Week - 3

Spring Data JPA - Quick Example

Code :

application.properties :

```
spring.application.name=orm-learn
logging.level.org.springframework=info
logging.level.com.cognizant=debug
logging.level.org.hibernate.SQL=trace
logging.level.org.hibernate.type.descriptor.sql=trace
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-
25.25logger{25} %25M %4L %m%n
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=root
spring.datasource.password=Aathira@14
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

Country.java :

```
package com.cognizant.orm_learn.model;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
```

```java
@Table(name = "country")
public class Country {

    @Id
    @Column(name = "code")
    private String code;

    @Column(name = "name")
    private String name;

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getName() {
        return name;
    }

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

    @Override
    public String toString() {
        return "Country{code='" + code + "', name='" + name + "'}";
    }
}
```

CountryRepository.java :

```
package com.cognizant.orm_learn.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.cognizant.orm_learn.model.Country;

@Repository
public interface CountryRepository extends JpaRepository<Country, String> {
}
```

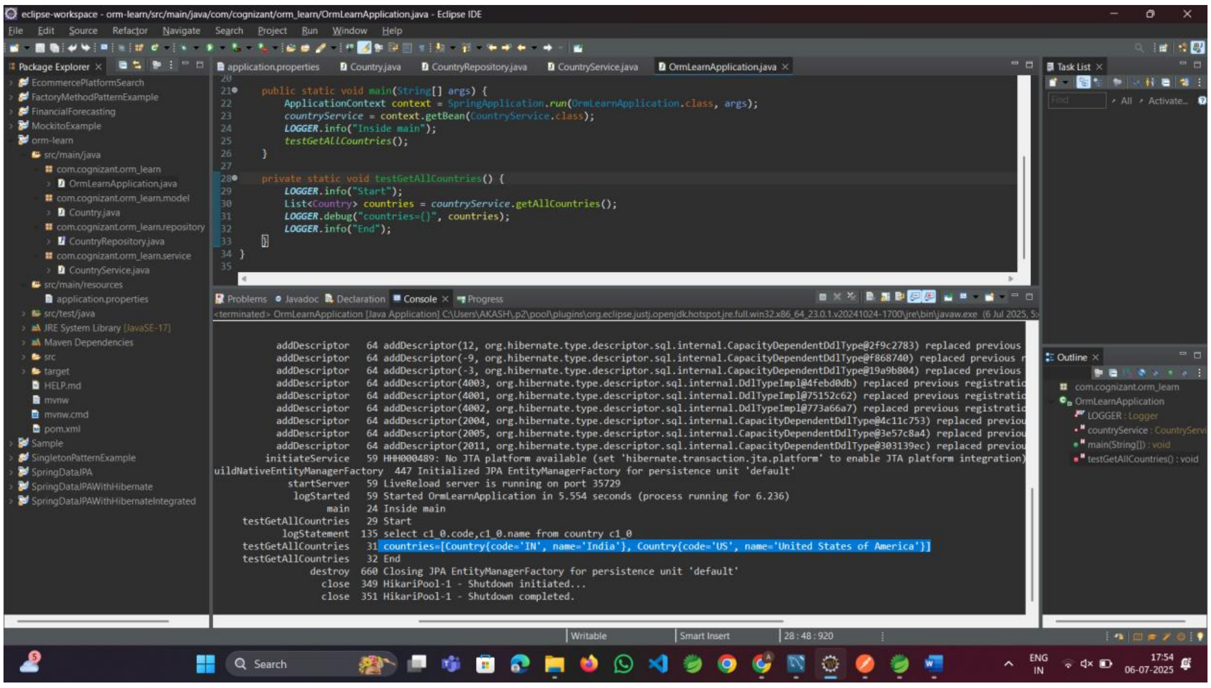
CountryService.java :

```
package com.cognizant.orm_learn.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.repository.CountryRepository;

import jakarta.transaction.Transactional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;


