```
=> NOTE :- Before learning springboot, we should have the knowledge of spring framework

=> Core concepts we should know :-
    -> Why we need spring framework ?
        = Dependecy Injection [lossly coupled]
        = Lightweight, simple and easy
        = Can be integrated with other frameworks
          etc
    -> What is bean ?
        = Bean is simple java object
        = Bean is the backbone in spring application which are managed by Spring IOC container
    -> How many ways we can create beans ?
        = We can create beans by 3 ways :-
            1. by using spring xml configuration
            2. by using @Component annotation [used at class level]
            3. by using @Bean (and @Configuration) annotation [used at method level]

==========> Springboot is used to boot our spring application ==========
```

=> Springboot :-

-> Springboot is an open source java-based framework which is developed by Pivotal Team

-> It is used to create a stand-alone and production-grade spring-based application that we can "just run" i.e. it provides an easier way to start springboot application

-> Springboot framework is developed on the top of core spring framework

-> Advantages of Springboot :-

1. It follows "Opinionated Defaults Configuration" approach to reduce developer efforts
2. It aviods boilerplate code, annotations and xml configurations which in turn reduces the developer time and increases the productivity
3. It can be easily integrated with Spring Modules i.e. Spring JDBC, Spring AOP, Spring Security etc
4. It provides embdded HTTP servers i.e. tomcat, jetty etc
5. It provides CLI (Command Line Interface) tool to develop and test springboot application from command prompt in an easy and quick way
6. It provides a lot of plugins to develop and test springboot applications
7. It provides a lot of plugins to work with embedded and in-memory databases etc

-> Key components of Springboot Framework :-

1. Springboot Starters
2. Springboot AutoConfigurator
3. Springboot CLI
4. Springboot Actuators

---

1. Springboot Starters :-
   -> Springboot starters are the "dependency descriptors"
   -> Springboot provides a number of starters that allow us to add jar files in the classpath
   -> In springboot, all the starters follows a similar name pattern i.e. "spring-boot-starter-*" for example spring-boot-starter-web, spring-boot-starter-jdbc etc
   -> Springboot starters are divided in 3 categories :-
      a. Application Starters
         = spring-boot-starter
         = spring-boot-starter-web
         = spring-boot-starter-jdbc
         = spring-boot-starter-aop
         = spring-boot-starter-test
           etc
      b. Technical Starters
         = spring-boot-starter-tomcat
         = spring-boot-starter-jetty
         = spring-boot-starter-logging
         = spring-boot-starter-log4j
           etc
      c. Production Starters
         = spring-boot-starter-actuators

```
    c. Production Starters
        = spring-boot-starter-actuators
-> NOTE : "spring-boot-starter" name is reserved for official spring boot artifacts

-> Syntax of springboot starters :-
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
            <version>2.7.0</version>
        </dependency>

-> There are many third party starters which we can include in our applications
-> Third party starters follows the below name patter :-
    = projectname-spring-boot-starter
```

2. Springboot AutoConfigurator :-

-> Springboot AutoConfigurators, automatically configures the spring application configurations based on the jar dependencies that we have added

-> All auto-configuration logic is implemented in "spring-boot-autoconfigure.jar"

-> pringboot AutoConfigurator provides one annotation i.e. @SpringBootApplication

= @SpringBootApplication = @Configuration + @ComponentScan + @EnableAutoConfiguration

```
<build>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.10.1</version>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
    </configuration>
  </plugin>
</plugins>
</build>
```

```xml
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.springframew
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <version>2.7.3</version>
    </dependency>
</dependencies>
```


spring.main.web-application-type=none

```
=> Different ways to create Springboot Applications :-
    1. using Maven Project in Eclipse
    2. installing STS Tool in Eclipse
    3. using STS (Spring Tool Suit) IDE
    4. using Springboot Initializr
    5. using Springboot CLI (Command Line Interface)

================================================================================

=> Springboot Annotations :-
    -> Springboot annotations are present in following packages :-
        = org.springframework.boot.autoconfigure
        = org.springframework.boot.autoconfigure.condition
    -> Some common annotations used are :-
        = @SpringBootApplication
        = @AutoConfiguration
        = @EnableAutoConfiguration
        = @ImportAutoConfiguration
        = Conditional Annotations :-
            - @Conditional
            - @ConditionalOnBean and @ConditionalOnMissingBean
            - @ConditionalOnClass and @ConditionalOnMissingClass
              etc
```

- Create an appropriate `ApplicationContext` instance (depending on your classpath)
- Register a `CommandLinePropertySource` to expose command line arguments as Spring properties
- Refresh the application context, loading all singleton beans
- Trigger any `CommandLineRunner` beans

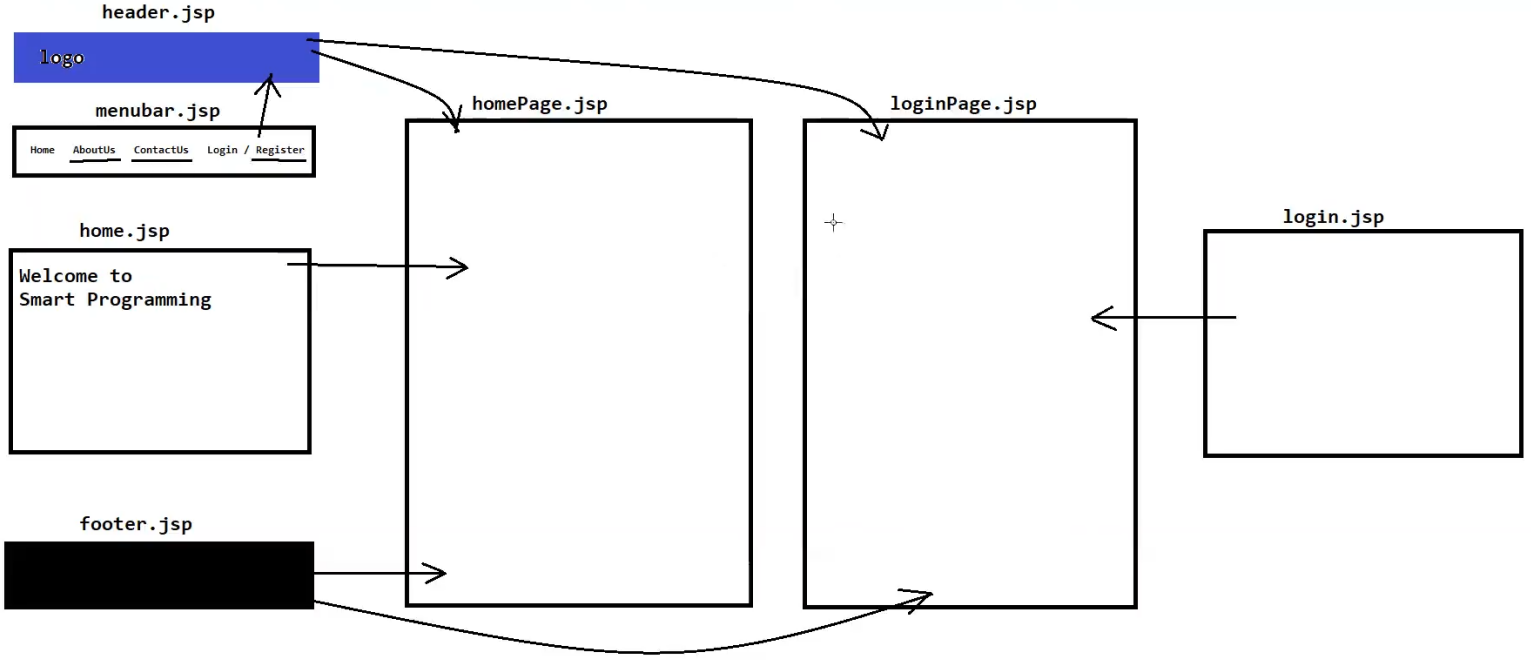
=> Tasks performed by run method :-
-> Calculate the duration of project started
-> Creates ApplicationContext object
-> Start the listeners
-> Prepares the environment i.e. production or dev or test environment
-> Print the banner
-> Trigger the Runners

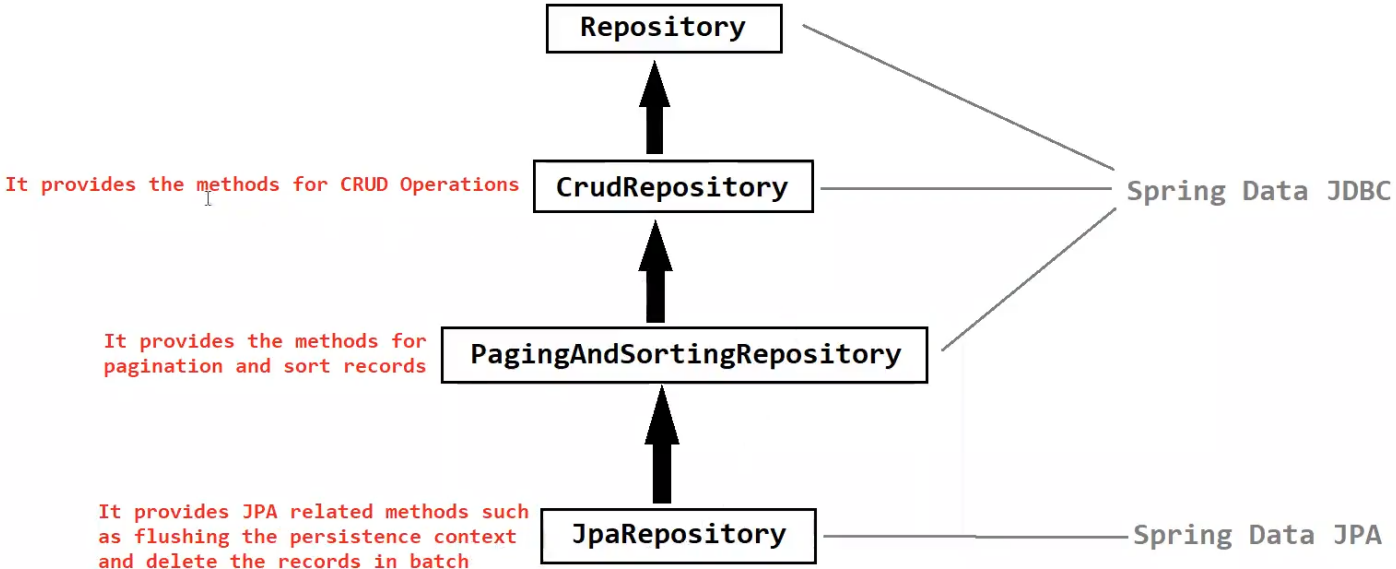
```
<dependency>
    <groupId>org.apache.tomcat</groupId>
    <artifactId>tomcat-jasper</artifactId>
    <version>10.0.27</version>
</dependency>
```

```
3 server.port=9090
```

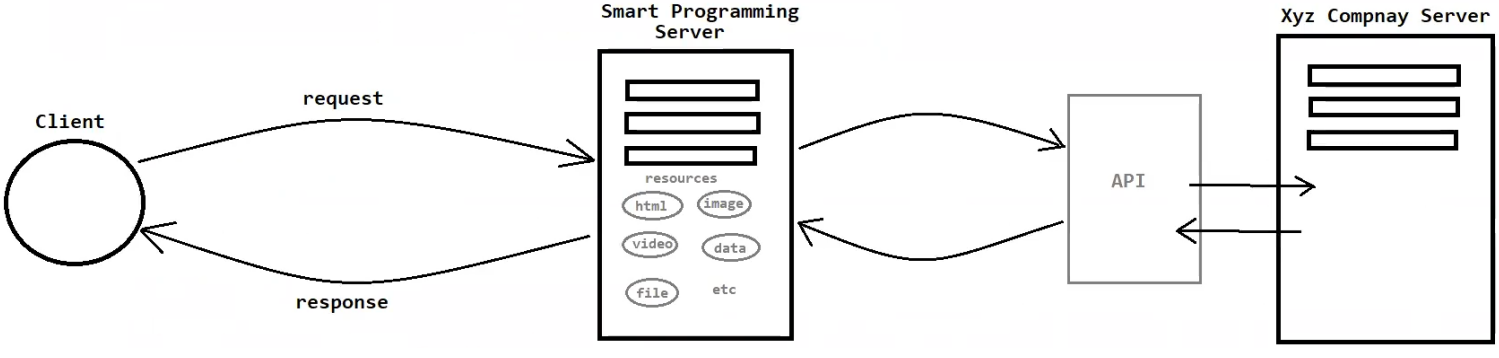
=> What is groovy :-
    -> Groovy is a programming language whose syntax is similar to java
    -> It follows the "Object Oriented Programming" paradiagm
    -> It is developed by Apache Foundation
    -> It can be used as a scripting langauge for java platform
    -> It is both static and dynamic language with features similar to python, ruby, smalltalk etc

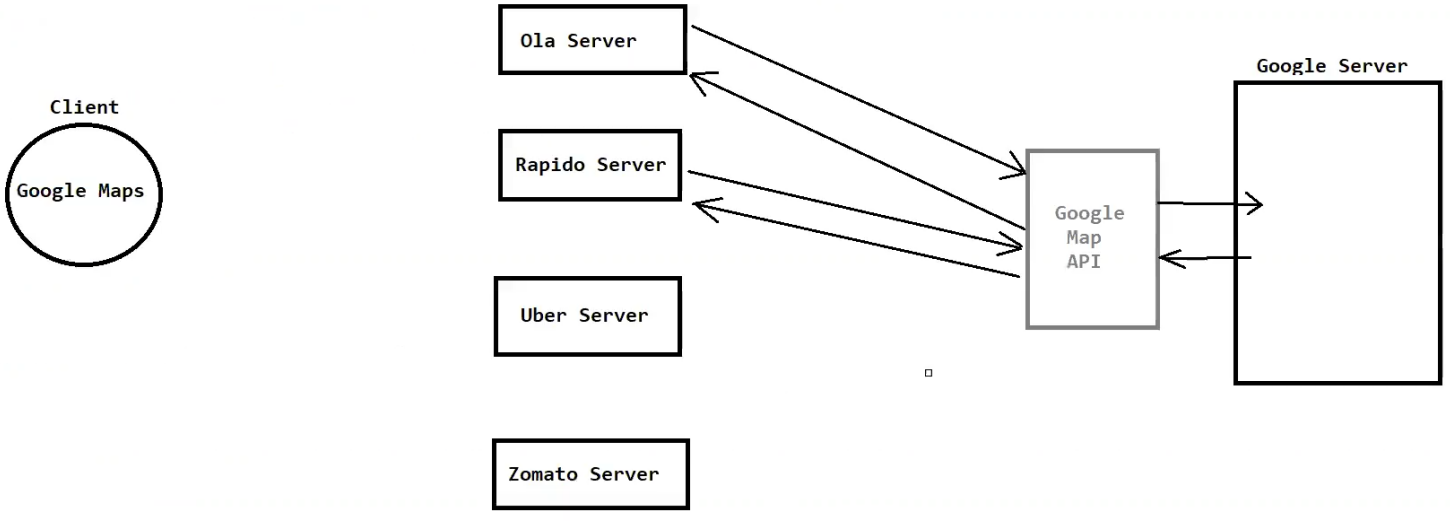
=> Run using Springboot CLI (Command Line Interface) [used for groovy]

