Path Variables

- It is used to extract values from the URL path.
- This is beneficial when we need to pass dynamic values in the URL and use them in the backend
- It is defined with the help of @PathVariable annotation.

When to use?

- If we want to pass dynamic values in the URL.

Ex. Fetch user by ID-> /users/{id}
Fetch the product by its name -> /products/{name}

Example:

Dependency -> web

```
package com.flynaut.pathVariableProject.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/users")
public class UserController {

    //Fetch user by ID
    @GetMapping("/{id}")
    public String getUserByID(@PathVariable int id){
        return "User with ID: "+id;
    }

    //Fetch user by name
```

```
    @GetMapping("/name/{username}")
    public String getUserByName(@PathVariable("username")
String name){
        return "User with name: "+name;
    }
}
```

Create a simple SpringBoot RestController which takes a number as a path variable and returns its square root.

```
package com.flynaut.pathVariableProject.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api")
public class SquareRootController {

    @GetMapping("/sqrt/{number}")
    public String getSquareRoot(@PathVariable double number){
        if (number < 0 ){
            return "Negative number is not supported!!";
        }

        double result = Math.sqrt(number);
        return "Square root of "+ number + " is: "+ result;
    }
}
```

StudentCRUD

Steps:

1. Create a SpringBoot Project with Spring initializer.

   Dependencies:

   - Spring Web
   - Spring Data JPA
   - MySQL Driver
   - Spring Dev Tools

2. Update Application.properties

```
spring.application.name=StudentProject

#Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/stude
nt_db
spring.datasource.username=springstudent
spring.datasource.password=springstudent

#Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.d
ialect.MySQLDialect

#HQL configuration
spring.jpa.format-sql = true
```

3. Create an entity package and inside it create a Student class.

```java
package com.flynaut.StudentProject.entity;


import jakarta.persistence.*;

@Entity  //marking this class as JPA entity
@Table(name="students")  //specifies the table name in the database
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(nullable=false)
    private String name;

    @Column(unique = true, nullable=false)
    private String email;

    @Column(nullable = false)
    private int age;

    @Column
    private String city;

    //getters & setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }
```

```
    public void setEmail(String email) {
        this.email = email;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }
}
```

4. Create a repository package and create an interface StudentRepository which will extend JPARepository.

```
package com.flynaut.StudentProject.repository;

import com.flynaut.StudentProject.entity.Student;
import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository   //Marking this interface as Spring Data
Repository
public interface StudentRepository extends
JpaRepository<Student,Integer> {

}
```

@Repository

Marking this interface as Spring Data Repository

It indicates that this interface is responsible for data interaction

extends JpaRepository<Student,Integer>

This interface extends JpaRepository, which provides us the built in CRUD operations.

JpaRepository<Student,Integer>

Student-> The entity class this repository manages

Integer-> the data type of primary key(id)

Built in methods from JpaRepository

1. save(Student student) -> save a student
2. findById(Integer id) -> retrieves a student by id
3. findAll()
4. deleteById()
5. delete()
6. count()

5. Create a service package and create studentService interface

Optional (Java SE 21 & JDK 21)

```java
package com.flynaut.StudentProject.service;

import com.flynaut.StudentProject.entity.Student;

import java.util.List;
import java.util.Optional;

public interface StudentService {

    Student addStudent(Student student);

    List<Student> getAllStudents();

    Optional<Student> getStudentById(int id);

    Student updateStudent(int id, Student student);

    void deleteStudent(int id);
}
```

6. Create a class studentServiceImpl which will implement StudentService.

```java
package com.flynaut.StudentProject.service;

import com.flynaut.StudentProject.entity.Student;
import
com.flynaut.StudentProject.repository.StudentRepository;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class StudentServiceImpl implements StudentService{
    @Autowired
    private StudentRepository studentRepository;

    @Override
    public Student addStudent(Student student) {
        return studentRepository.save(student);
    }

    @Override
    public List<Student> getAllStudents() {
        return studentRepository.findAll();
    }
```

```
    @Override
    public Optional<Student> getStudentById(int id) {
        return studentRepository.findById(id);
    }

    @Override
    public Student updateStudent(int id, Student student) {
        Student existingStudent =
studentRepository.findById(id).orElse(null);
        if (existingStudent != null){
            existingStudent.setName(student.getName());
            existingStudent.setEmail(student.getEmail());
            existingStudent.setAge(student.getAge());
            existingStudent.setCity(student.getCity());
            return  studentRepository.save(existingStudent);
        }
        throw new RuntimeException("Student Not Found with
ID: "+ id);
    }

    @Override
    public void deleteStudent(int id) {
        studentRepository.deleteById(id);
    }
}
```

7. Create a controller package and create a StudentController Class.

```
package com.flynaut.StudentProject.controller;

import com.flynaut.StudentProject.entity.Student;
import com.flynaut.StudentProject.service.StudentService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/students")
public class StudentController {

    @Autowired
    private StudentService studentService;

    @PostMapping
```

```
    public Student addStudent(@RequestBody Student student){
        return studentService.addStudent(student);
    }

    //TASK: create method to add multiple students

    @GetMapping
    public List<Student> getAllStudents(){
        return studentService.getAllStudents();
    }

    @PutMapping("/{id}")
    public ResponseEntity<Student>
updateStudent(@PathVariable int id,@RequestBody Student
student){
        try {
            return
ResponseEntity.ok(studentService.updateStudent(id,student));
        } catch (RuntimeException e) {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<String>
deleteStudent(@PathVariable int id){
        studentService.deleteStudent(id);
        return ResponseEntity.ok("Student Deleted
Successfully!!");
    }

}
```