11-06-2024

Keywords:

Keywords are the reserved words which has specific meaning in java programing language

Data Types::

byte(1 byte)

short(2 bytes)

int (4 bytes)

float (4bytes)

Long (8 bytes)

Double(8 bytes)

Char(2 bytes)

Boolean

Control Statements:

if

else

for

while

do

switch

case

break

continue

Default

Access Modifiers:

Public

Private

Protected

OOP’s::

.........

Class

New

Interface

Extends

Implements

This

Super

Abstract

InstanceOf

Package

Import

Enum

Modifiers::

.................

Static

Final

Synchronized

Transient

Strictfp

Void

Return

Native

assert

Const

Goto

Volatile

Exception Handling:

.......................

Try::

Catch

Finally

Throw

Throws

True

False

Null

String str=null

what is the default value of String (for any Objects default is null)

Note: All Keywords should start with small case letters

Identifiers:

....................

The name which we use to refer uniquely that is an identifier:

Class Test

{

Int a=10

Public void m1()

{

System.out.println(“Hello Innover”);

}

}

Rules for Identifiers:

....................

1)Always identifier should have letters(a-z or A -Z) and also digits(any numbers).

2)always identifier name must start with letter

3)and some symbols we can use as staring letter for identifier those are underscore(\_)and dollar($)

4) we cannot start any identifier with any special symbol and can’t start with digits.

5) we should not use keywords or reserved words as identifier.

Variables:

Class Test

{

int a; //instance variables

int b; //instance variable

static int c; //static variable

Public void m1(int a, int b)

{

Int d=20 //local Variable the scope is limited to m1() method

Int e=a+b+d;

Sop(e);

}

}

12/06/2024

..............

OOP’s(Object Oriented Programing)

Class:

class is blueprint or virtual entity or group of objects we can call as class

Object :

Instance of a class

Encapsulation :

Class Employee

{

private int empNo;

private String empName;

private double empSalary;

// getters and setter methods for the properties of class

Public int getEmpNo()

{

Return empNo;

}

Public void setEmpNo(int empNo)

{

Sop(empNo);  
}

}

Binding data members and members functions as single unit .we will achieve by declaring properties or data members of a class as private .

Abstraction::

Hiding the internal logic (the code what we write inside a method ) to the end user enable the services what can avail.

abstract Class A

{

public void m1() //instance method

{

Sop(“Hello Innover ”);

}

abstract public void m2() ; //abstract method

}

Class B extends A

{

public void m2()

{

Sop(“implementing abstract method in class B ”);

}

}

Note: By using abstract class we won’t able to achieve 100 % abstraction because sometimes we can also have instance methods .

All abstract methods from abstract will be implementing in it child class .

Key points about abstract class's:

1)we will achieve partial implementation by using abstract classes.

2)we can have constructor inside abstract classes

3) Abstract class will be having both abstract methods and instance methods.

4)it is not possible to create Object for abstract classes .

Inheritance:

Getting the properties and methods of super class into child class .

Class A

{

int a;

int b;

public void m1()

{

Sop(“class A m1 method ”);

}

Public void m2()

{

Sop(“class A m2 method ”);

}

}

Class B extends A

{

Public void m3()

{

Sop(“class B m3() method ”);

}  
}

B b=new B();

b.m3()// class B m3() method

b.m1()// class A m1() method

We have different types inheritance

Polymorphism:(Many forms )::

Method overloading

Class A

{

Public void m1(int a,int b)

{

Sop(a+b);

}

Public void m1(int a,int b,int c)

{

Sop(a+b);

}

}

Example 2:

Class A

{

Public void m1(int a,int b)

{

Sop(a+b);

}

Public void m1(int a,floa b)

{

Sop(a+b);

}

}

Example 3: order of arguments are different

Class A

{

Public void m1(int a,float b)

{

Sop(a+b);

}

Public void m1(float a,int b)

{

Sop(a+b);

}

}

Method Overriding :

13/06/2024

Exception Handling:

.............................

Class Test

{

Public static void main(String args[])

{

St1

St2

St3

St4

try

{

... sop(“Exception (mistake) or wrong code “)

}

Sop(“we should not keep any statement in bw try and catch”)

catch(NullPointerException ne)

{

Sop(“catch block is used for handling an exception”)

}

St8

Db connection

st9

st10

}

}

Is it possible to have only try block ?

