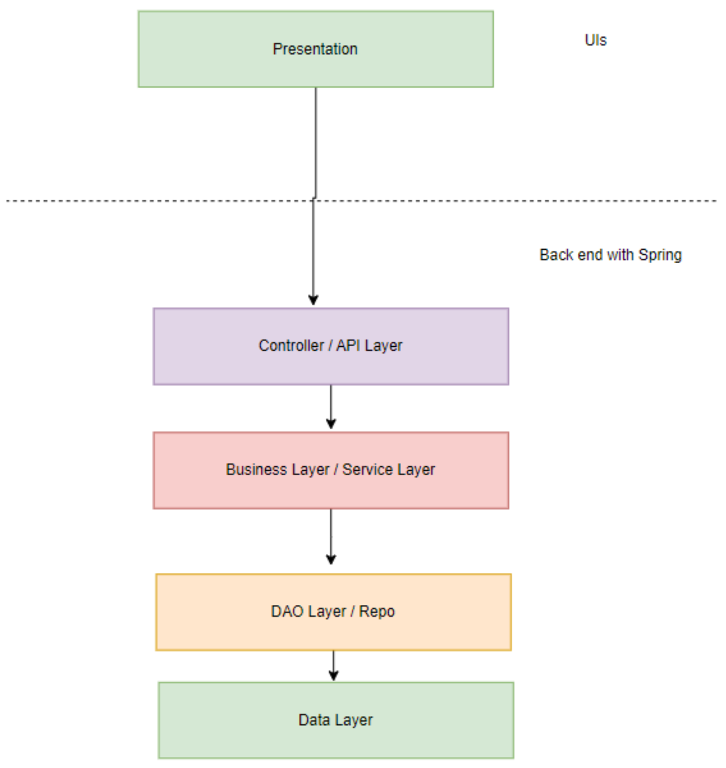

Data Sources

- Datasources that are defined by JDBC Driver
  - DriverManagerDataSource
  - SimpleDriverDataSource
  - SingleConnectionDataSource
- Datasources that are defined by JNDI
- Datasources that are pool connections
  - Apache Common DBCP
  - Hikari DBCP

88



89



90

```
Customer customer = new Customer(dto.getId(),dto.getName(),dto.getAddress());
```

91

```
Customer customer = mapper.map(dto, Customer.class);
```

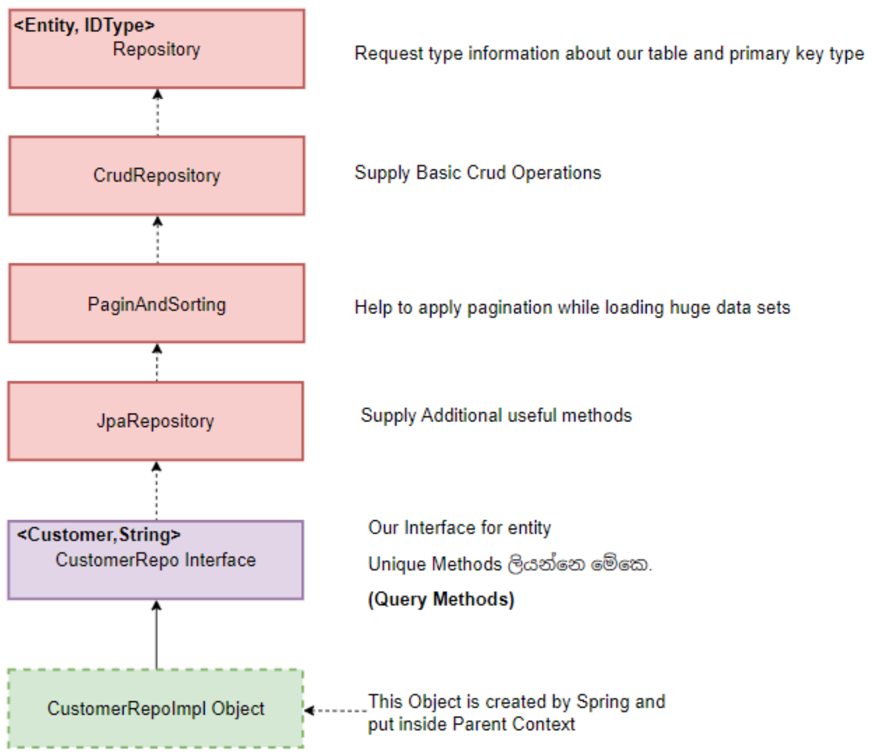
92

```
mapper.map(param1, param2.class);
```

93

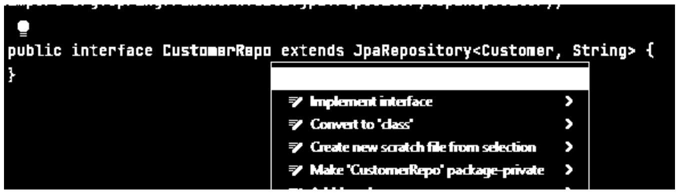
```
@Override
public List<CustomerDTO> getAllCustomers() {
   List<Customer> all = customerRepo.findAll();
   return mapper.map(all, new TypeToken<List<CustomerDTO>>() {
   }.getType());
   //new TypeToken<>(){}.getType()
   //new TypeToken<List<CustomerDTO>>(){}.getType()
}
```