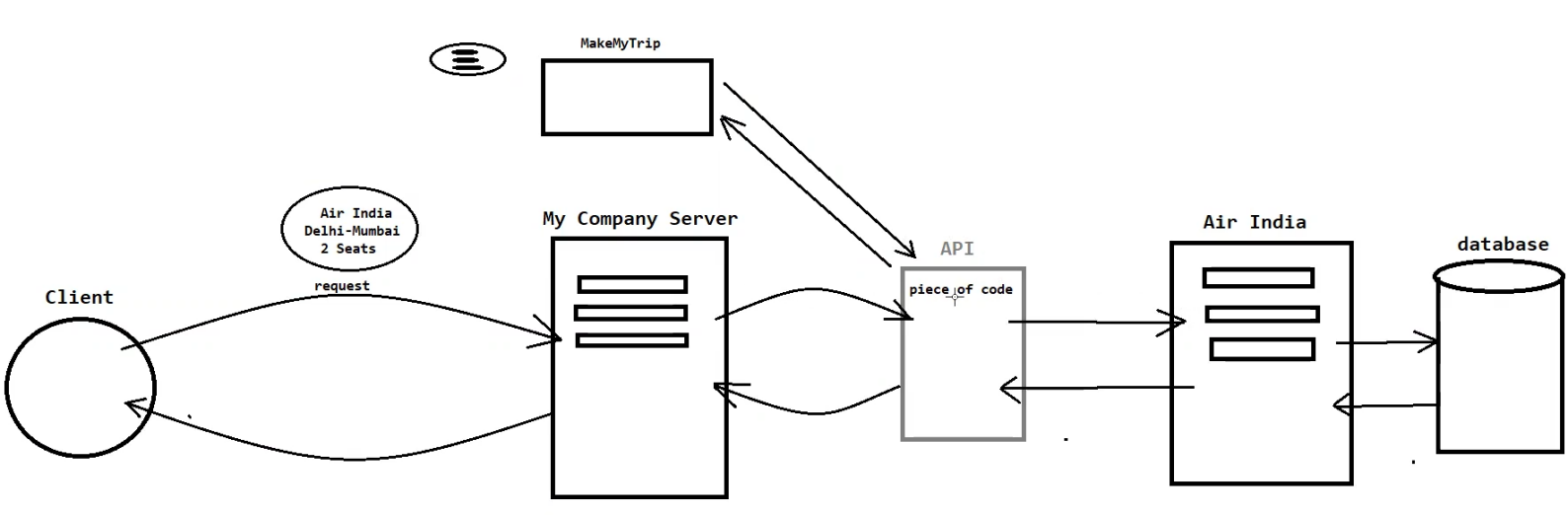

header.jsp
logo
menubar.jsp
Home
AboutUs
ContactUs
Login / Register
homePage.jsp
loginPage.jsp
login.jsp
home.jsp
Welcome to
Smart Programming
footer.jsp


Repository
It provides the methods for CRUD Operations
CrudRepository
Spring Data JDBC
It provides the methods for pagination and sort records
PagingAndSortingRepository
It provides JPA related methods such as flushing the persistence context and delete the records in batch
JpaRepository
Spring Data JPA

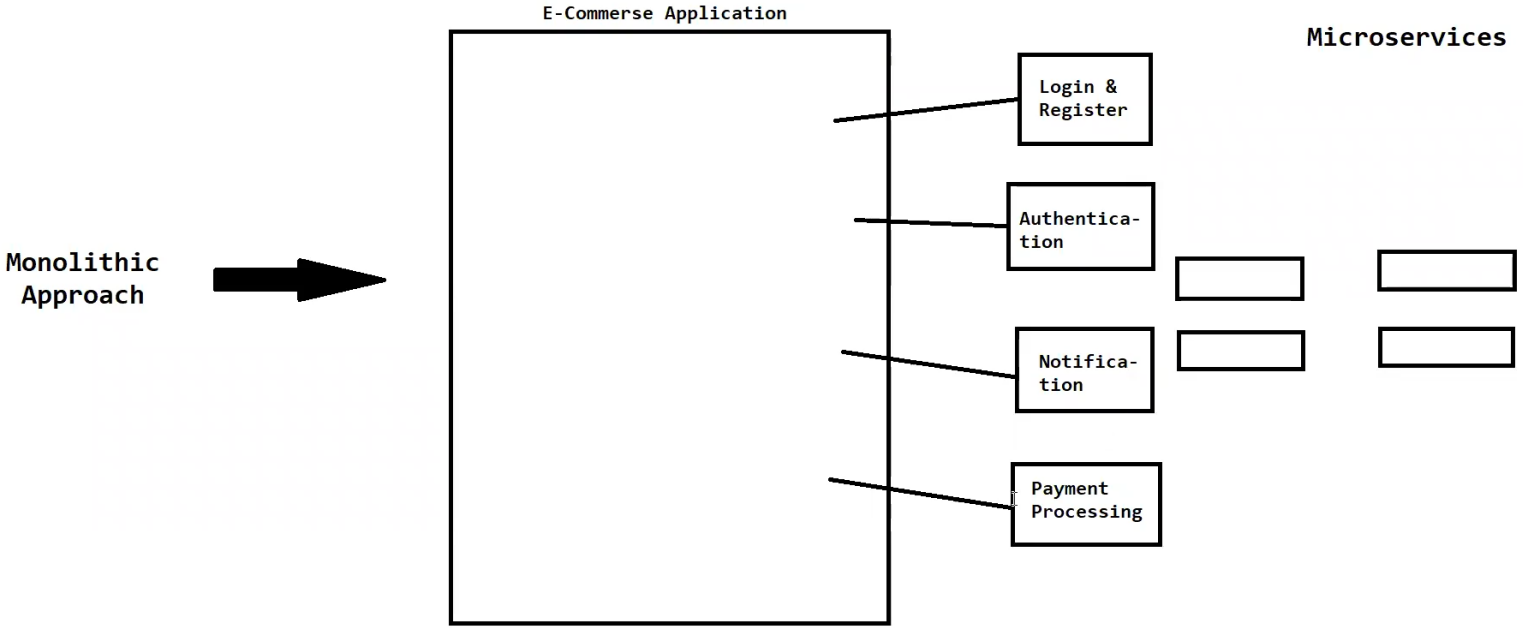
=> API :-
-> Full form of API is Application Programming Interface
-> API is an application component (or piece of code) which is used for communication between different systems over the network
-> API acts as bridge between different systems or devices
-> For example :-
= Login API's (google and facebook login API)
= Google Map API's
= Payment Gateway API's
= Airline, Bus, Train Booking API's
= Movie Booking API's
= Weather API's
= Youtube, Amazon, Flipkart API's etc
etc
-> Types of API's :-
1. Public or Open API's
= These are open to the public which can be used by anyone.
= These API's can be paid or free of cost
2. Partner API's
= These API's are meant for business-to-business partnership
3. Private or Internal API's
= These are internal to any specific enterprice application within the company
4. Composit API's
= These combine any two or more different API's for any system or project


Smart Programming
Server
Xyz Compnay Server
Client
request
response
resources
html
image
video
data
file
etc
API


Client
Google Maps
Ola Server
Rapido Server
Uber Server
Zomato Server
Google
Map
API
Google Server


MakeMyTrip
Air India
Delhi-Mumbai
2 Seats
request
Client
My Company Server
API
piece of code
Air India
database

=> Web Services :-
    -> Web Services are a type of API used to exchange the data (XML or JSON) between different systems over the network
    -> Web services are independent of the platform on which they are developed. For example if we have created web services in java, then it can communicate with any other language also like python or php etc
    -> Web services use HTTP protocol
    -> NOTE :-
        = All web services are API's but not all API's are web services
        = To test web services we will use "Postman" tool
    -> There are 2 types of web services :-
        = SOAP web services
        = RESTful web services


E-Commerse Application
Microservices
Monolithic Approach
Login & Register
Authentica-tion
Notifica-tion
Payment Processing

=> Microservices :-

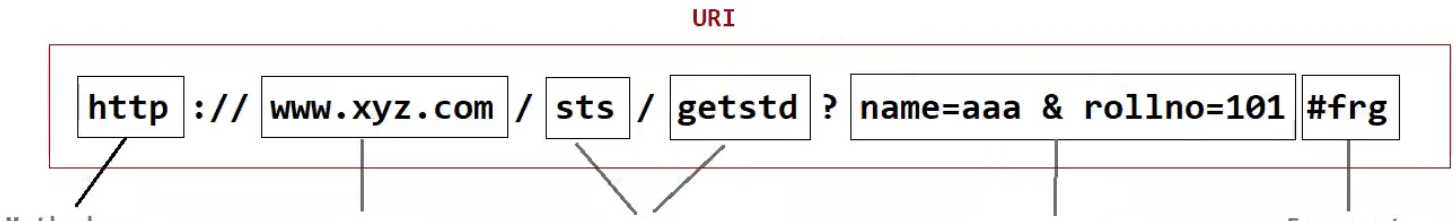
-> Microservices is software architecture used to design an application. The idea behind the application development is that "create small independent components (or services) which performs simple task, then merge or integrate them into a single unit to build a large complex application"

-> Monolithic approch was replaced by Microservices

-> Advantages of Microservices :-

= easy to scale
= fast to develop
= easy to maintain
= easy to test
= easy to deploy
etc

-> There are many technologies to develop microservices but mostly used technology is Java and its frameworks (Springboot, Spark, DropWizard, Jersey etc)


URI
http
://
www.xyz.com
/
sts
/
getstd
?
name=aaa & rollno=101
#frg
Method or Protocol
Domain (host + port number) localhost:8080
Path
Query String
Fragment

=> Resources :-
    -> Resource is the simple files i.e. html or images or data which are present on server
    -> To access the resource from server we use URI

=> HTTP :-
    -> Full form is "Hyper Text Transfer Protocol"
    -> HTTP is a protocol (TCP/IP based communication protocol) that is used to transfer the resources over www (world wide web)
    -> Other common protocols are :-
        = FTP (File Transfer Protocol)
        = SMTP (Simple Mail Transfer Protocol)
        = IMAP (Internet Message Access Protocol)
        = MIME (Multipurpose Internet Mail Extension)
          etc
    -> HTTP was developed by Tim-Berners-Lee and his team
    -> HTTP request methods are :-
        = GET
        = POST
        = PUT
        = PATCH
        = DELETE
          etc

XML Format

```
<StudentDetails>

    <student id="101">
        <name> deepak </name>
        <email> deepak@gmail.com </email>
        <gender> male </gender>
        <city> delhi </city>
    </student>

    <student id="102">
        <name> amit </name>
        <email> amit@gmail.com </email>
        <gender> male </gender>
        <city> banglore </city>
    </student>

</StudentDetails>
```

JSON Format

```
{
    {
        "name":"deepak",
        "email":"deepak@gmail.com",
        "gender":"male",
        "city":"delhi"
    },
    {
        "name":"amit",
        "email":"amit@gmail.com",
        "gender":"male",
        "city":"banglore"
    }
}
```

=> What is difference between SOAP and REST :-

= SOAP : Simple Object Access Protocol
REST : REpresentational State Transfer

= SOAP : It is a "protocol" which follows a strict standard to allow communication between the client and server
REST : It is an "architectural style" that does not follows any strict standard (but follows its own 6 constraints)

= SOAP cannot use REST because SOAP is a protocol without any architectural style and REST does not follows the protocol
REST can use SOAP because it is an architectural style which can have protocols

= SOAP transports the data in standard XML format
REST transports the data in different formats i.e. JSON, XML, Plain Text, HTML etc

= SOAP requires more bandwidth as compared to REST
REST requires less bandwidth as compared to SOAP

= SOAP is difficult to implement
REST to easy to implement

= SOAP is outdated web service
REST is traditional web service

```
=> Difference between REST and RESTful :-
    -> REST is the set of constraints. There are 6 main constraints :-
        = Client-Server Architecture
        = Stateless
        = Uniform Interface
        = Cacheable
        = Layered System
        = Code on Demand (optional)
    -> RESTful refers to the API adhering to these constraints


=> RESTful Web Services :-
    -> REST full form is
        = REpresentational - it should support JSON and XML
        = State - State means data or value of an object
        = Transfer - transfer the state using HTTP protocol
    -> REST was developed by Roy Fielding (he is the only one who provides the HTTP specifications)
    -> We can develop RESTful web services in multiple frameworks i.e. JAX-RS, Springboot, Jersey
    etc
```

Methods	API URI (http://localhost:8080/base_url)	Task
POST	/student	Add New Student
GET	/student	Get all the students
GET	/student/{id1}	Get single student according to provided id
PUT	/student	Update the student
DELETE	/student/{id1}	Delete single student according to provided id

POST http://localhost:8080/student

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "id":"104",
    "name":"ddd",
    "gender":"female",
    "city":"chandigarh"
}
```

```
//  @PostMapping("/student")
//  public Student addStudent(@RequestParam("id1") int id,
//                            @RequestParam("name1") String name,
//                            @RequestParam("gender1") String gender,
//                            @RequestParam("city1") String city)
//  {
//      Student std = new Student();
//      std.setId(id);
//      std.setName(name);
//      std.setGender(gender);
//      std.setCity(city);
//
//      Student student = studentService.addStudent(std);
//      return student;
//  }