It is not possible every try block must have catch or finally

Finally Block?

It is mandatory block it will be executing all the time if we get exception or not

The purpose of finally block is to do resource clean up

Ex : if any db connection those should terminate or if any files are opened those should close .

Try

{

}

Catch()

{

}

finally

{

}

Try

{

}

finally

{

}

Try

{

}

Catch(ArrayIndexOutOfBoundException ae)

{

}

Catch(IndexOutOfBoundException i)

{

}

Catch(RuntimeException re)

{

}

Throw::

1)we can create custom exception

2)it is followed with Object

3)we can throw only one exception at a time

4)it comes inside a method

Thow new NotElgiableToCastTheVote();

Throws:

It is followed with method signature

It will throw multiple exceptions at a time

Try

{

}

Catch( NullpointerExcepion ne | RuntimeExcetion re | Exception e)

Try

{

10k

Try

{

100

200

}

}

catch()

{

}

14/06/2024

.................

Multithreading:

Class MyThread extends Thread

{

Public void run()

{

For(int i=1;i<5;i++)

{

System.out.println(i);

}  
}  
}

Public class MainThread

{

Public static void main(String args[])

{

MyThread mt=new MyThread();

Thread t=new Thread(mt);

t.start(); //runnable state

}

}

Step1:Thread t=new Thread(); // new state or born state

Step2:t.start() //runnable sate

Step3:when thread is executing run() method thread will be in running state

Int a=10

Thread t1 is trying to update a=20

Sop(a)//20

Thread t2 is trying to update a=30

Sop(a)

Synchronized

Class Test

{

Public void m1()

{

sop(“hello Innover”);

Out this 1000 lines of code

Synchronized

{

200

300

}

}

{

}  
}

Sop(a)//

Synchronization

Deadlock?

............

Executor Framework

.............................

If want to create Thread

Thread t=new Thread();

100 Threads

Thread t1=new Thread();

Thread t2=new Thread();

.....

Thread t100=new Thread();

FixedNumberOfThread Pools100 1000

SingleThreadPool

CachedThreadPool

SchedulerThreadPool

17-06/2024

...............

Java full stack developer

Front end ::

....................

HTML

CSS

JAVA Script

Angular js

React js

Middle ware ::

Java

JDBC

Servlets

JSP

Spring framework

Spring boot

Microservices

Backed systems

RDBS::SQL

Oracle

MySQL

Db2

Non relational(NO SQL)

......

Mango DB

Spring Framework::

................................

Spring IOC

Spring MVC

Spring AOP

Spring DAO

Class Employee

{

Private int empNo;

Private String empName;

Private EmpAddress empAddress;

Employee e=new Employee()  
}

Class EmpAddress;

{

Priavate String presentAddress;

Priavate String permanentAddress;

EmpAddress ed=new EmpAddress()

}

Spring IOC: Inversion of controller

HAS Relationship:

At runtime we have pass object .

.

When it comes spring framework as a programmer we won’t create any object .

The Object will create by spring containers that object we say as “bean “

We have 2 container to create an object

1.Bean Factory container.

Serialization ::

Interface Loan

{

Public abstract void intrestRates();

}

Class A implements Loan

{

Public void intrestRates()

{

Sop(“personal interests rates 10%”)

}

}

Class B implements Loan

{

Public void intrestRates()

{

Sop(“home interests' rates 8%”)

}

Serializable::

It is marker interface it won’t have any methods inside interface.

Class Employee implements serializable

{

}

2.Application Context Container.

@Controller

Class Employee

{

Int empNo;

String empName ;

Address add;

}

@Controller

Class Address

{

}

Bean lifecycle::

...............

Bean creation

Injecting the bean (the process of injecting the bean is called DI(dependency injection ) )

Destroy bean

Bean scopes :

Types of decency injections

18/06/2024

......................

Types of Dependency Injection.

........................