94



<Entity, IDType>
Repository
Request type information about our table and primary key type
CrudRepository
Supply Basic Crud Operations
PaginAndSorting
Help to apply pagination while loading huge data sets
JpaRepository
Supply Additional useful methods
<Customer,String>
CustomerRepo Interface
Our Interface for entity
Unique Methods ලියන්නෙ මෙතෙ.
(Query Methods)
CustomerRepoImpl Object
This Object is created by Spring and put inside Parent Context

95



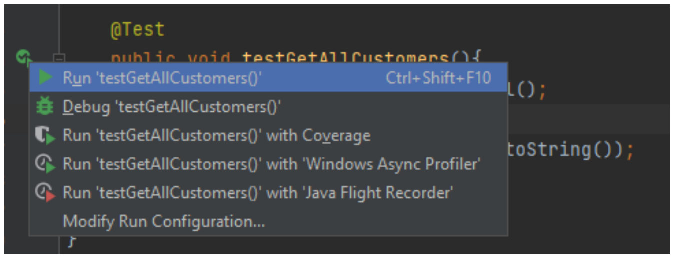
```
public interface CustomerRepo extends JpaRepository<Customer, String> {
}
```

Implement interface
Convert to 'class'
Create new scratch file from selection
Make 'CustomerRepo' package-private

96

```
@WebAppConfiguration
@ContextConfiguration(classes = {WebRootConfig.class})
@ExtendWith(SpringExtension.class)
@Transactional
class CustomerRepoTest {
  @Autowired
  CustomerRepo repo;

  @Test
  public void testGetAllCustomers(){
    List<Customer> all = repo.findAll();
    for (Customer customer: all){
      System.out.println(customer.toString());
    }
  }
}
```

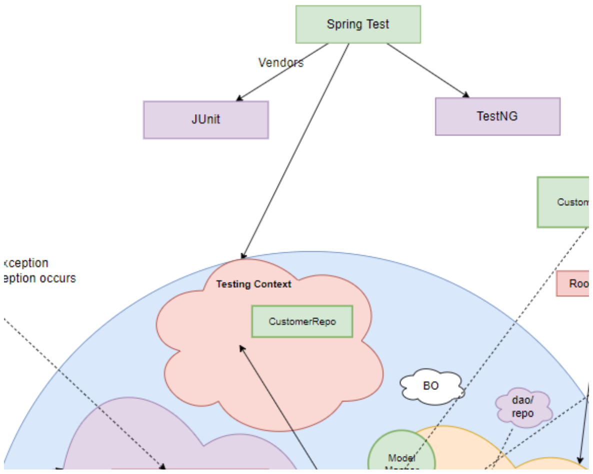
97



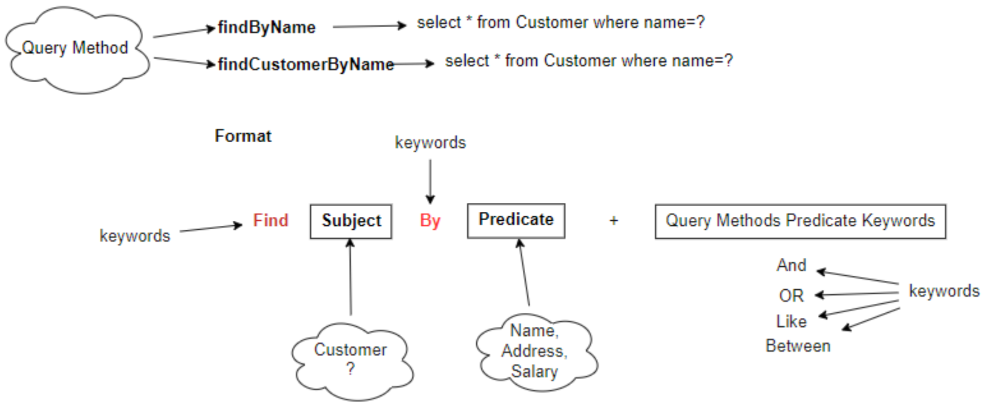
```
@Test
public void testGetAllCustomers(){
l();
toString());
}
```

Run 'testGetAllCustomers()' Ctrl+Shift+F10
Debug 'testGetAllCustomers()'
Run 'testGetAllCustomers()' with Coverage
Run 'testGetAllCustomers()' with 'Windows Async Profiler'
Run 'testGetAllCustomers()' with 'Java Flight Recorder'
Modify Run Configuration...

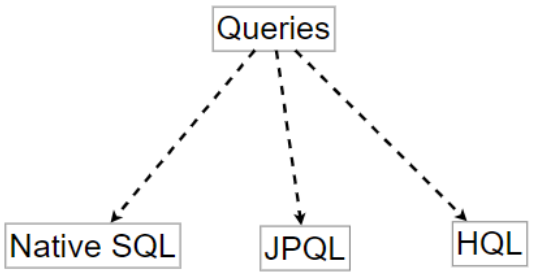
98



99



100



1. Native query

```
@Query(value = "select * from Customer",nativeQuery = true)
List<Customer> getAllCustomers1();
```

2. JPQL

```
@Query(value = "select c from Customer c")
List<Customer> getAllCustomers2();
```

3. HQL

```
@Query(value = "from Customer c")
List<Customer> getAllCustomers3();
```

named parameters

```
@Query(value = "select * from Customer where name=:nm",nativeQuery = true)
List<Customer> searchCustomerWithName(@Param("nm") String name);
```

Positional parameters

```
@Query(value = "select * from Customer where name=?1 and address=?2 ",nativeQuery = true)
List<Customer> searchCustomerWithName(String name, String address);
```