    @PostMapping("/student")
    public Student addStudent(@RequestBody Student std)
    {
        Student student = studentService.addStudent(std);
        return student;
    }
}
```

=> Jackson :-
-> Jackson is a popular JSON (JavaScript Object Notation) processing library in java
-> It is used to convert/serialize Java Objects/Map to JSON and vice-versa
-> Features :-
1. Easy to use
2. High Performance
3. No need to create mapping
4. Integration with java frameworks
5. Open Source (free to use)
6. No Dependency
-> Jackson provides 3 ways to process the JSON :-
1. Data Binding :-
= It is a way to convert JSON to POJO and vice-versa using property accessor or using annotations
= It is of 2 types :-
a. Simple Data Binding :-
-) It converts JSON to Java Maps, Lists, Strings, Numbers, Booleans and Null Objects and vice-versa
b. Full Data Binding
-) It converts JSON from any Java type and vice-versa
= Classes or Interfaces used for this are :-
1. ObjectMapper
2. ObjectReader
3. ObjectWriter

2. Streaming API :-
= It is used to process JSON data in streaming manner, We can read and write JSON data incrementally, without loading the entire JSON structure into memory
= It is useful when dealing with large JSON documents or contineous streams of JSON data as it provides efficient and memory-friendly procressing
= It is the most powerful approach among others
= Classes or Interfaces used for this are :-
A. JsonParser
B. JsonGenerator

3. Tree Model :-
= It is used to create in-memory tree representation of the JSON document similar to DOM tree
= It is not much fast as compared to streaming API, but it is most flexible approach to read and write JSON data
= Classes or Interfaces used for this are :-
A. JsonNode
B. ObjectNode
C. ArrayNode
etc

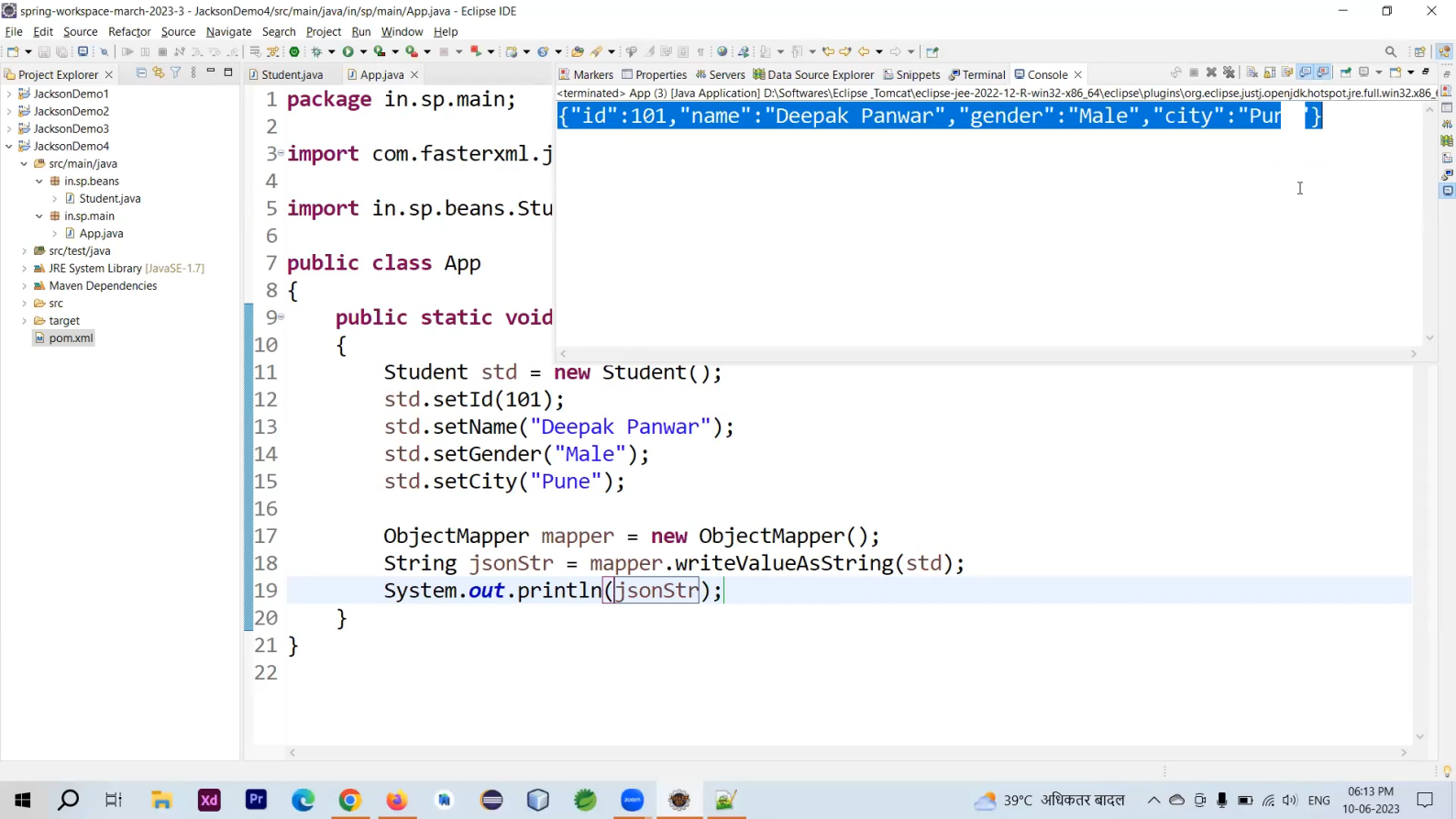
```
<!-- https://mvnrepository.com/artifact/com.fasterxml
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.14.2</version>
</dependency>
```

```
public class App
{
    public static void main( String[] args ) throws Exception
    {
        String json = "{\r\n"
                        + " \"id\" : 101,\r\n"
                        + " \"name\" : \"aaa\",\r\n"
                        + " \"gender\" : \"male\",\r\n"
                        + " \"city\" : \"chandigarh\"\r\n"
                        + "}";

        ObjectMapper objectMapper = new ObjectMapper();

        Student std = objectMapper.readValue(json, Student.class);

        System.out.println("Id : "+std.getId());
        System.out.println("Name : "+std.getName());
        System.out.println("Gender : "+std.getGender());
        System.out.println("City : "+std.getCity());
    }
}
```


spring-workspace-march-2023-3 - JacksonDemo4/src/main/java/in/sp/main/App.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo2
JacksonDemo3
JacksonDemo4
src/main/java
in.sp.beans
Student.java
in.sp.main
App.java
src/test/java
JRE System Library [JavaSE-1.7]
Maven Dependencies
src
target
pom.xml
Student.java
App.java
1 package in.sp.main;
2
3 import com.fasterxml.j
4
5 import in.sp.beans.Stu
6
7 public class App
8 {
9 public static void
10 {
11 Student std = new Student();
12 std.setId(101);
13 std.setName("Deepak Panwar");
14 std.setGender("Male");
15 std.setCity("Pune");
16
17 ObjectMapper mapper = new ObjectMapper();
18 String jsonStr = mapper.writeValueAsString(std);
19 System.out.println(jsonStr);
20 }
21 }
22
Markers Properties Servers Data Source Explorer Snippets Terminal Console
<terminated> App (3) [Java Application] D:\Softwares\Eclipse _Tomcat\eclipse-jee-2022-12-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_
{"id":101,"name":"Deepak Panwar","gender":"Male","city":"Pur }
39°C अधिकतर बादल
ENG
06:13 PM
10-06-2023

spring-workspace-march-2023-3 - JacksonDemo5/src/main/java/in/sp/main/App.java - Eclipse IDE

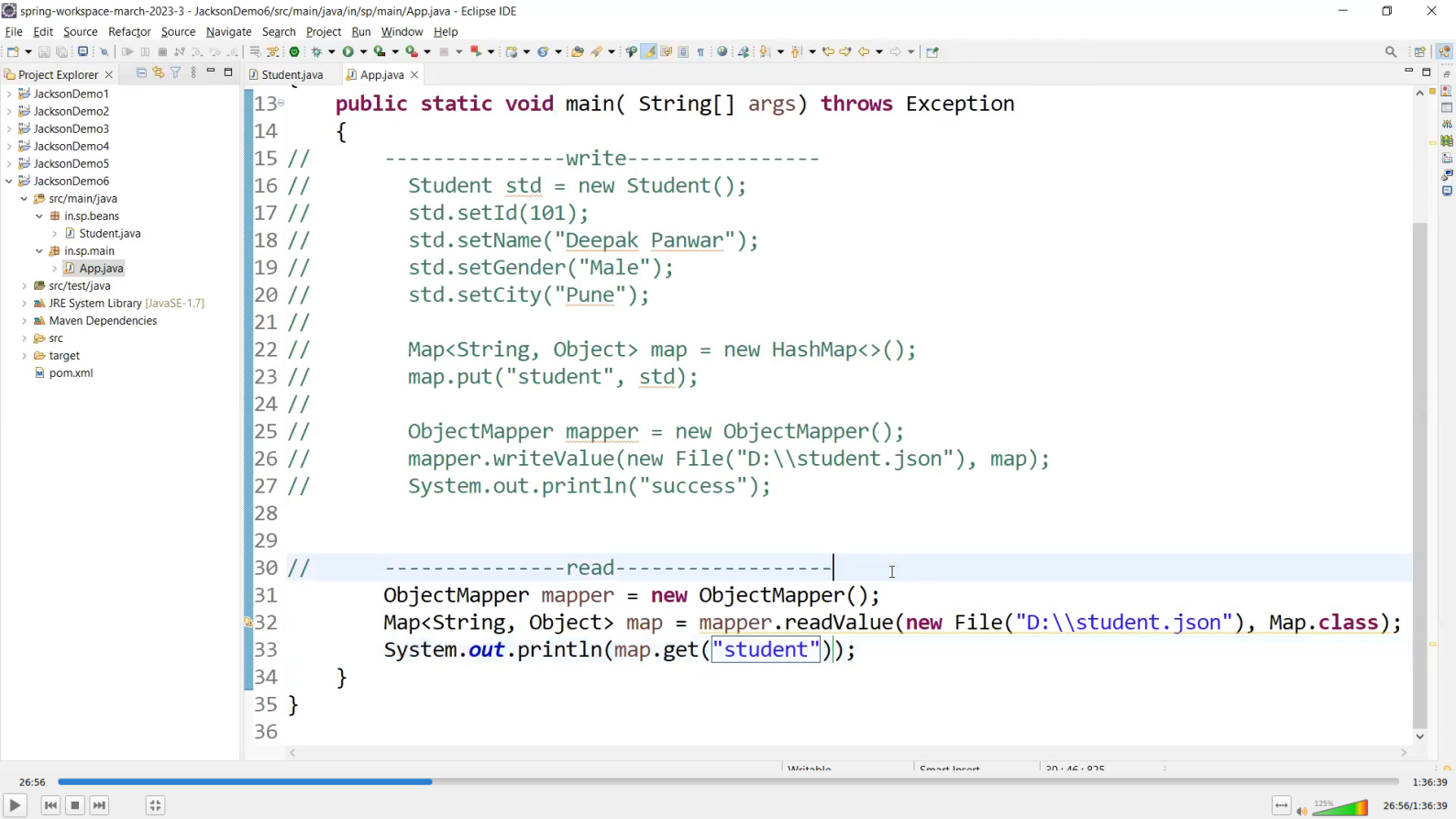
File Edit Source Refactor Source Navigate Search Project Run Window Help

App.java TypeReference.class

```
1  package in.sp.main;
2
3  import java.util.Map;
4
5  import com.fasterxml.jackson.core.type.TypeReference;
6  import com.fasterxml.jackson.databind.ObjectMapper;
7
8  public class App
9  {
10     public static void main( String[] args) throws Exception
11     {
12         String jsonStr = "{\r\n"
13                          + " \"id\" : 101,\r\n"
14                          + " \"name\" : \"aaa\",\r\n"
15                          + " \"gender\" : \"male\",\r\n"
16                          + " \"city\" : \"chandigarh\"\r\n"
17                          + "}";
18
19         ObjectMapper mapper = new ObjectMapper();
20         Map<String, Object> map = mapper.readValue(jsonStr, new TypeReference<Map<String, Object>>(){});
21         System.out.println(map);
22     }
23 }
24
```

Writable Smart Insert 20 : 32 [23]

वॉचलिस्ट सुझाव ENG 06:19 PM 10-06-2023


spring-workspace-march-2023-3 - JacksonDemo6/src/main/java/in/sp/main/App.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
src/main/java
in.sp.beans
Student.java
in.sp.main
App.java
src/test/java
JRE System Library [JavaSE-1.7]
Maven Dependencies
src
target
pom.xml
Student.java
App.java
13 public static void main(String[] args) throws Exception
14 {
15 // ---------------write----------------
16 // Student std = new Student();
17 // std.setId(101);
18 // std.setName("Deepak Panwar");
19 // std.setGender("Male");
20 // std.setCity("Pune");
21 //
22 // Map<String, Object> map = new HashMap<>();
23 // map.put("student", std);
24 //
25 // ObjectMapper mapper = new ObjectMapper();
26 // mapper.writeValue(new File("D:\\student.json"), map);
27 // System.out.println("success");
28
29
30 // ---------------read-----------------
31 ObjectMapper mapper = new ObjectMapper();
32 Map<String, Object> map = mapper.readValue(new File("D:\\student.json"), Map.class);
33 System.out.println(map.get("student"));
34 }
35 }
36
26:56
1:36:39
26:56/1:36:39

spring-workspace-march-2023-3 - JacksonDemo7/src/main/java/in/sp/main/App.java - Eclipse IDE

File Edit Source Refactor Source Navigate Search Project Run Window Help

Project Explorer

- JacksonDemo1
- JacksonDemo2
  - src/main/java
    - in.sp.beans
    - in.sp.main
      - App.java
  - src/test/java
  - JRE System Library [JavaSE-1.7]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- JacksonDemo3
- JacksonDemo4
- JacksonDemo5
- JacksonDemo6
- JacksonDemo7
  - src/main/java
    - in.sp.beans
      - Posts.java
    - in.sp.main
      - App.java
  - src/test/java
  - JRE System Library [JavaSE-1.7]
  - Maven Dependencies
  - src
  - target
  - pom.xml

App.java | Posts.java