    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
}
```

OrmLearnApplication.java :

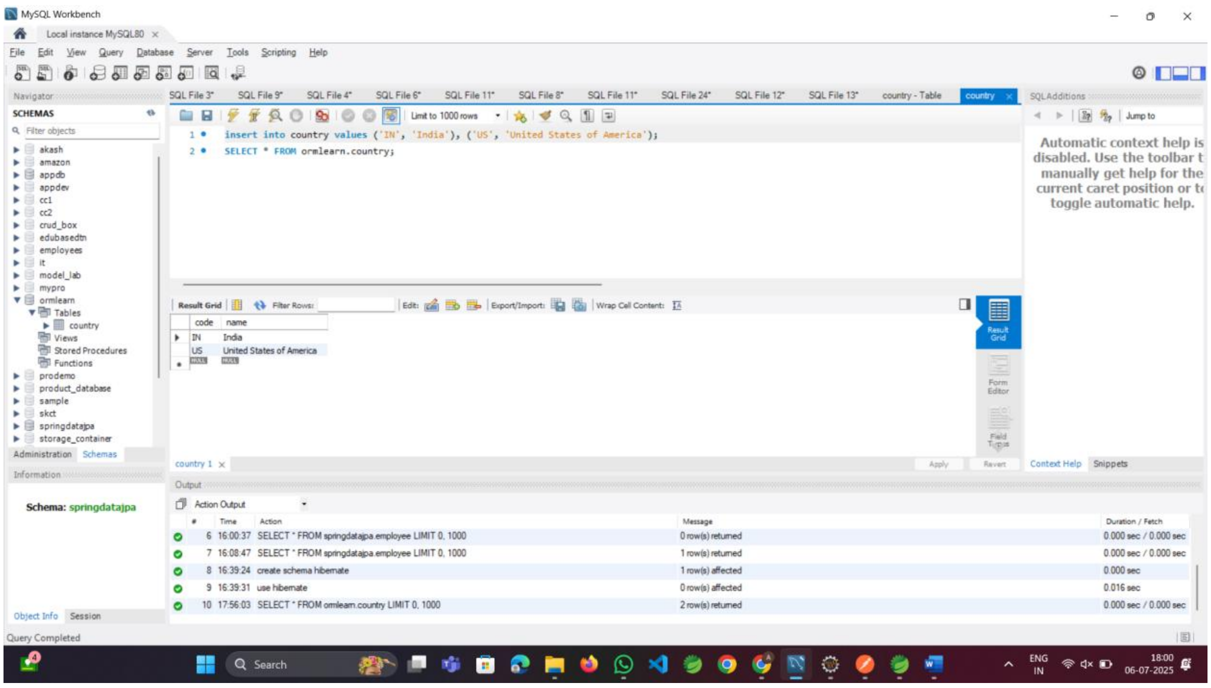
```
package com.cognizant.orm_learn;


import java.util.List;


import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;


import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.service.CountryService;


@SpringBootApplication
public class OrmLearnApplication {


    private static final Logger LOGGER =
LoggerFactory.getLogger(OrmLearnApplication.class);


    private static CountryService countryService;


    public static void main(String[] args) {
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
        countryService = context.getBean(CountryService.class);
        LOGGER.info("Inside main");
        testGetAllCountries();
    }

    private static void testGetAllCountries() {
        LOGGER.info("Start");
        List<Country> countries = countryService.getAllCountries();
        LOGGER.debug("countries={}", countries);
        LOGGER.info("End");
    }
}
```

Output :



```
    public static void main(String[] args) {
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);
        countryService = context.getBean(CountryService.class);
        LOGGER.info("Inside main");
        testGetAllCountries();
    }

    private static void testGetAllCountries() {
        LOGGER.info("Start");
        List<Country> countries = countryService.getAllCountries();
        LOGGER.debug("countries={}", countries);
        LOGGER.info("End");
    }
}
```

```
<terminated> OrmLearnApplication [Java Application] C:\Users\AKASH\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.1.v20241024-1700\jre\bin\javaw.exe (6 Jul 2025, 5:

         addDescriptor   64 addDescriptor(12, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@2f9c2783) replaced previous
         addDescriptor   64 addDescriptor(-9, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@f868740) replaced previous r
         addDescriptor   64 addDescriptor(-3, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@19a9b804) replaced previous
         addDescriptor   64 addDescriptor(4003, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@4febd0db) replaced previous registratio
         addDescriptor   64 addDescriptor(4001, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@75152c62) replaced previous registratio
         addDescriptor   64 addDescriptor(4002, org.hibernate.type.descriptor.sql.internal.DdlTypeImpl@773a66a7) replaced previous registratio
         addDescriptor   64 addDescriptor(2004, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@4c11c753) replaced previou
         addDescriptor   64 addDescriptor(2005, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@3e57c8a4) replaced previou
         addDescriptor   64 addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.CapacityDependentDdlType@303139ec) replaced previou
        initiateService   59 HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
uildNativeEntityManagerFactory  447 Initialized JPA EntityManagerFactory for persistence unit 'default'
           startServer   59 LiveReload server is running on port 35729
            logStarted   59 Started OrmLearnApplication in 5.554 seconds (process running for 6.236)
                  main   24 Inside main
   testGetAllCountries   29 Start
          logStatement  135 select c1_0.code,c1_0.name from country c1_0
   testGetAllCountries   31 countries=[Country{code='IN', name='India'}, Country{code='US', name='United States of America'}]
   testGetAllCountries   32 End
               destroy  660 Closing JPA EntityManagerFactory for persistence unit 'default'
                 close  349 HikariPool-1 - Shutdown initiated...
                 close  351 HikariPool-1 - Shutdown completed.
```