Constructor Based Injection

Setter method based injection

Field or Property based injection

@component is the combination of 3 annotations

1. @controller
2. @Service
3. @Repository

@component = @controller

+ @Service+ @Repository

@Component

Public class Employee

{

}

MVC ::

Model view controller

@Controller

Public class EmpController

{

}

@Service

Public class EmpService

{

}

@Repository

Public class EmpRepo

{

}

Constructor Based DI::

@Contoller

Public class Employee

{

Private int empno;

Private String empName;

Private Address address;

@Autowired

Public Employee(Address address)

{

this. address= address;

}

}

Setter Method Based DI:

Public class Employee

{

Private Address address;

@Autowired

Public void setAddress(Address address)

{

This.address=address;

}

}

Field or Property Based DI:

Public class Employee

{

@Autowired

Private Address address;

}

Bean Scopes::

Singleton scope

Prototype Scope

Application scope

Request scope

Session scope

Spring Boot:

...................

1)Enable autoconfiguration

2)Started dependencies

3)inbuild Embedded server

4)devtools

5)Actuator

6)Profiles

7)standalone application

8)convention over configuration

@Component

Public class Address

{

Private String presentAddress;

Private String permanentAddress;

}

@Autowired is an annotation in Spring Framework that enables dependency injection for Java classes. It allows Spring to automatically inject dependencies into the class, eliminating the need for manual configuration. This annotation can be used to inject dependencies into fields, methods, and constructors.

19/06/2024

.......

Spring Boot Features ::

......................

1.Convention over configuration

2.inbuilt server

3.in memory

4.enable auto configurations

5.starter dependencies

6.devtools

7.profiles

9.Actutator

10.application. properties files

11.to handle exception globally @ControllerAdvice

12.we can develop standalone application

Annotations::

WebApplicationContext container

1.@SpringBootApplication =@EnableAutoConfiguration+configuration+@componentScan

@SpringBootApplication

public class SpringWebAppllication

{

public static void main(String args[])

{

}

}

2.@RestController

It is same as like @controller annotation

But it is the combination of

@RestController =@Controller+@ResponseBody

Example:

@RestController

Class EmployeeController

{

}

@Autowired

@Component

@Service

@Controller

@Repository

@Entity

@Entity

@Table(name="emp\_table")

Public class Employee

{

@Id

@EmbededId

@GeneratedType(strategy=generationTyp.identity)

Private int empNo;

Private String empName;

Private String email;  
}

@OneToOne Annotation

@OneToMany Annotation

@ManyToMany Annotation

@ManyToOne Annotation

@Transational

@TransactionalManagement

@EnableWebSecurity

@EnableScheduling

@ControllerAdvice

@ExceptionHandler

CURD

Création(post Method)

Update(put method)

Retrive(get method)

Delete(delete method)

@GetMapping

@PostMapping

@PutMapping

@DeleteMapping

@RequestBody

@PathParam

@QueryParam

@RequestMapping

@ModelAttribute

@ConfigurationProperties

@Value

@Validate

@Bean

@Required

WEB SERVICES :

.......................

SOAP webservice :WSDL

RESTful Webservice:: HTTP

20/06/2024::

...................

JDBC (java data base connectivity)::

To connect over java application to db and perform different types CURD operations

1.Driver Interface

2.Connection interface

3.Statement interface or prepared statement or callable interface

4.ResultSet Interface

ORM Framework:

Object Relational mapping ::

Hibernate and JPA

Hibernate is opensource

Automatic table creation

Automatic primary key generation

High performance

Exceptions will be handled by Hibernate

HQL()

Java Object is having 3 states in hibernate

.Object before associating with session: Transient State

When Object is getting associate with session: Persistent state

If Object got removed from session: Detached state

1.Create Persistent class

2.Create the mapping file Persistent class

3.Create the Configuration file

4.Create the class which retrieves or persistent the object

5.Load the JAR file

6.Run the application

JPA(JAVA Persistent API)::

JPQL

Entity Manager

JPA Rules or Standards to perform different database operations

21/06/2024

How To Create REST API In Spring boot ::

......................................

GET method =@GetMapping (getting or retrieve resource)

POST Method =@PostMapping (Creating new Resource)

PUT Method=@PutMapping (Updating the resource)

DELETE Method =@DeleteMapping (Deleting the resource)

Step1:Employee Table

Empid empName email

101 deepak [deepak@innoverdigital.com](mailto:deepak@innoverdigital.com)

102 Anish [anish@innoverdigital.com](mailto:anish@innoverdigital.com)

Let's assume we have 30 records in DB table .