```
3  import java.net.URL;
4
5  import com.fasterxml.jackson.databind.ObjectMapper;
6
7  import in.sp.beans.Posts;
8
9  public class App
10 {
11     public static void main( String[] args) throws Exception
12     {
13         String url = "https://jsonplaceholder.typicode.com/posts";
14
15         ObjectMapper mapper = new ObjectMapper();
16         Posts[] posts_arr = mapper.readValue(new URL(url), Posts[].class);
17         for(Posts posts : posts_arr)
18         {
19             System.out.println("Userid : "+posts.getUserId());
20             System.out.println("Id : "+posts.getId());
21             System.out.println("Title : "+posts.getTitle());
22             System.out.println("Body : "+posts.getBody());
23             System.out.println("---------------------------------------");
24         }
25     }
26 }
```

Writable | Smart Insert | 24 : 10 : 751

39°C अधिकतर बादल ENG 06:40 PM 10-06-2023

spring-workspace-march-2023-3 - JacksonDemo7/src/main/java/in/sp/beans/Posts.java - Eclipse IDE

File Edit Source Refactor Source Navigate Search Project Run Window Help

Project Explorer

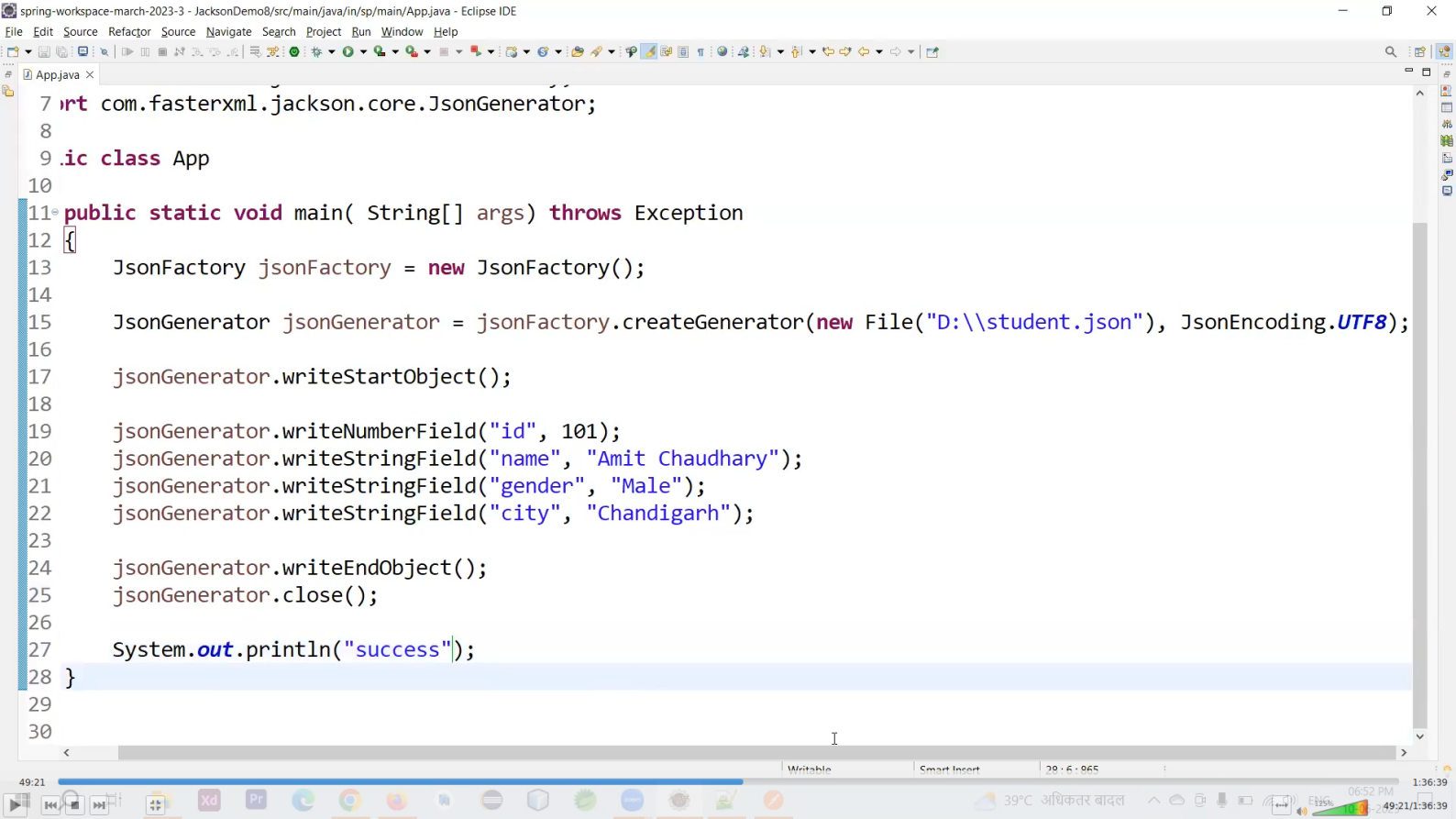
- JacksonDemo1
- JacksonDemo2
- JacksonDemo3
- JacksonDemo4
- JacksonDemo5
- JacksonDemo6
- JacksonDemo7
  - src/main/java
    - in.sp.beans
      - Posts.java
    - in.sp.main
      - App.java
  - src/test/java
  - JRE System Library [JavaSE-1.7]
  - Maven Dependencies
  - src
  - target
  - pom.xml

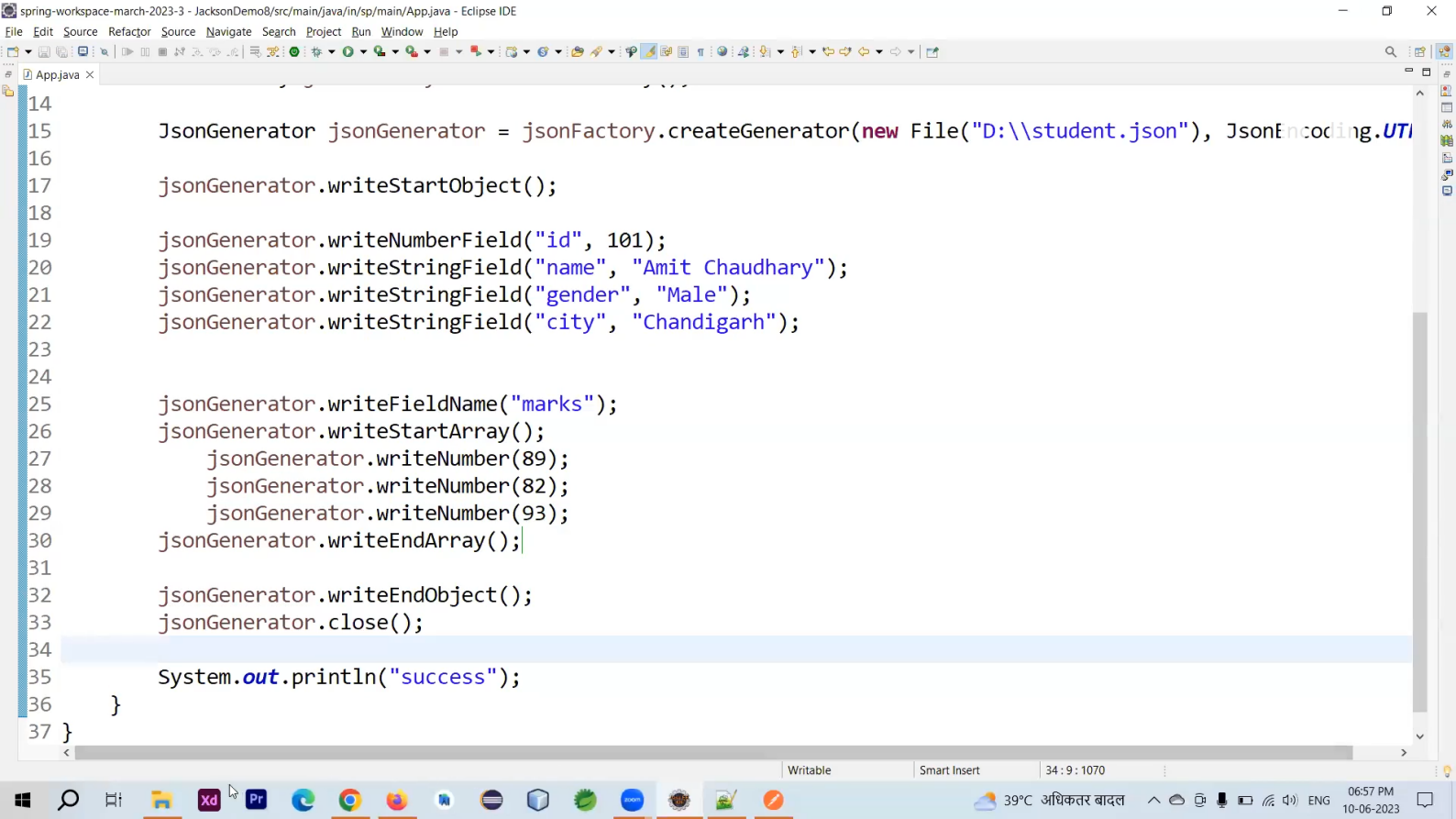
App.java | Posts.java

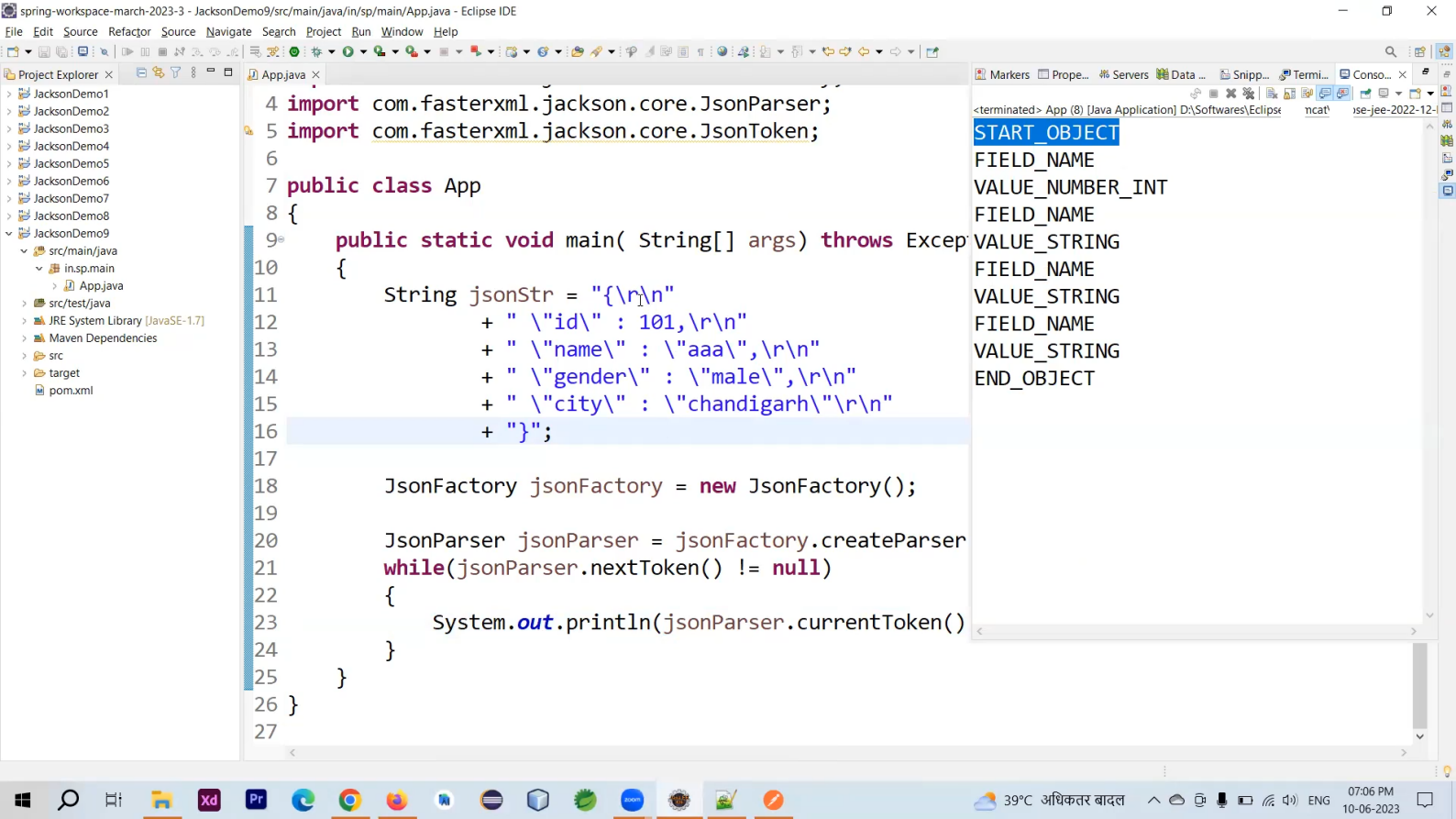
```
1  package in.sp.beans;
2
3  public class Posts
4  {
5      private int userId;
6      private int id;
7      private String title;
8      private String body;
9
10     public int getUserId() {
11         return userId;
12     }
13     public void setUserId(int userId) {
14         this.userId = userId;
15     }
16     public int getId() {
17         return id;
18     }
19     public void setId(int id) {
20         this.id = id;
21     }
22     public String getTitle() {
23         return title;
24     }
```

Writable | Smart Insert | 9 : 5 : 136

39°C अधिकतर बादल ENG 06:36 PM 10-06-2023


spring-workspace-march-2023-3 - JacksonDemo8/src/main/java/in/sp/main/App.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
App.java
rt com.fasterxml.jackson.core.JsonGenerator;
ic class App
public static void main(String[] args) throws Exception
{
JsonFactory jsonFactory = new JsonFactory();
JsonGenerator jsonGenerator = jsonFactory.createGenerator(new File("D:\\student.json"), JsonEncoding.UTF8);
jsonGenerator.writeStartObject();
jsonGenerator.writeNumberField("id", 101);
jsonGenerator.writeStringField("name", "Amit Chaudhary");
jsonGenerator.writeStringField("gender", "Male");
jsonGenerator.writeStringField("city", "Chandigarh");
jsonGenerator.writeEndObject();
jsonGenerator.close();
System.out.println("success");
}
Writable
Smart Insert
28 : 6 : 865
49:21
1:36:39


spring-workspace-march-2023-3 - JacksonDemo8/src/main/java/in/sp/main/App.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
App.java
JsonGenerator jsonGenerator = jsonFactory.createGenerator(new File("D:\\student.json"), JsonE
jsonGenerator.writeStartObject();
jsonGenerator.writeNumberField("id", 101);
jsonGenerator.writeStringField("name", "Amit Chaudhary");
jsonGenerator.writeStringField("gender", "Male");
jsonGenerator.writeStringField("city", "Chandigarh");
jsonGenerator.writeFieldName("marks");
jsonGenerator.writeStartArray();
jsonGenerator.writeNumber(89);
jsonGenerator.writeNumber(82);
jsonGenerator.writeNumber(93);
jsonGenerator.writeEndArray();
jsonGenerator.writeEndObject();
jsonGenerator.close();
System.out.println("success");
}
}
Writable
Smart Insert
34 : 9 : 1070
39°C अधिकतर बादल
ENG
06:57 PM
10-06-2023


spring-workspace-march-2023-3 - JacksonDemo9/src/main/java/in/sp/main/App.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
JacksonDemo7
JacksonDemo8
JacksonDemo9
src/main/java
in.sp.main
App.java
src/test/java
JRE System Library [JavaSE-1.7]
Maven Dependencies
src
target
pom.xml
App.java
4 import com.fasterxml.jackson.core.JsonParser;
5 import com.fasterxml.jackson.core.JsonToken;
6
7 public class App
8 {
9 public static void main(String[] args) throws Excep
10 {
11 String jsonStr = "{\r\n"
12 + " \"id\" : 101,\r\n"
13 + " \"name\" : \"aaa\",\r\n"
14 + " \"gender\" : \"male\",\r\n"
15 + " \"city\" : \"chandigarh\"\r\n"
16 + "}";
17
18 JsonFactory jsonFactory = new JsonFactory();
19
20 JsonParser jsonParser = jsonFactory.createParser
21 while(jsonParser.nextToken() != null)
22 {
23 System.out.println(jsonParser.currentToken()
24 }
25 }
26 }
27
Markers Prope... Servers Data ... Snipp... Termi... Conso...
<terminated> App (8) [Java Application] D:\Softwares\Eclipse ncat\ se-jee-2022-12-
START_OBJECT
FIELD_NAME
VALUE_NUMBER_INT
FIELD_NAME
VALUE_STRING
FIELD_NAME
VALUE_STRING