```sql
insert into country values ('IN', 'India'), ('US', 'United States of America');
SELECT * FROM ormlearn.country;
```

code	name
IN	India
US	United States of America
NULL	NULL

Schema: springdatajpa

```
#   Time      Action                                                Message            Duration / Fetch
6   16:00:37  SELECT * FROM springdatajpa.employee LIMIT 0, 1000    0 row(s) returned  0.000 sec / 0.000 sec
7   16:08:47  SELECT * FROM springdatajpa.employee LIMIT 0, 1000    1 row(s) returned  0.000 sec / 0.000 sec
8   16:39:24  create schema hibernate                               1 row(s) affected  0.000 sec
9   16:39:31  use hibernate                                         0 row(s) affected  0.016 sec
10  17:56:03  SELECT * FROM ormlearn.country LIMIT 0, 1000          2 row(s) returned  0.000 sec / 0.000 sec
```

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

Query Completed

Difference between JPA, Hibernate and Spring Data JPA

For Hibernate :

Code :

application.properties :

```
spring.application.name=hibernate
spring.datasource.url=jdbc:mysql://localhost:3306/hibernate
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=Aathira@14
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
debug=true
```

AssetVendor.java :

```
package com.hibernate.hibernate.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import jakarta.validation.constraints.Size;

@Entity
```

```
@Table(name = "assetvendor")
public class AssetVendor {

	@Id
	@GeneratedValue(strategy = GenerationType.UUID)
	@Column(name = "vendor_Id", nullable = false, updatable = false)
	private java.util.UUID id;

	@Column(name = "vendor_Name")
	private String vendorName;

	@Column(name = "vendor_Number")
	@Size(min = 10, max = 10)
	private String vendorNumber;

	@Column(name = "vendor_Brand")
	private String vendorBrand;

	@Column(name = "vendor_Address")
	private String vendorAddress;

	public AssetVendor() {
	}

	public java.util.UUID getId() {
		return id;
	}

	public void setId(java.util.UUID id) {
		this.id = id;
}

public String getVendorName() {
	return vendorName;
}

public void setVendorName(String vendorName) {
	this.vendorName = vendorName;
}

public String getVendorNumber() {
	return vendorNumber;
}

public void setVendorNumber(String vendorNumber) {
	this.vendorNumber = vendorNumber;
}

public String getVendorBrand() {
	return vendorBrand;
}

public void setVendorBrand(String vendorBrand) {
	this.vendorBrand = vendorBrand;
}

public String getVendorAddress() {
	return vendorAddress;
}
```

```
		public void setVendorAddress(String vendorAddress) {
			this.vendorAddress = vendorAddress;
		}
}
```

AssetVendorConfig.java :

```
package com.hibernate.hibernate.config;

import java.util.Properties;

import javax.sql.DataSource;

import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.*;
import org.springframework.core.env.Environment;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration
@EnableTransactionManagement
@PropertySource("classpath:application.properties")
public class AssetVendorConfig {

    @Autowired
    private Environment env;

    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource ds = new DriverManagerDataSource();
        ds.setDriverClassName(env.getRequiredProperty("spring.datasource.driver-class-
name"));
        ds.setUrl(env.getRequiredProperty("spring.datasource.url"));
        ds.setUsername(env.getRequiredProperty("spring.datasource.username"));
        ds.setPassword(env.getRequiredProperty("spring.datasource.password"));
        return ds;
    }

    @Bean
    public LocalSessionFactoryBean sessionFactory() {
        LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();
        factoryBean.setDataSource(dataSource());
        factoryBean.setPackagesToScan("com.hibernate.hibernate.entity");
        factoryBean.setHibernateProperties(hibernateProperties());
        return factoryBean;
    }

    private Properties hibernateProperties() {
        Properties props = new Properties();
        props.put("hibernate.dialect", "org.hibernate.dialect.PostgreSQLDialect");
        props.put("hibernate.hbm2ddl.auto", "update");
        props.put("hibernate.show_sql", "true");
        return props;
    }

    @Bean
    public HibernateTransactionManager transactionManager(SessionFactory sessionFactory)
{
        HibernateTransactionManager txManager = new HibernateTransactionManager();
        txManager.setSessionFactory(sessionFactory);
        return txManager;
    }
}
```

AssetVendorDAOImpl :

```
package com.hibernate.hibernate.dao.daoimpl;


import java.util.List;
import java.util.UUID;


import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;


import com.hibernate.hibernate.dao.AssetVendorDAO;
import com.hibernate.hibernate.entity.AssetVendor;


import jakarta.persistence.criteria.CriteriaBuilder;
import jakarta.persistence.criteria.CriteriaQuery;
import jakarta.persistence.criteria.Root;
import jakarta.transaction.Transactional;


@Repository
@Transactional
public class AssetVendorDAOImpl implements AssetVendorDAO {


        private static final Logger logger =
LoggerFactory.getLogger(AssetVendorDAO.class);
```