@table(name=”employee\_table”)

@Entity

Public class Employee

{

@Id

@GeneratedValue(strategy=GeneratedType.IDENTITY)Private Long empId;

Private String empName;

Private String email;

//Default Constructor

Public Employee()

{

}

Public Employee(Long empId,String empName,String email)

{

this.empId= empId;

This.empName=empName;

This.email=email;

}

//Getter and setter methods

public Long getEmpId()

{

return empId;

}

Public void setEmpId(Long empId)

{

This.empId=empId;

}

Public String empName()

{

Return empName;

}

Public void setEmpName(Stirng empName)

{

This.empName=empName;  
}

Public String getEmail()

{

Return email;

}

Public void setEmail(String email)

{

This.email=email;  
}

}

If we use data jpa as dependency

We have some interfaces which can use to perform db operations .

Curdepository

PagingAndSortingRepository

JPARepository

Create The Employee Repository

@Repository

Public interface EmployeeRepository extends JPARepository<Employee,Long>

{

}

Employee Service Class

@Service

Public class EmployeeService

{

@Autowired

Private EmployeeRepository employeeRepository;

Public Optional<Employee> getEmployeeById(Long empId)

{

return employeeRepository.findById(Long empId);

}

}

@RestController

@RequestMapping(“/api/employees”)

Public class EmployeeController

{

@Autowired

Private EmployeeService employeeService;

@GetMapping(“/{empId}”)

Public ResponseEntity<Employee> getEmployeeById(@PathVariable Long empId)

{

Optional<Employee> employee=EmployeeService.getEmployeeById(empId);

If(employee.isPresent())

{

return ResponseEntity.ok(employee.get());

}

esle

{

return ResponseEntity.notFound().build()

}

}

}

https://www.innoverdigital.com//api/employees/ {empId}

Run the application

@SpringBootApplication

public class EmployeeApplication

{

Public static void main(String args[])

{

SpringApplication.run(EmployeeApplication.class,args)

}

}

https://www.innoverdigital.com//api/employees/ 102

Application.properties

#Database configurations

Spring.datasource.url=jdbc:h2:mem:testbd

Driverclassname

Username

Pass

Dialet

Pom.xml

All the dependencies we keep in this file

24/06/2024

................

Creating REST API for POST Method by using spring boot.

The purpose of POST method is to create new resources .

We use Postmapping annotation to map the request.

Pom.xml

<dependencies>

<! ....Spring boot starter web dependency..>

<dependency>

<groupId>org.springFramework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!..spring boot starter data jpa dependecy..>

<dependency>

<groupId>org.springFramework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!..MY SQL databse ..>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springFramework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

Creating application configuration file

Application.properties

Spring.datasource.url=jdbc:mysql://loclahost::3306/databsename

Spring.dtasource.username=root

Spring.dtasoure.password=Innover@2024

Spring.jpa.show-sql=true

Spring.jpa.databse-platform=org.hibernate.dialect.MySQL5Dailet

Spte3: Creating entity class to map the db table

Import javax.persistence.Entity;

Import javax.persistence.Id;

Import javax.persistence.GenerationType;

Import javax.persistence.GeneratedValue;

@Entity

Public class Employee

{

@id

@GeneratedValue(strategy=GenerationTpe.IDENTITY)

Private Long empId;

Private String empName;

Private Double empSalary;

//getters and setter

}

@Repository

Public interfce EmployeeRepository extends JpaReposioty<Employee,Long>

{

}

@Service

Public class EmployeeService

{

@Autowired

Private EmployeeRepository empRepo;

Public Employee saveEmployee(Employee emp)

{

Return empRepo.save(emp);  
}  
}

Create Controller class to map the request and process the request

@RestController

@RequestMapping(“/api/employess”)

Public class EmployeeController

{

@Autowired

Private EmployeeService empService;

@PostMapping

Public ResponseEntity<Employee> createEmployee(@RequestBody Employee emp)

{

Employee saveEmp= empService. SaveEmployee(emp);

Return ResponseEntiy.ok(saveEmp);

}

}

<https://localhost:8080/api/employees>

@SpringBootApplication

public class EmployeeApplication

{

Public static void main(String args[])

{

SpringApplication.run(EmployeeApplication.class,args)

}

}

25/06/2024

Create REST API for PUT method by using spring boot.

The purpose of put method is to “update the resource”.

If suppose record is not available then it will create the record.

Here put method is having @PutMapping annotation to update the record in spring boot

Stpe1:

Create pom.xml file

Where keep all the dependencies for spring boot ,jpa, databse .