FIELD_NAME
VALUE_STRING
END_OBJECT
39°C अधिकतर बादल
ENG
07:06 PM
10-06-2023

spring-workspace-march-2023-3 - JacksonDemo9/src/main/java/in/sp/main/App.java - Eclipse IDE

File Edit Source Refactor Source Navigate Search Project Run Window Help

Project Explorer

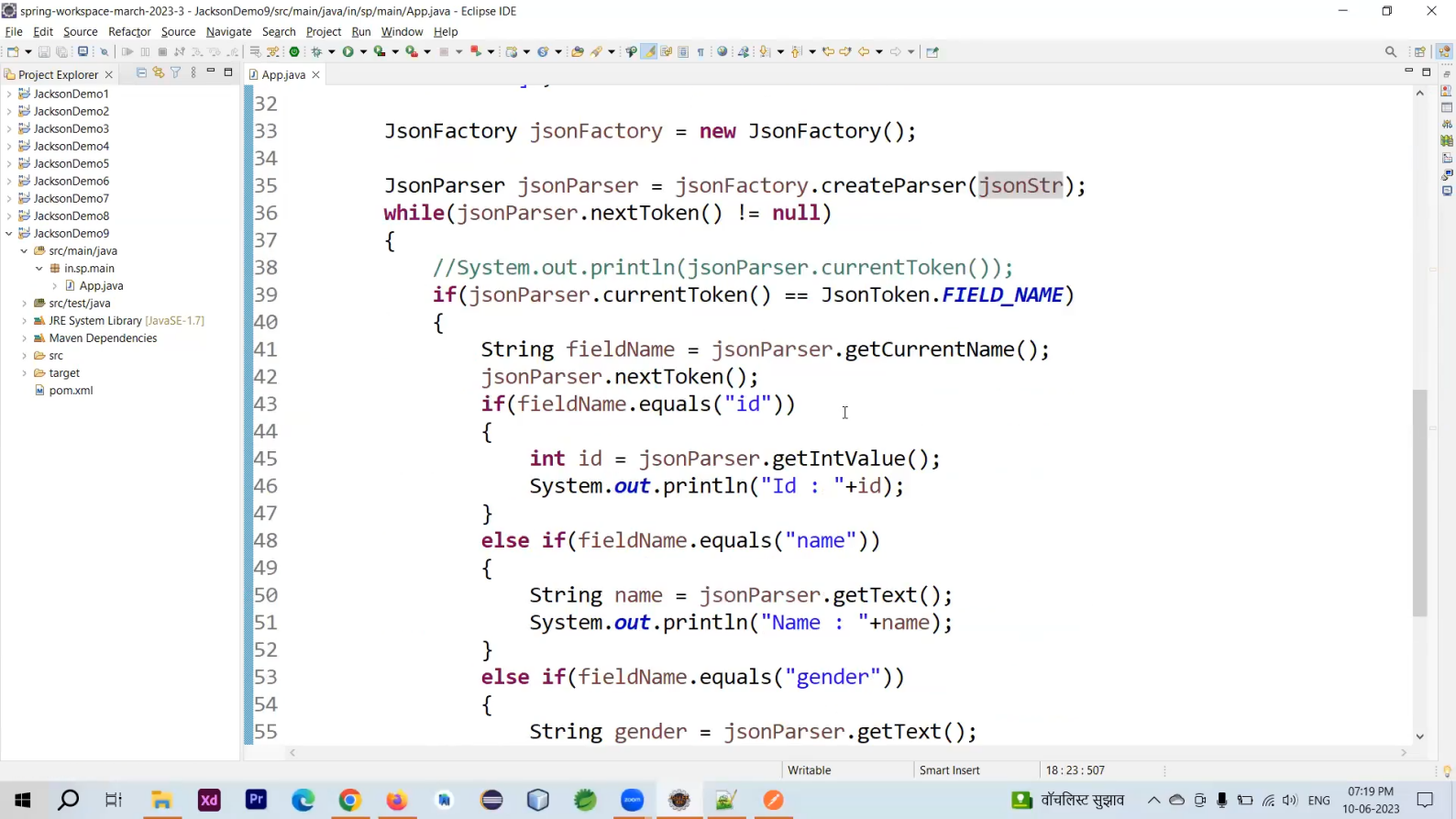
- JacksonDemo1
- JacksonDemo2
- JacksonDemo3
- JacksonDemo4
- JacksonDemo5
- JacksonDemo6
- JacksonDemo7
- JacksonDemo8
- JacksonDemo9
  - src/main/java
    - in.sp.main
      - App.java
  - src/test/java
  - JRE System Library [JavaSE-1.7]
  - Maven Dependencies
  - src
  - target
  - pom.xml

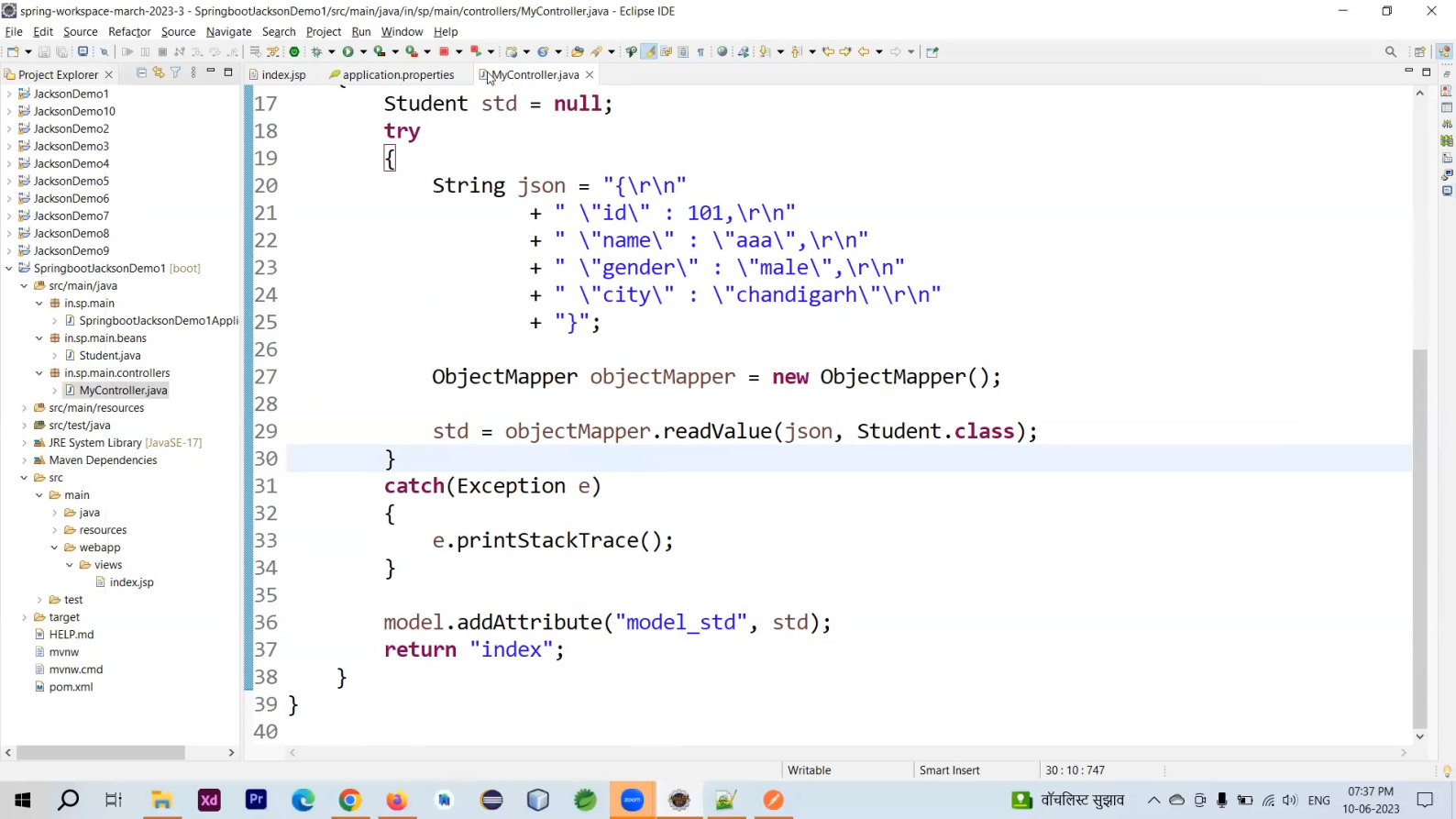
App.java

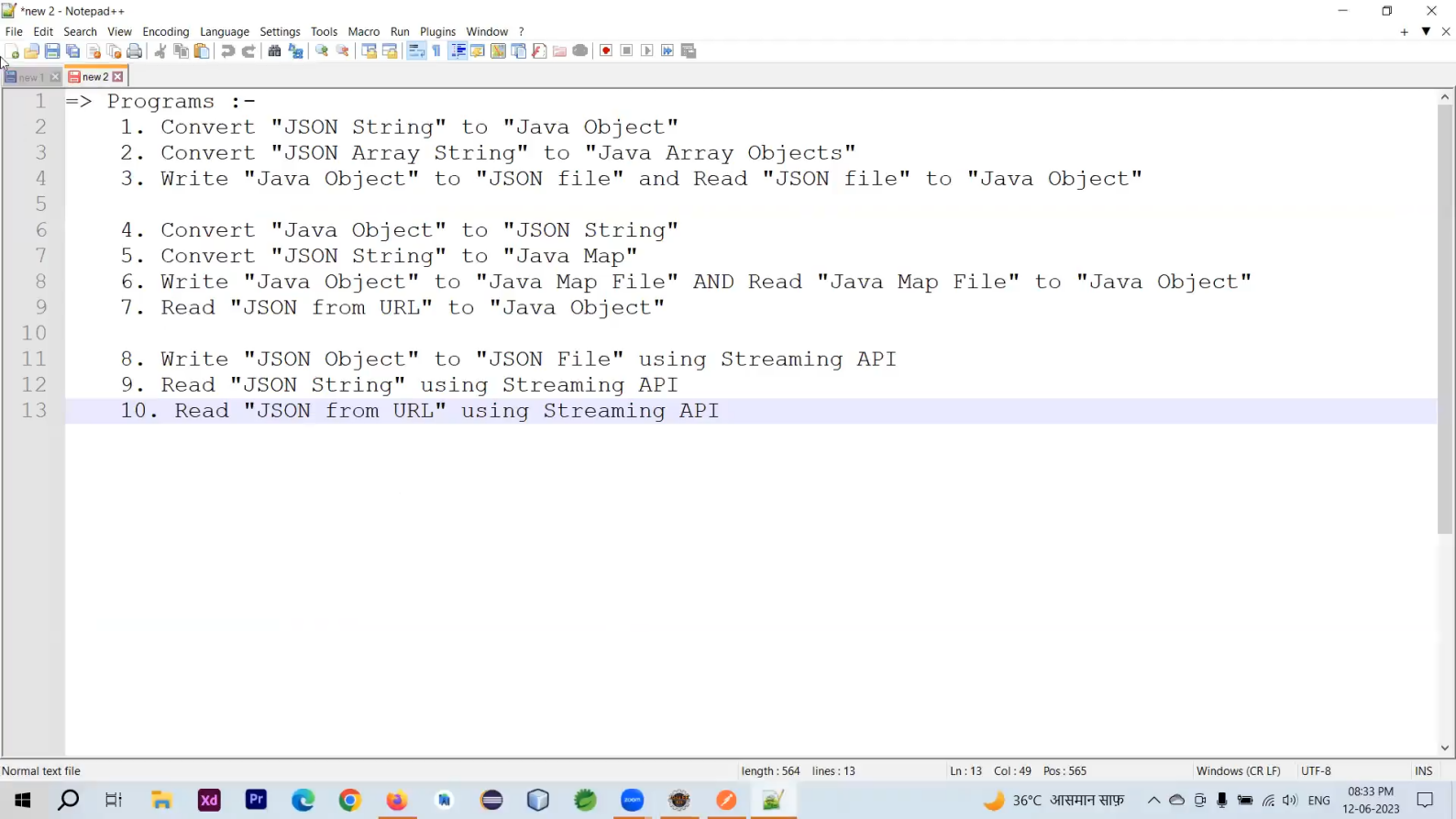
```
 9    public static void main( String[] args) throws Exception
10    {
11 //      String jsonStr = "{\r\n"
12 //              + " \"id\" : 101,\r\n"
13 //              + " \"name\" : \"aaa\",\r\n"
14 //              + " \"gender\" : \"male\",\r\n"
15 //              + " \"city\" : \"chandigarh\"\r\n"
16 //              + "}";
17
18        String jsonStr = "[\r\n"
19                + " {\r\n"
20                + "     \"id\" : 101,\r\n"
21                + "     \"name\" : \"aaa\",\r\n"
22                + "     \"gender\" : \"male\",\r\n"
23                + "     \"city\" : \"delhi\"\r\n"
24                + " },\r\n"
25                + " {\r\n"
26                + "     \"id\" : 102,\r\n"
27                + "     \"name\" : \"bbb\",\r\n"
28                + "     \"gender\" : \"female\",\r\n"
29                + "     \"city\" : \"pune\"\r\n"
30                + " }\r\n"
31                + "]";
32
```

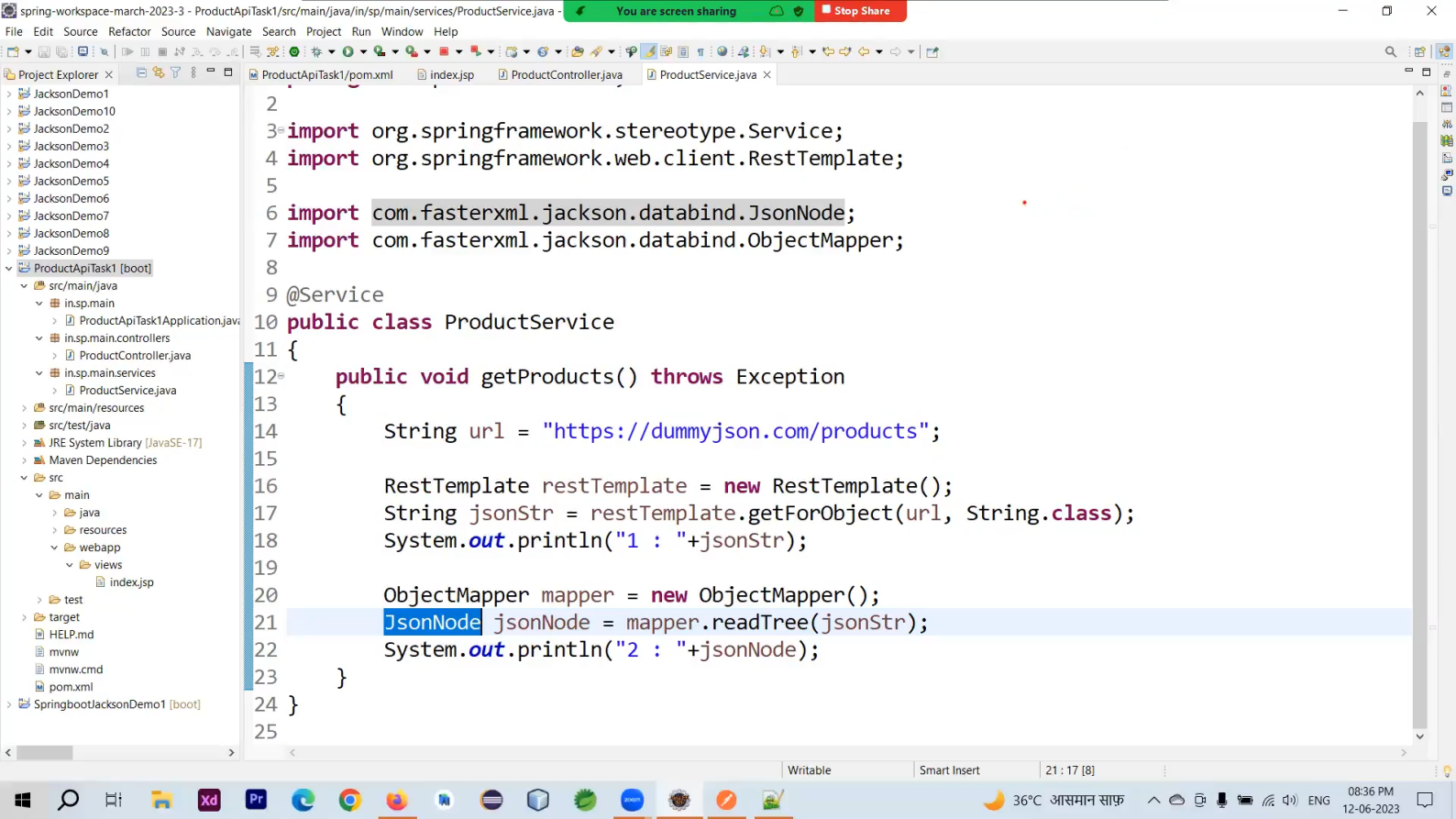
Writable | Smart Insert | 66 : 34 : 1966

वॉचलिस्ट सुझाव ENG 07:20 PM 10-06-2023


spring-workspace-march-2023-3 - JacksonDemo9/src/main/java/in/sp/main/App.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
JacksonDemo7
JacksonDemo8
JacksonDemo9
src/main/java
in.sp.main
App.java
src/test/java
JRE System Library [JavaSE-1.7]
Maven Dependencies
src
target
pom.xml
App.java
32
33 JsonFactory jsonFactory = new JsonFactory();
34
35 JsonParser jsonParser = jsonFactory.createParser(jsonStr);
36 while(jsonParser.nextToken() != null)
37 {
38 //System.out.println(jsonParser.currentToken());
39 if(jsonParser.currentToken() == JsonToken.FIELD_NAME)
40 {
41 String fieldName = jsonParser.getCurrentName();
42 jsonParser.nextToken();
43 if(fieldName.equals("id"))
44 {
45 int id = jsonParser.getIntValue();
46 System.out.println("Id : "+id);
47 }
48 else if(fieldName.equals("name"))
49 {
50 String name = jsonParser.getText();
51 System.out.println("Name : "+name);
52 }
53 else if(fieldName.equals("gender"))
54 {
55 String gender = jsonParser.getText();
Writable
Smart Insert
18 : 23 : 507
वॉचलिस्ट सुझाव
ENG
07:19 PM
10-06-2023


spring-workspace-march-2023-3 - SpringbootJacksonDemo1/src/main/java/in/sp/main/controllers/MyController.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo10
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
JacksonDemo7
JacksonDemo8
JacksonDemo9
SpringbootJacksonDemo1 [boot]
src/main/java
in.sp.main
SpringbootJacksonDemo1Appli
in.sp.main.beans
Student.java
in.sp.main.controllers
MyController.java
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
main
java
resources
webapp
views
index.jsp
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml
index.jsp
application.properties
MyController.java
17 Student std = null;
18 try
19 {
20 String json = "{\r\n"
21 + " \"id\" : 101,\r\n"
22 + " \"name\" : \"aaa\",\r\n"
23 + " \"gender\" : \"male\",\r\n"
24 + " \"city\" : \"chandigarh\"\r\n"
25 + "}";
26
27 ObjectMapper objectMapper = new ObjectMapper();
28
29 std = objectMapper.readValue(json, Student.class);
30 }
31 catch(Exception e)
32 {
33 e.printStackTrace();
34 }
35
36 model.addAttribute("model_std", std);
37 return "index";
38 }
39 }
40
Writable
Smart Insert
30 : 10 : 747
वॉचलिस्ट सुझाव
ENG
07:37 PM
10-06-2023


*new 2 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 1
new 2
=> Programs :-
1. Convert "JSON String" to "Java Object"
2. Convert "JSON Array String" to "Java Array Objects"
3. Write "Java Object" to "JSON file" and Read "JSON file" to "Java Object"
4. Convert "Java Object" to "JSON String"
5. Convert "JSON String" to "Java Map"
6. Write "Java Object" to "Java Map File" AND Read "Java Map File" to "Java Object"
7. Read "JSON from URL" to "Java Object"
8. Write "JSON Object" to "JSON File" using Streaming API
9. Read "JSON String" using Streaming API
10. Read "JSON from URL" using Streaming API
Normal text file
length : 564 lines : 13
Ln : 13 Col : 49 Pos : 565
Windows (CR LF)
UTF-8
INS
36°C आसमान साफ़
ENG
08:33 PM
12-06-2023


spring-workspace-march-2023-3 - ProductApiTask1/src/main/java/in/sp/main/services/ProductService.java -
You are screen sharing
Stop Share
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo10