```
@Autowired
private SessionFactory sessionFactory;

public Session getSession() {
	return sessionFactory.getCurrentSession();
}

@Override
public AssetVendor saveVendor(AssetVendor assetVendor) {
	getSession().persist(assetVendor);
	return assetVendor;
}

@Override
public List<AssetVendor> getAllVendor(AssetVendor assetVendor) {
	Session session = getSession();
	CriteriaBuilder cb = session.getCriteriaBuilder();
	CriteriaQuery<AssetVendor> cq = cb.createQuery(AssetVendor.class);
	Root<AssetVendor> root = cq.from(AssetVendor.class);

	cq.orderBy(cb.asc(root.get("vendorName")));

	return session.createQuery(cq).getResultList();
}

@Override
public void deleteAssetVendor(AssetVendor assetVendor, UUID vendorId) {
	Session session = getSession();
	AssetVendor vendor = session.get(AssetVendor.class, vendorId);
```

```
          if (vendor != null) {
                    getSession().remove(vendor);
          }
}

@Override
public AssetVendor updateVendor(AssetVendor assetVendor, UUID vendorId) {
          Session session = getSession();
          AssetVendor updateVendor = session.get(AssetVendor.class, vendorId);

          if (updateVendor != null) {

                    if (assetVendor.getVendorName() != null) {

updateVendor.setVendorName(assetVendor.getVendorName());
                    }

                    if (assetVendor.getVendorNumber() != null) {

updateVendor.setVendorNumber(assetVendor.getVendorNumber());
                    }

                    if (assetVendor.getVendorBrand() != null) {

updateVendor.setVendorBrand(assetVendor.getVendorBrand());
                    }

                    if (assetVendor.getVendorAddress() != null) {

updateVendor.setVendorAddress(assetVendor.getVendorAddress());
                    }
```

```
				session.merge(updateVendor);
				logger.info("Vendor Updated Successfully");
			} else {
				logger.warn("Vendor is not found");
			}
			return updateVendor;
		}
}
```

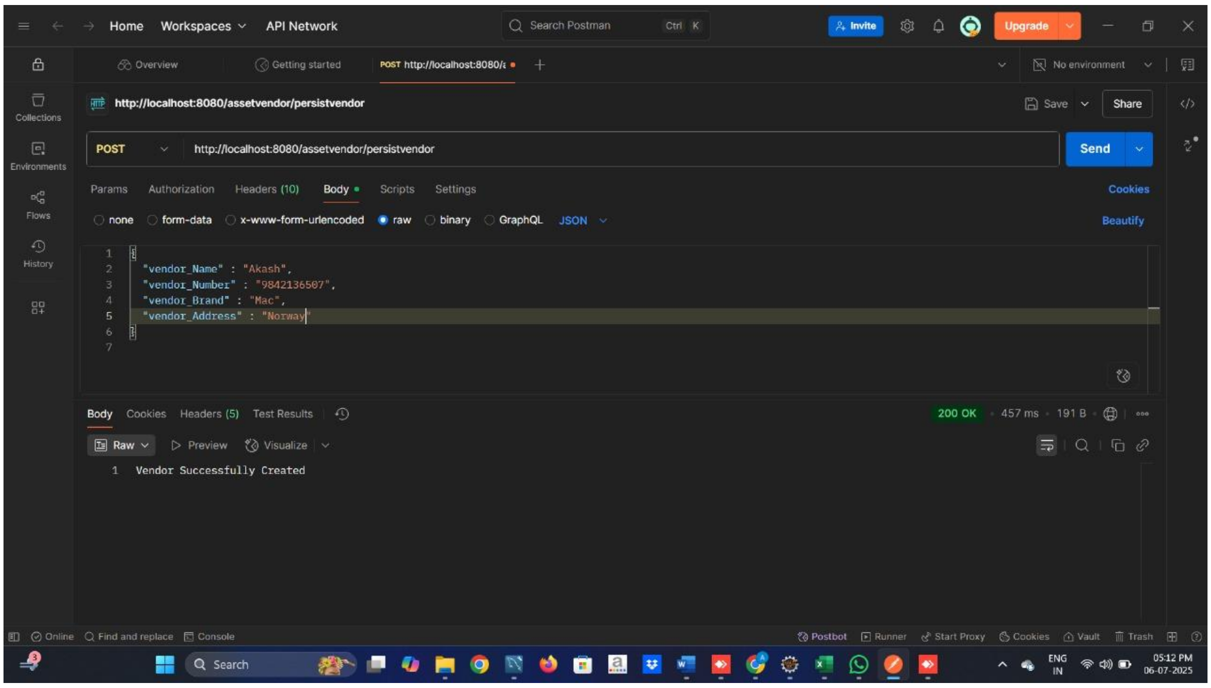
AssetVendorController :