Step2:

Application.properties

To configure the database connection details

Step3:

Create Entity class :by using @Entity annotation purpose is to map the database table

Example :

Employee entity class

Step4:Create Repository by @Repository annotation the purpose is to perform database curd operations.

Example:

Employee Repository

Step5:Create Service class by using @service annotation .the purspose is to write business logic.

Example:

Employee Service class

Step6:Create Controller class By using RestController for mapping the request

Example :

Employee Controller

Pom.xml(Keep all required starter dependencies )

.....................

<Dependencies>

<!-- spring boot starter web dependency...>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa<artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-coonector-java<artifactId>

<dependency>

<! ..Spring boot starter Test dependency..>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test<artifactId>

</dependency>

</dependencies>

Step2:application.properties file

.

Spring.datasource.url=jdbc:mysql://localhost:3306/data base name

Spring.datasource.username=root

Spring.datasource.password=Innover@2024

Spring.jpa.show-sql=true

Spring.jpa.databse-platform=org.hibernate.dialect.MySQl5Dialect

Step3:

Import javax.persistence.Entity;

Import javax.persistence.GeneratedValu;

Import javax.persistence.GenerationType;

Import javax.persistence.Id;

@Entity

Public class Emolyee

{

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

Private Long id;

Private String name;

Private double salary;

//getters and setters

}

Import org.springframework.data.jpa.repository.Jparepository;

Import org.springframework.stereotype.Repository;

@Repository

Public interface EmployeeRepository extends JpaRepository<Employee,Long>

{

}

@Service

Public class EmployeeServicee

{

Private EmployeeRepository employeeRepo;

Public Employee updateEmployee(Long id,Employee employeeDetails){

Optional<Employee> employeeOptional= employeeRepo.findById(id);

If(employeeOptional.isPresnt())

{

Employee emp= employeeOptional.get();

Emp.SetName(employeeDetails.getName());

Emp.SetSalary(employeeDetails.getSalary());

Return employeeRepo.save(emp);

}

Else

{

Throw new RuntimeException(“EMployee not found ”)

}

}

@RestController

@RequestMapping(“/api/employees”)

Public class EmployeeController

{

@Autowired

Private EmployeeService employeeService;

@PutMapping(“{id}”)

Public ResponseEntity<Employee> updateEmployee(@PathVariable Long id,@RequestBody Employee employeeDetails)

{

Employee updateEmployee= employeeService.updateemployee(id,employeeDetails);

Retun ResponseEntity.ok(updateEmployee);

}

}

Http://loclashost:8080/api/employees/20

26/06/2027

Creating REST API for Delete Method by using spring boot:

Step1: pom.xml file : will keep all dependencies

Step2:application.properties configure the data base connection details

Stpe3:Define the entity class :to map database table

Step4:define repository ::to perform database operation

Step5:Define service class :to write business logic

Step6:define controller class:mapping the request

@RestController

@RequestMapping (“/api/employees”)

Public class EmployeeController

{

@Autowired

Private EmployeeService employeeService;

@DeleteMapping(“/{id}”)

Public ResponseEntity<String> deleteEmployee(@Pathvariable long id)

{

Try

{

EmployeeService.deleteEmployee(id);

Retrun ResponseEntity.ok(“Employee record deleted successfully”);

}

Catch(RuntimeException e)

{

Return ResponseEntity.status(404).body(e.getMessage());

}

}

}

@Servcie

Public class EmployeeService

{

@Autowired

Private EmployeeRepository empRepo;

Public void deleteEmployee(Long id)

{

If(EmpRepo.existsById(id))

{

EmpRepo.deleteById(id);

}

Else

{

Throw new RuntimeException(“employee record not found”);

}

}

}

1XX:: generic info error status code

2XX:this is success status codes

3xx:redirecting status codes

4xx:client side error codes

5XX: Server side error codes

Creating Rest API for PATCH Method by using Spring boot.