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
JacksonDemo7
JacksonDemo8
JacksonDemo9
ProductApiTask1 [boot]
src/main/java
in.sp.main
ProductApiTask1Application.java
in.sp.main.controllers
ProductController.java
in.sp.main.services
ProductService.java
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
main
java
resources
webapp
views
index.jsp
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml
SpringbootJacksonDemo1 [boot]
ProductApiTask1/pom.xml
index.jsp
ProductController.java
ProductService.java
2
3 import org.springframework.stereotype.Service;
4 import org.springframework.web.client.RestTemplate;
5
6 import com.fasterxml.jackson.databind.JsonNode;
7 import com.fasterxml.jackson.databind.ObjectMapper;
8
9 @Service
10 public class ProductService
11 {
12 public void getProducts() throws Exception
13 {
14 String url = "https://dummyjson.com/products";
15
16 RestTemplate restTemplate = new RestTemplate();
17 String jsonStr = restTemplate.getForObject(url, String.class);
18 System.out.println("1 : "+jsonStr);
19
20 ObjectMapper mapper = new ObjectMapper();
21 JsonNode jsonNode = mapper.readTree(jsonStr);
22 System.out.println("2 : "+jsonNode);
23 }
24 }
25
Writable
Smart Insert
21 : 17 [8]
36°C आसमान साफ़
ENG
08:36 PM
12-06-2023

spring-workspace-march-2023-3 - ProductApiTask2/src/main/java/in/sp/main/controllers/ProductController.java - Eclipse IDE

File Edit Source Refactor Source Navigate Search Project Run Window Help

Project Explorer

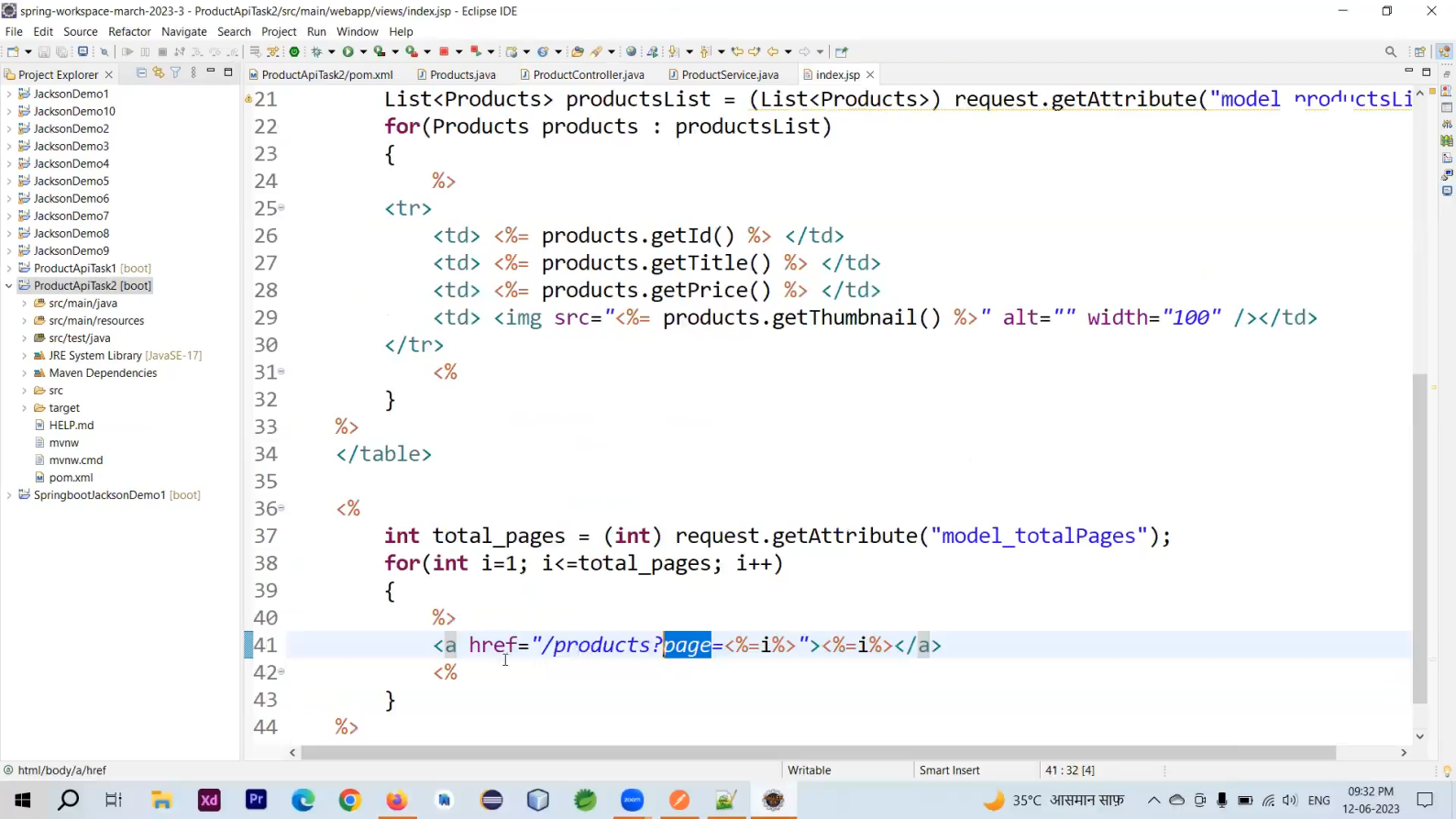
- JacksonDemo1
- JacksonDemo10
- JacksonDemo2
- JacksonDemo3
- JacksonDemo4
- JacksonDemo5
- JacksonDemo6
- JacksonDemo7
- JacksonDemo8
- JacksonDemo9
- ProductApiTask1 [boot]
- ProductApiTask2 [boot]
  - src/main/java
  - src/main/resources
  - src/test/java
  - JRE System Library [JavaSE-17]
  - Maven Dependencies
  - src
  - target
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
- SpringbootJacksonDemo1 [boot]

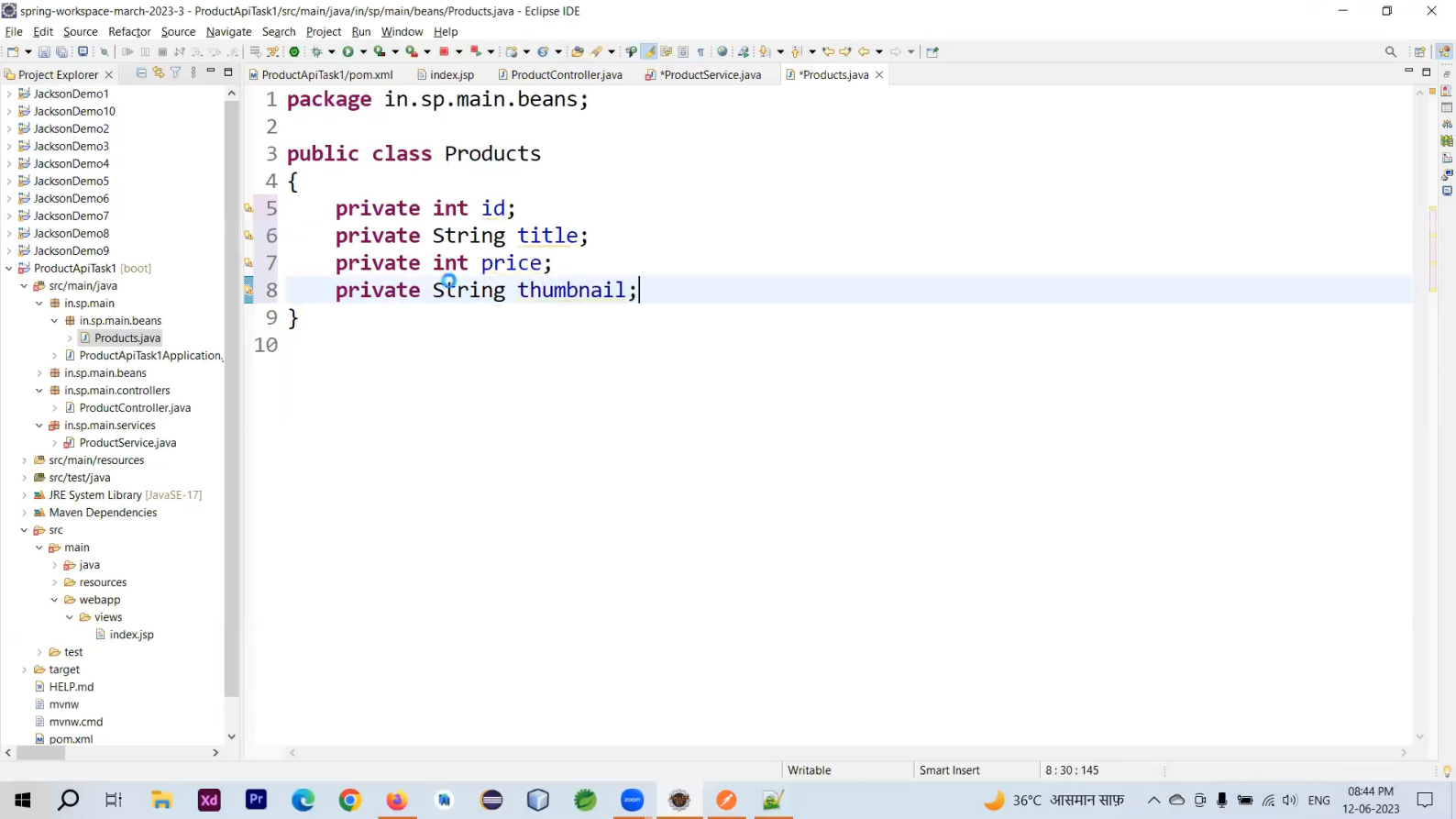
ProductApiTask2/pom.xml | Products.java | ProductController.java | ProductService.java | index.jsp

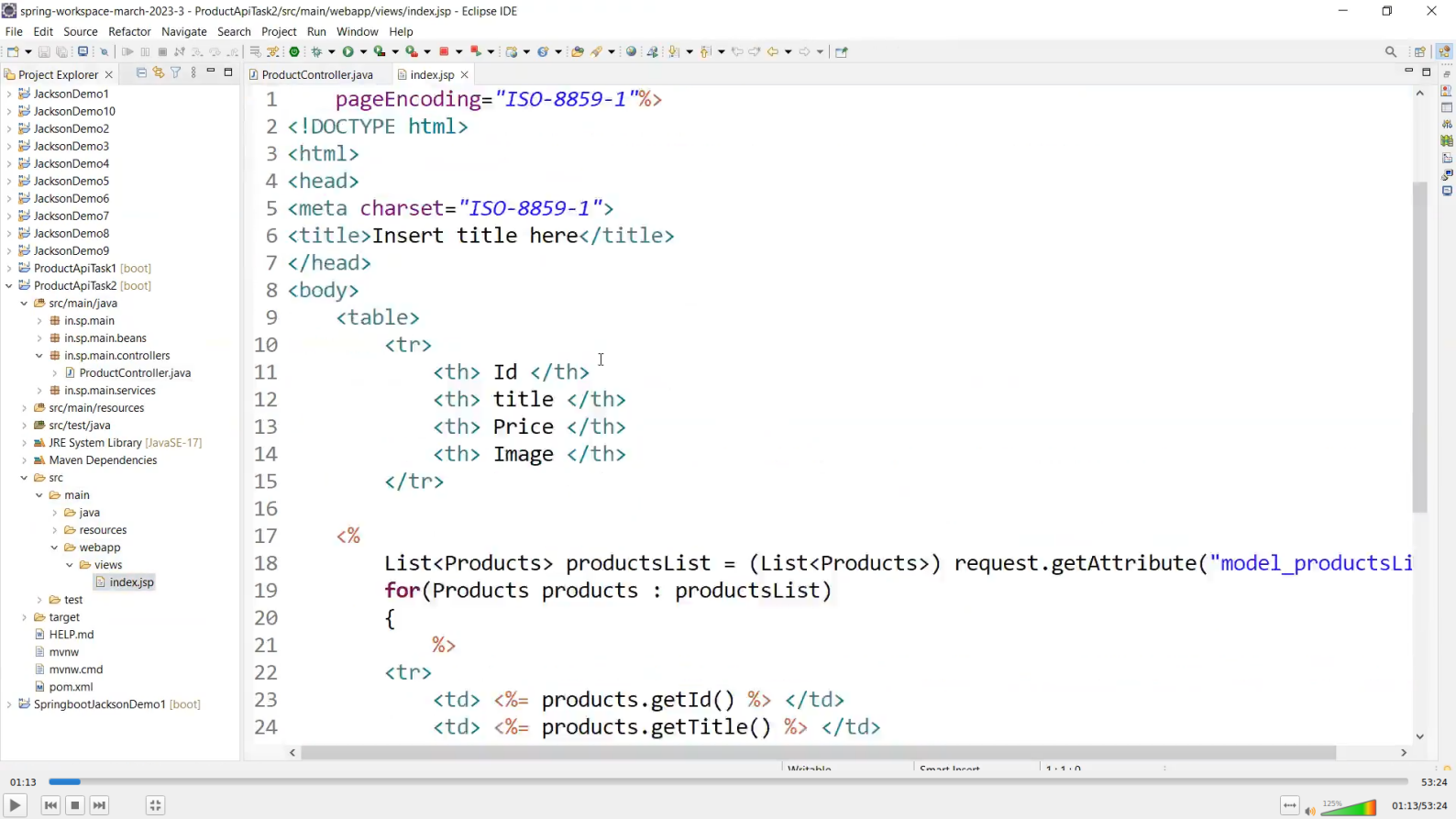
```
17      private static final int PAGE_SIZE = 10;
18
19      @Autowired
20      private ProductService productService;
21
22      @GetMapping("/products")
23      public String getProductsList(Model model, @RequestParam(defaultValue = "1") int page) th
24      {
25          System.out.println("1111111");
26          List<Products> products_list = productService.getProducts();
27          System.out.println("2222222");
28          int total_products = products_list.size();
29          int total_pages = total_products / PAGE_SIZE;
30          int start_index = (page-1)*PAGE_SIZE;           //0     -   10  -   20
31          int end_index = start_index+PAGE_SIZE - 1;      //9     -   19  -   29
32
33          model.addAttribute("model_productsList", products_list.subList(start_index, end_index
34          model.addAttribute("model_totalPages", total_pages);
35  //      model.addAttribute("model_currentPage", page);
36
37          return "index";
38      }
39  }
40
```

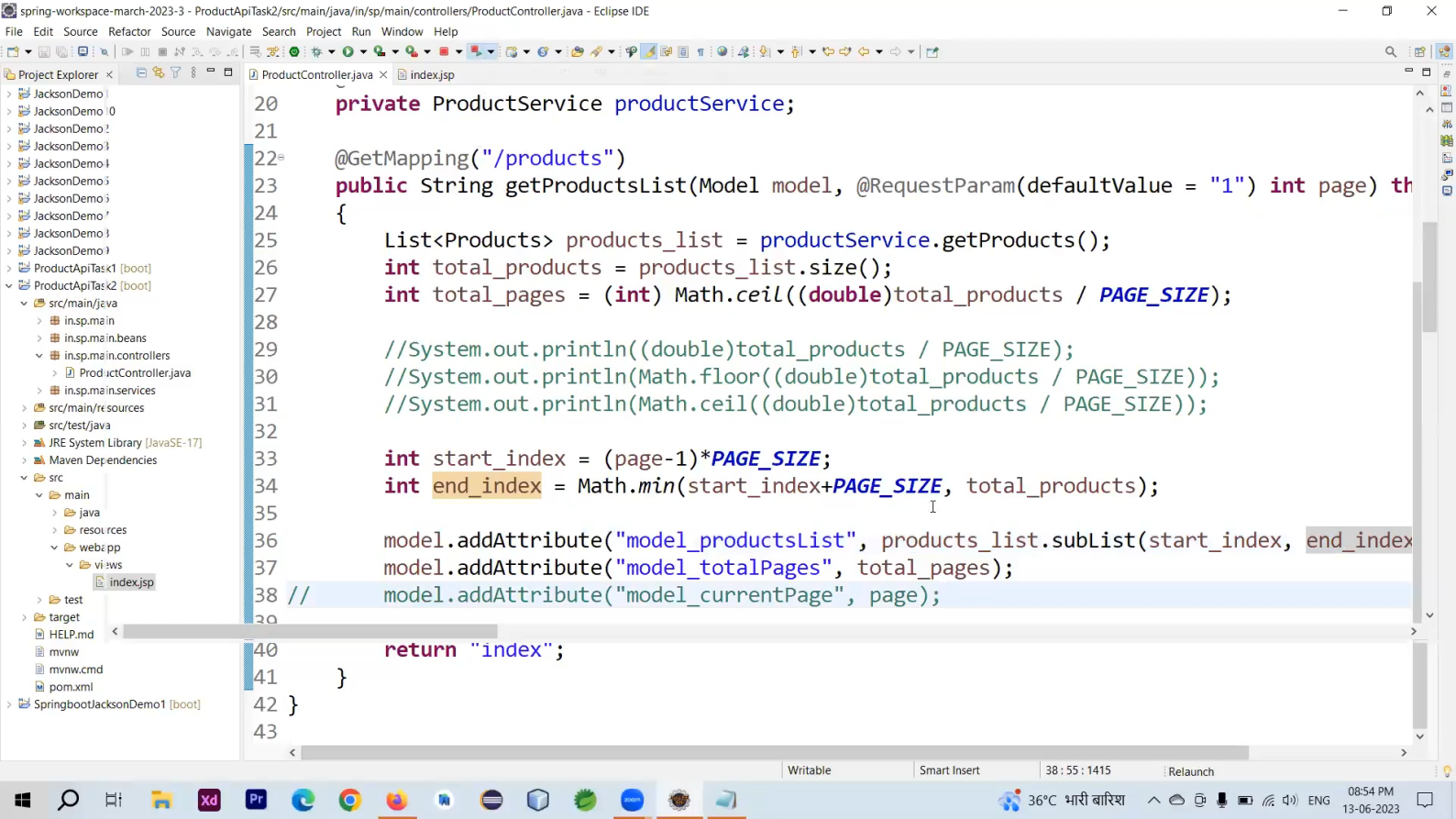
Writable | Smart Insert | 31 : 79 : 1053

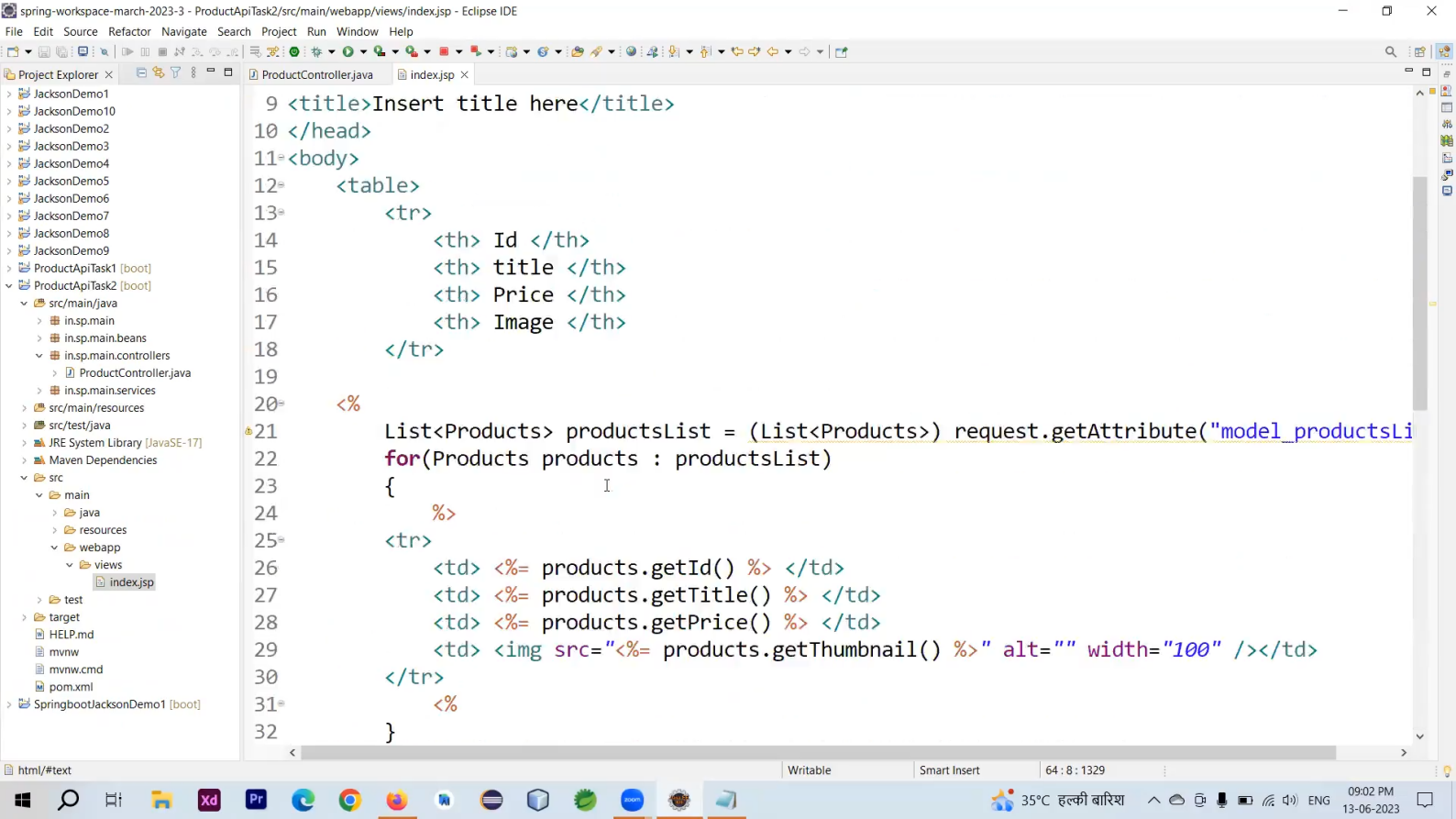
35°C आसमान साफ़ ENG 09:30 PM 12-06-2023

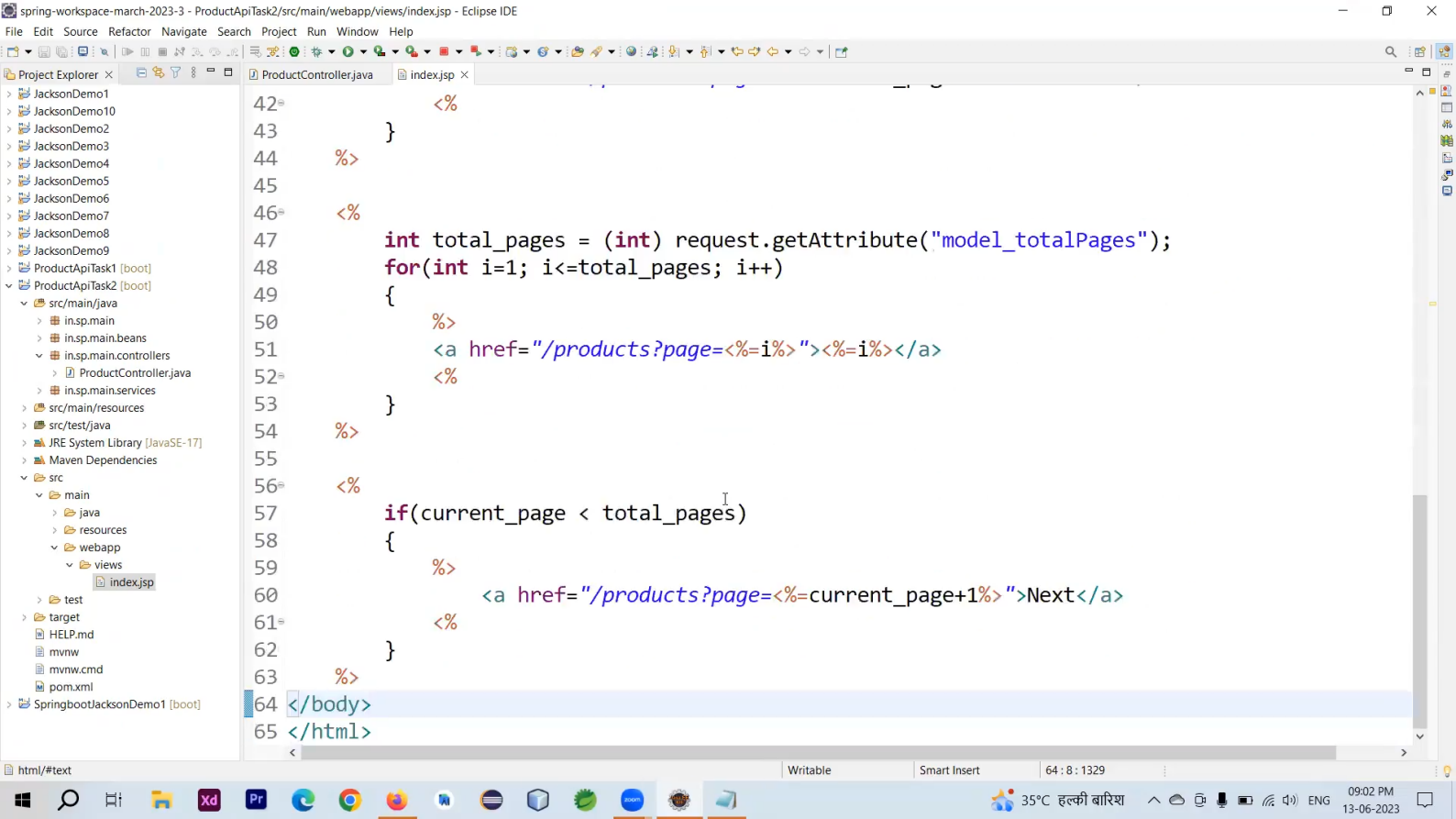

spring-workspace-march-2023-3 - ProductApiTask2/src/main/webapp/views/index.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo10
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
JacksonDemo7
JacksonDemo8
JacksonDemo9
ProductApiTask1 [boot]
ProductApiTask2 [boot]
src/main/java
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml
SpringbootJacksonDemo1 [boot]
ProductApiTask2/pom.xml
Products.java
ProductController.java
ProductService.java
index.jsp
21 List<Products> productsList = (List<Products>) request.getAttribute("model productsLi
22 for(Products products : productsList)
23 {
24 %>
25 <tr>
26 <td> <%= products.getId() %> </td>
27 <td> <%= products.getTitle() %> </td>
28 <td> <%= products.getPrice() %> </td>
29 <td> <img src="<%= products.getThumbnail() %>" alt="" width="100" /></td>
30 </tr>
31 <%
32 }
33 %>
34 </table>
35
36 <%
37 int total_pages = (int) request.getAttribute("model_totalPages");
38 for(int i=1; i<=total_pages; i++)
39 {
40 %>
41 <a href="/products?page=<%=i%>"><%=i%>
42 <%
43 }
44 %>
html/body/a/href
Writable
Smart Insert
41 : 32 [4]
35°C आसमान साफ़
ENG
09:32 PM
12-06-2023


spring-workspace-march-2023-3 - ProductApiTask1/src/main/java/in/sp/main/beans/Products.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo10
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
JacksonDemo7
JacksonDemo8
JacksonDemo9
ProductApiTask1 [boot]
src/main/java
in.sp.main
in.sp.main.beans
Products.java
ProductApiTask1Application.
in.sp.main.beans
in.sp.main.controllers
ProductController.java
in.sp.main.services