```
package com.hibernate.hibernate.controller;

import java.util.List;
import java.util.UUID;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.hibernate.hibernate.dao.AssetVendorDAO;
import com.hibernate.hibernate.entity.AssetVendor;
import com.hibernate.hibernate.service.AssetVendorService;
```

```
@RestController
@RequestMapping("/assetvendor")
public class AssetVendorController {

	private static final Logger logger =
LoggerFactory.getLogger(AssetVendorController.class);

	@Autowired
	private AssetVendorService assetVendorService;

	@Autowired
	private AssetVendorDAO assetVendorDAO;

//	@PostMapping("/savevendor")
//	public String saveVendor(@Valid AssetVendor vendor) {
//		assetVendorService.saveVendor(vendor);
//		return "Vendor Saved Successfully";
//	}

	@PostMapping("/persistvendor")
	public String saveVendor(@RequestBody AssetVendor assetVendor) {
		AssetVendor assetVendorObj = null;
		AssetVendor saveVendor = convertToEntity(assetVendor);
		try {
			logger.debug("Created Vendor Details");
			assetVendorObj = assetVendorDAO.saveVendor(assetVendor);
			logger.debug("Created Vendor Successfully");
		} catch (Exception e) {
			logger.debug("Error while creating Vendor : " + e.getMessage());
		}
		return "Vendor Successfully Created";
	}

	private AssetVendor convertToEntity(AssetVendor assetVendor) {
		AssetVendor entity = new AssetVendor();
		entity.setVendorName(assetVendor.getVendorName());
		entity.setVendorBrand(assetVendor.getVendorBrand());
		entity.setVendorNumber(assetVendor.getVendorNumber());
		entity.setVendorAddress(assetVendor.getVendorAddress());
		return entity;
	}

	@GetMapping("/getallvendor")
	public List<AssetVendor> getAllVendor(AssetVendor assetVendor) {
		List<AssetVendor> assetVendorList = null;
		try {
			logger.debug("Successfully got the Vendors");
			assetVendorList = assetVendorDAO.getAllVendor(assetVendor);
		} catch (Exception e) {
			logger.debug("Error while getting All Vendor : " + e.getMessage());
		}
		return assetVendorList;
	}

	@DeleteMapping("/deletevendor/{vendorId}")
	public String deleteVendor(AssetVendor assetVendor, @PathVariable UUID
vendorId) {
		try {
			assetVendorDAO.deleteAssetVendor(assetVendor, vendorId);
			return "Vendor Deleted Successfully";
		} catch (Exception e) {
			logger.debug("Cannot find the Vendor : " + e.getMessage());
```

```
                        return "Cannot Delete the Vendor";
                }
        }

        @PutMapping("/updatevendor/{vendorId}")
        public AssetVendor updateVendor(@RequestBody AssetVendor assetVendor,
@PathVariable UUID vendorId) {
                try {
                        AssetVendor updatedVendor =
assetVendorDAO.updateVendor(assetVendor, vendorId);
                        return updatedVendor;
                } catch (Exception e) {
                        logger.debug("Cannot Update the Vendor");
                        return assetVendor;
                }
        }
}
```

Output :



http://localhost:8080/assetvendor/persistvendor

POST http://localhost:8080/assetvendor/persistvendor

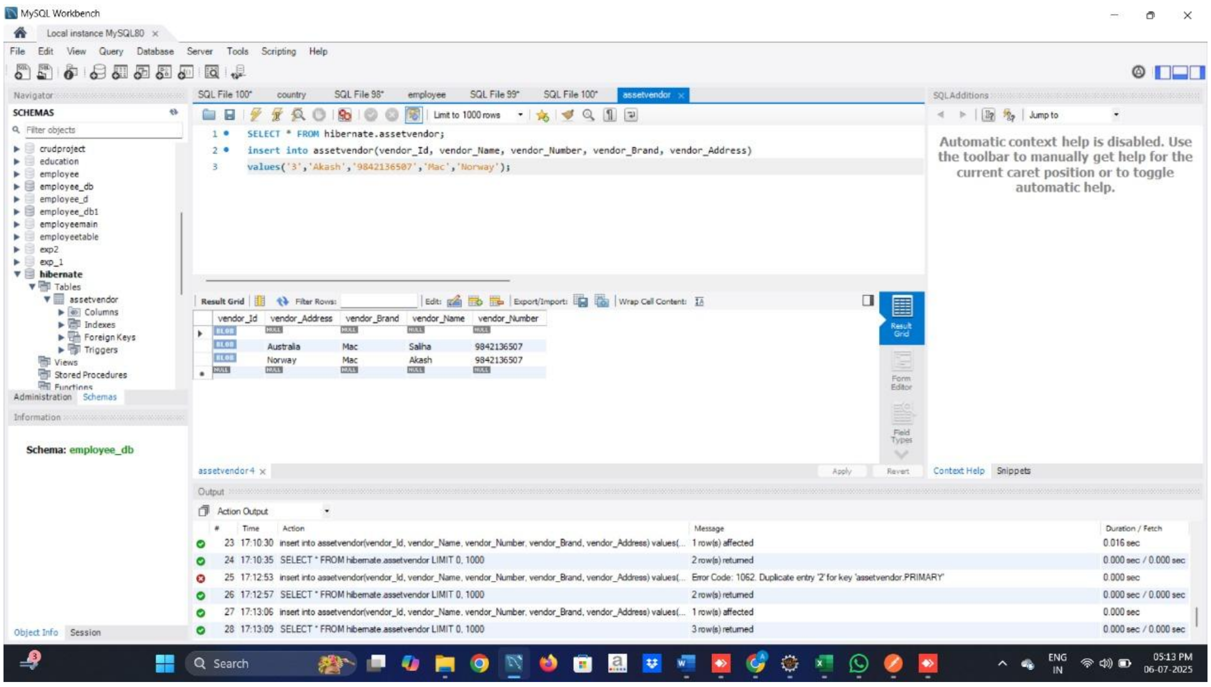
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
 "vendor_Name" : "Akash",
 "vendor_Number" : "9842136507",
 "vendor_Brand" : "Mac",
 "vendor_Address" : "Norway"
}
```