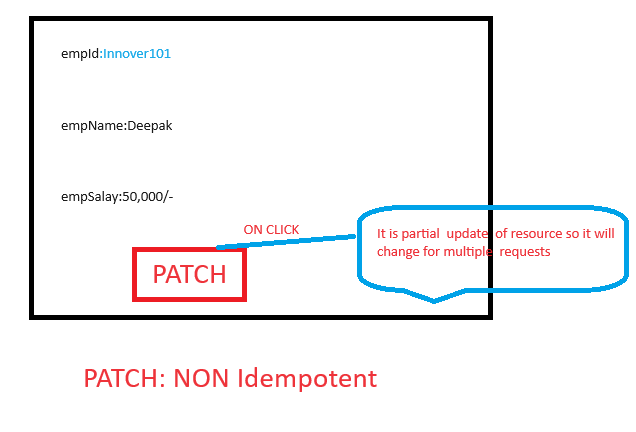
Put Method:

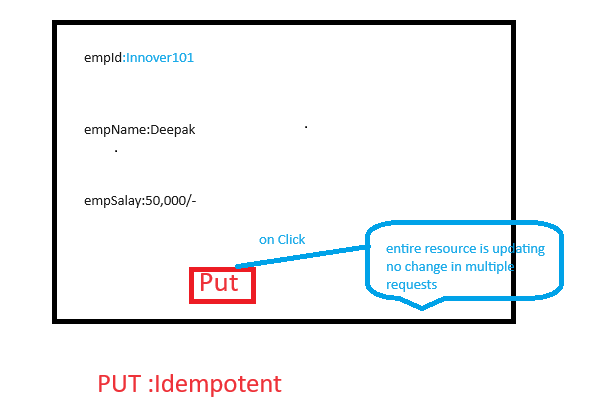
..............

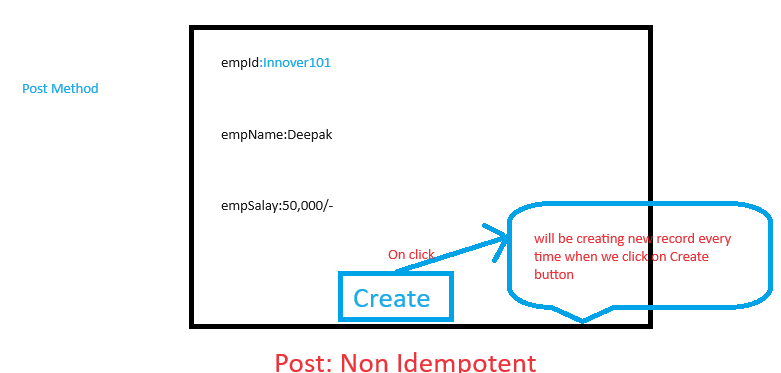
1)Put method modifies the entire resource.

2)put method client is requesting that the stored version be replaced .

3)put method is idempotent







HTTP MethodsGET, HEAD, PUT, DELETE, OPTIONS, and TRACE are idempotent methods, meaning they are safe to be retried or executed multiple times without causing unintended side effects. In contrast, POST and PATCH are generally considered non-idempotent, as their outcomes may vary with each request.

End TO End Application(REST API ) flow for PATCH METHOD:

Step1:pom.xml file contains dependecies for spring boot ,JPA and Database

Step2: application. Properties

Configure the database connection

Step3: Create entity class to Map the database table.

Step4: Create Repository to perform CURD or database operations

Step5: Create service class to keep or write business logic .

Step6: Create Controller class to map the User request.

@RestController

@ResuestMapping(“/api/employees”)

Public class EmployeeController

{

@Autowired

Private EmployeeService employeeService;

@PatchMapping(“/{id}”)

Public ResponseEntity<Employee> patchEmployee(@PathVariable Long Id,@RequestBody Employee employeeDetails)

{

Try

{

Employee updateEmployee= employeeService.patchEmployee(id, employeeDetails);

Return ResponseEntity.ok(employeeDetails);

}

Catch(RuntimeException e)

{

Return ResponseEntity.status(404).body(null);

}

}

}

@Service

Public class EmployeeService

{

@Autowired

Private EmployeeRepository empRepo;

Public Employee patchEmployee(Long id ,Employee employeeDetails)

{

Optional<Employee> optionalEmployee= empRepo.findById(id);

If(optionalEmployee.isPresent())

{

Employee employee= optionalEmployee.get();

If(employeeDetails.getName() !=null)

{

Employee.setName(employeeDetails.getName());

}

If(employeeDetails.getSalary() !=null)

{

Employee.setSalary(employeeDetails.getSalary());

}

Return emprepo.save(employee);

}

Else

{

Throw new RuntimeException(“Employee not Found ”);

}

}

}

28/06/2024

..............