ProductService.java
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
main
java
resources
webapp
views
index.jsp
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml
ProductApiTask1/pom.xml
index.jsp
ProductController.java
*ProductService.java
*Products.java
1 package in.sp.main.beans;
2
3 public class Products
4 {
5 private int id;
6 private String title;
7 private int price;
8 private String thumbnail;
9 }
10
Writable
Smart Insert
8 : 30 : 145
36°C आसमान साफ़
ENG
08:44 PM
12-06-2023


spring-workspace-march-2023-3 - ProductApiTask2/src/main/webapp/views/index.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo10
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
JacksonDemo7
JacksonDemo8
JacksonDemo9
ProductApiTask1 [boot]
ProductApiTask2 [boot]
src/main/java
in.sp.main
in.sp.main.beans
in.sp.main.controllers
ProductController.java
in.sp.main.services
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
main
java
resources
webapp
views
index.jsp
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml
SpringbootJacksonDemo1 [boot]
ProductController.java
index.jsp
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<table>
<tr>
<th> Id </th>
<th> title </th>
<th> Price </th>
<th> Image </th>
</tr>
<%
List<Products> productsList = (List<Products>) request.getAttribute("model_productsLi
for(Products products : productsList)
{
%>
<tr>
<td> <%= products.getId() %> </td>
<td> <%= products.getTitle() %> </td>
01:13
53:24
01:13/53:24


spring-workspace-march-2023-3 - ProductApiTask2/src/main/java/in/sp/main/controllers/ProductController.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer
ProductController.java
index.jsp
20 private ProductService productService;
21
22 @GetMapping("/products")
23 public String getProductsList(Model model, @RequestParam(defaultValue = "1") int page) th
24 {
25 List<Products> products_list = productService.getProducts();
26 int total_products = products_list.size();
27 int total_pages = (int) Math.ceil((double)total_products / PAGE_SIZE);
28
29 //System.out.println((double)total_products / PAGE_SIZE);
30 //System.out.println(Math.floor((double)total_products / PAGE_SIZE));
31 //System.out.println(Math.ceil((double)total_products / PAGE_SIZE));
32
33 int start_index = (page-1)*PAGE_SIZE;
34 int end_index = Math.min(start_index+PAGE_SIZE, total_products);
35
36 model.addAttribute("model_productsList", products_list.subList(start_index, end_index
37 model.addAttribute("model_totalPages", total_pages);
38 // model.addAttribute("model_currentPage", page);
40 return "index";
41 }
42 }
43
Writable
Smart Insert
38 : 55 : 1415
Relaunch
36°C भारी बारिश
ENG
08:54 PM
13-06-2023


spring-workspace-march-2023-3 - ProductApiTask2/src/main/webapp/views/index.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo10
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
JacksonDemo7
JacksonDemo8
JacksonDemo9
ProductApiTask1 [boot]
ProductApiTask2 [boot]
src/main/java
in.sp.main
in.sp.main.beans
in.sp.main.controllers
ProductController.java
in.sp.main.services
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
main
java
resources
webapp
views
index.jsp
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml
SpringbootJacksonDemo1 [boot]
ProductController.java
index.jsp
9 <title>Insert title here</title>
10 </head>
11 <body>
12 <table>
13 <tr>
14 <th> Id </th>
15 <th> title </th>
16 <th> Price </th>
17 <th> Image </th>
18 </tr>
19
20 <%
21 List<Products> productsList = (List<Products>) request.getAttribute("model_productsLi
22 for(Products products : productsList)
23 {
24 %>
25 <tr>
26 <td> <%= products.getId() %> </td>
27 <td> <%= products.getTitle() %> </td>
28 <td> <%= products.getPrice() %> </td>
29 <td> <img src="<%= products.getThumbnail() %>" alt="" width="100" /></td>
30 </tr>
31 <%
32 }
html/#text
Writable
Smart Insert
64 : 8 : 1329
35°C हल्की बारिश
ENG
09:02 PM
13-06-2023


spring-workspace-march-2023-3 - ProductApiTask2/src/main/webapp/views/index.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
JacksonDemo1
JacksonDemo10
JacksonDemo2
JacksonDemo3
JacksonDemo4
JacksonDemo5
JacksonDemo6
JacksonDemo7
JacksonDemo8
JacksonDemo9
ProductApiTask1 [boot]
ProductApiTask2 [boot]
src/main/java
in.sp.main
in.sp.main.beans
in.sp.main.controllers
ProductController.java
in.sp.main.services
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
main
java
resources
webapp
views
index.jsp
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml
SpringbootJacksonDemo1 [boot]
ProductController.java
index.jsp
<%
}
%>
<%
int total_pages = (int) request.getAttribute("model_totalPages");
for(int i=1; i<=total_pages; i++)
{
%>
<a href="/products?page=<%=i%>"><%=i%>
<%
}
%>
<%
if(current_page < total_pages)
{
%>
<a href="/products?page=<%=current_page+1%>">Next
<%
}
%>
</body>
</html>
html/#text
Writable
Smart Insert
64 : 8 : 1329
35°C हल्की बारिश
ENG
09:02 PM
13-06-2023