Body Cookies Headers (5) Test Results 200 OK 457 ms 191 B

Raw

```
Vendor Successfully Created
```



```
SELECT * FROM hibernate.assetvendor;
insert into assetvendor(vendor_Id, vendor_Name, vendor_Number, vendor_Brand, vendor_Address)
values('3','Akash','9842136507','Mac','Norway');
```

vendor_Id	vendor_Address	vendor_Brand	vendor_Name	vendor_Number
BLOB	NULL	NULL	NULL	NULL
BLOB	Australia	Mac	Saiha	9842136507
BLOB	Norway	Mac	Akash	9842136507
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                                                        Message                                                             Duration / Fetch
23  17:10:30  insert into assetvendor(vendor_Id, vendor_Name, vendor_Number, vendor_Brand, vendor_Address) values(...  1 row(s) affected                                                   0.016 sec
24  17:10:35  SELECT * FROM hibernate.assetvendor LIMIT 0, 1000                                             2 row(s) returned                                                   0.000 sec / 0.000 sec
25  17:12:53  insert into assetvendor(vendor_Id, vendor_Name, vendor_Number, vendor_Brand, vendor_Address) values(...  Error Code: 1062. Duplicate entry '2' for key 'assetvendor.PRIMARY'  0.000 sec
26  17:12:57  SELECT * FROM hibernate.assetvendor LIMIT 0, 1000                                             2 row(s) returned                                                   0.000 sec / 0.000 sec
27  17:13:06  insert into assetvendor(vendor_Id, vendor_Name, vendor_Number, vendor_Brand, vendor_Address) values(...  1 row(s) affected                                                   0.000 sec
28  17:13:09  SELECT * FROM hibernate.assetvendor LIMIT 0, 1000                                             3 row(s) returned                                                   0.000 sec / 0.000 sec
```

For Spring Data JPA :

Code :

application.properties :

```
spring.application.name=SpringDataJPA
spring.datasource.url=jdbc:mysql://localhost:3306/springdatajpa
spring.datasource.username=root
spring.datasource.password=Aathira@14
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

Employee.java :

```
package com.SpringDataJPA.SpringDataJPA.model;

import jakarta.persistence.*;

@Entity
@Table(name = "employee")
public class Employee {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Integer id;
      private String name;
  private String department;
  public Integer getId() {
            return id;
      }
      public void setId(Integer id) {
```

```
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getDepartment() {
		return department;
	}
	public void setDepartment(String department) {
		this.department = department;
	}
}
```

EmployeeRepository.java :

```
package com.SpringDataJPA.SpringDataJPA.repository;

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

import com.SpringDataJPA.SpringDataJPA.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer>{

}
```

EmployeeService.java :