Maven Build Tool:

......................

Project::

We will have to download all required libraries or jars from different websites.

Version: 3.0 -> 5.0

Maven is one of the build tool which will be used in the project .

What all things we can do by using maven:

Project structure::

C:\Users\INN949\Downloads\demo (1)\demo\src\main\java\com\example\demo

Pom.xml:: (project object model ) maven 2 used call as project.xml file from -> pom.xml 3

Package

Documentation

Reports

Pom.xml:::

...............

2 things in pom.xml

1)Dependencies ::: libraries

<Dependencies>

<!-- spring boot starter web dependency...>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa<artifactId>

<version>30</>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-coonector-java<artifactId>

<dependency>

<! ..Spring boot starter Test dependency..>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test<artifactId>

</dependency>

</dependencies>

Local repository::

Remote repository

<Project>::it is root tag in pom.xml file

<groupId>: it is sub element of project .

It specifies the id for the project group

<artifactId>:: it is sub element of project .

It specifies the id for the project.

<version>: it is the sub element of project. It specifies the version of the artifact under given group.

<scope>

<dependencies>

<dependency>

<packaging>

<name>

<url>

Maven build cycle::

............................

1.validate

2.complie

3.test

4.package

5.integration test

6.verify

7.Intall

8.deployee

2)Plugins: configurations

04/07/2024:

................

Step1:Create a Resposiotry on GitHub

1.Sign in to git hub

2.create new repository

Click on “+ “ icon int the upper right of the page and select “new Repository”.

<https://github.com/new>

3. a.Repository name

Ex: spring boot demo

b.Description (optional): add a description for repository.

c.public or private :

d. initialize the reposiotry:

It is optional you can README file

Click on Create repository button .

Step2: Set up Git on Local machine

a. intstall Git

Downalod and install git ::

<https://git-scm.com/downloads>

b.Configure git

Set up git username and email by ruuning the commands in terminal

Git config - -global user.name “uday”

Git config –global user.mail “uday@iinoverdigital.com”

Step3:Clone Repository URL

a.copy the repository URL.

Go to GitHub repository and click the “code” button to copy the URL

b.clone the reposiotry

Open terminal or command prompt

Git clone “git hub url”

Step4: create spring boot project

Step5: initialize and commit the spring boot project

a. initialize Git

Git init

b.add the project file to the staging area

Git add “file name”

Git add.

c. commit changes to local repository

Git commit –m “initial commit for the project”

Step6: Push the project to GItHub

1.add remote Reposiotry

Git remote add origin “git hub url”

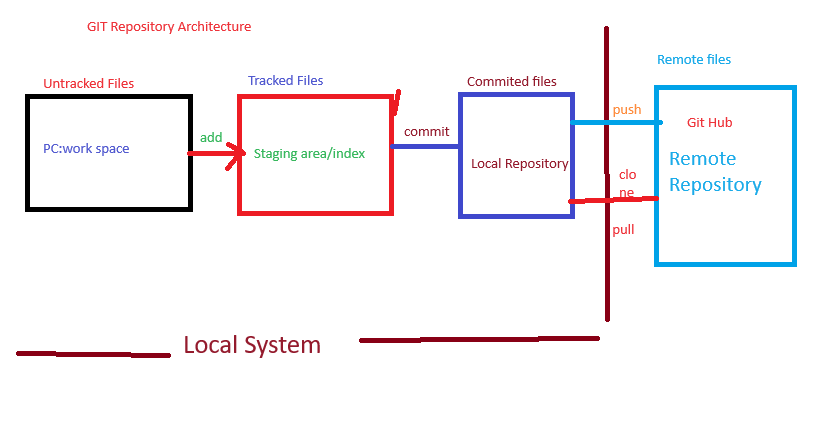
2.push the changes

Git push –u origin master

SCM (Source code management)

Control version system(CVS)

Git



INN949@INN-LT-04-H2H5T MINGW64 /c/my workspace (master) ::

git init:::To initialize git repository

git add java.txt test.txt

Or

git add .\*

git -a

git status ::

The status of the files

git config –global [user.mail="deepak@innoverdigital.com](mailto:user.mail="deepak@innoverdigital.com)' user.name="deepak"

git commit –m " commit messages"

git log