```
    <tr>
        <th> Id </th>
        <th> title </th>
        <th> Price </th>
        <th> Image </th>
    </tr>
    <c:forEach var="productsList" items="${model_productsList}">
        <tr>
            <td> ${productsList.getId()} </td>
            <td> ${productsList.getTitle()} </td>
            <td> ${productsList.getPrice()} </td>
            <td> <img src="${productsList.getThumbnail()}" alt="" width="100" /> </td>
        </tr>
    </c:forEach>
</table>
```

spring-workspace-march-2023-3 - ProductApiTask3/src/main/webapp/views/index.jsp - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- JacksonDemo1
- JacksonDemo10
- JacksonDemo2
- JacksonDemo3
- JacksonDemo4
- JacksonDemo5
- JacksonDemo6
- JacksonDemo7
- JacksonDemo8
- JacksonDemo9
- ProductApiTask1 [boot]
- ProductApiTask2 [boot]
- ProductApiTask3 [boot]
  - src/main/java
    - in.sp.main
    - in.sp.main.beans
      - Products.java
    - in.sp.main.controllers
      - ProductController.java
    - in.sp.main.services
  - src/main/resources
  - src/test/java
  - JRE System Library [JavaSE-17]
  - Maven Dependencies
  - src
    - main
      - java
      - resources
      - webapp
        - views
          - index.jsp
    - test
  - target
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
- SpringbootJacksonDemo1 [boot]

ProductController.java | index.jsp | ProductApiTask3/pom.xml | Products.java

```
22          <c:forEach var="productsList" items="${model_productsList}">
23              <tr>
24                  <td> ${productsList.getId()} </td>
25                  <td> ${productsList.getTitle()} </td>
26                  <td> ${productsList.getPrice()} </td>
27                  <td> <img src="${productsList.getThumbnail()}" alt="" width="100" /> </td>
28              </tr>
29          </c:forEach>
30      </table>
31
32      <c:if test="${model_currentPage > 1}">
33          <a href="/products?page=${model_currentPage-1}">Previous</a>
34      </c:if>
35
36      <c:forEach var="pageNo" begin="1" end="${model_totalPages}">
37          <a href="/products?page=${pageNo}">${pageNo}</a>
38      </c:forEach>
39
40      <c:if test="${model_currentPage < model_totalPages}">
41          <a href="/products?page=${model_currentPage+1}">Next</a>
42      </c:if>
43
44  </body>
45  </html>
```

Writable | Smart Insert | 43 : 1 : 1179

35°C हल्की बारिश ENG 09:16 PM 13-06-2023