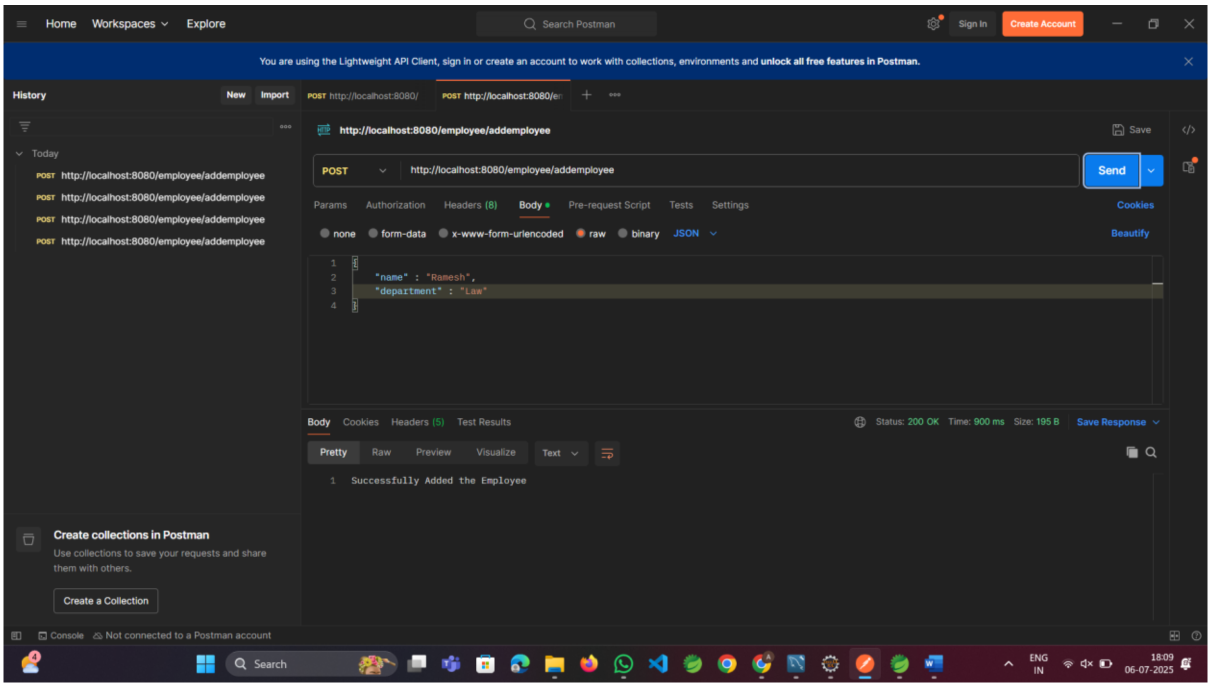
```
package com.SpringDataJPA.SpringDataJPA.service;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.SpringDataJPA.SpringDataJPA.model.Employee;
import com.SpringDataJPA.SpringDataJPA.repository.EmployeeRepository;

import jakarta.transaction.Transactional;

@Service
public class EmployeeService {

	@Autowired
	private EmployeeRepository employeeRepository;

	@Transactional
	public void addEmployee(Employee employee) {
		employeeRepository.save(employee);
	}

}
```

EmployeeController.java :

```
package com.SpringDataJPA.SpringDataJPA.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.SpringDataJPA.SpringDataJPA.model.Employee;
import com.SpringDataJPA.SpringDataJPA.service.EmployeeService;

@RestController
@RequestMapping("employee")
public class EmployeeController {

        @Autowired
        private EmployeeService employeeService;

        @PostMapping("/addemployee")
        public String addEmployee(@RequestBody Employee employee) {
                employeeService.addEmployee(employee);
                return "Successfully Added the Employee";
        }
}
```

Output :



You are using the Lightweight API Client, sign in or create an account to work with collections, environments and **unlock all free features in Postman.**

POST http://localhost:8080/employee/addemployee
POST http://localhost:8080/employee/addemployee
POST http://localhost:8080/employee/addemployee
POST http://localhost:8080/employee/addemployee

http://localhost:8080/employee/addemployee

POST http://localhost:8080/employee/addemployee

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | JSON

```
{
    "name" : "Ramesh",
    "department" : "Law"
}
```

Body | Cookies | Headers (5) | Test Results — Status: 200 OK Time: 900 ms Size: 195 B — Save Response

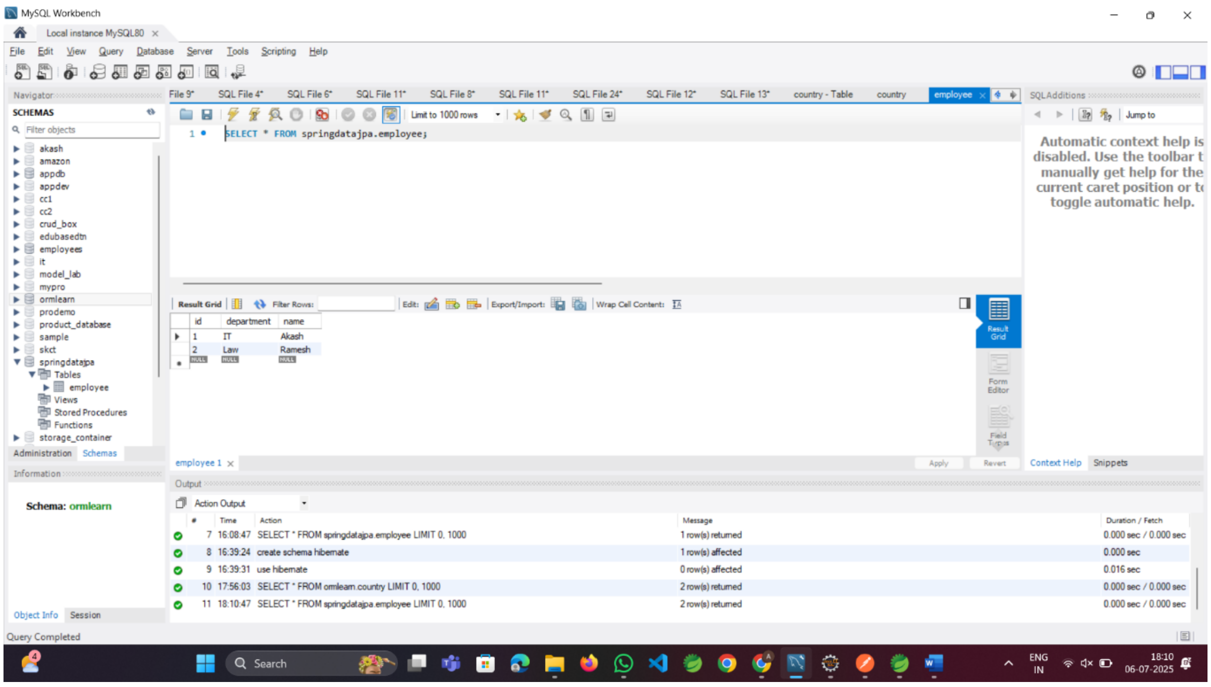
Pretty | Raw | Preview | Visualize | Text

```
Successfully Added the Employee
```

Create collections in Postman
Use collections to save your requests and share them with others.

Not connected to a Postman account

18:09
06-07-2025

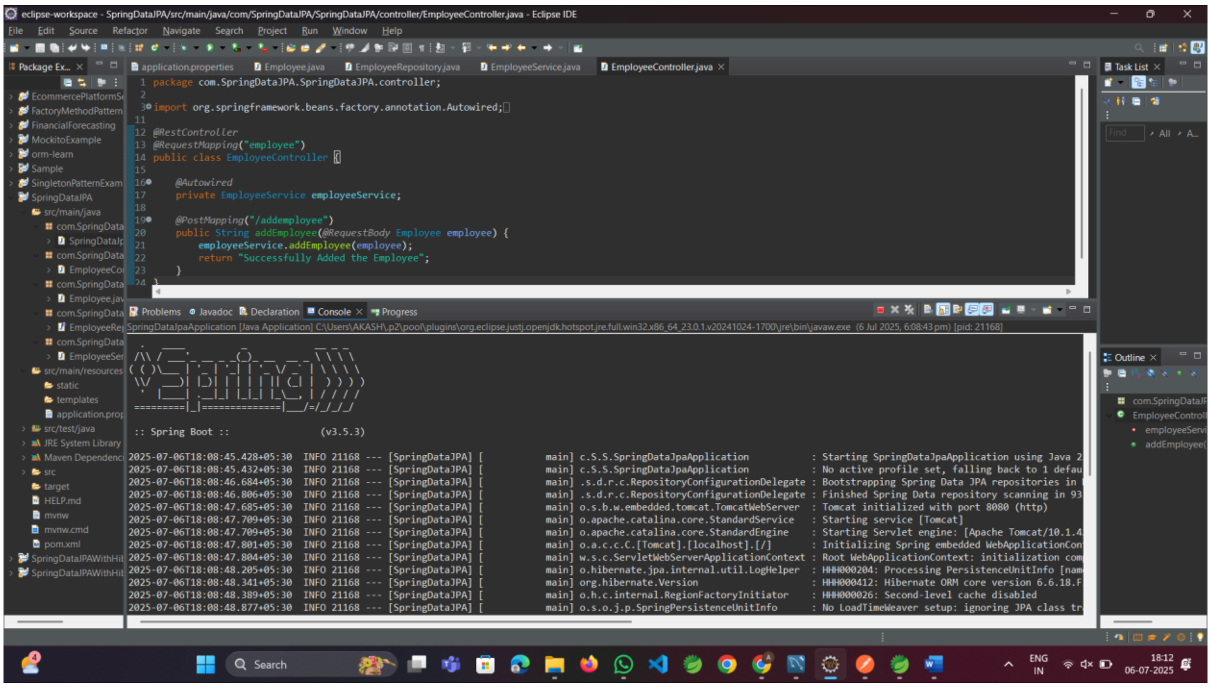


```
SELECT * FROM springdatajpa.employee;
```

id	department	name
1	IT	Akash
2	Law	Ramesh
NULL	NULL	NULL

#	Time	Action	Message	Duration / Fetch
7	16:08:47	SELECT * FROM springdatajpa.employee LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
8	16:39:24	create schema hibernate	1 row(s) affected	0.000 sec
9	16:39:31	use hibernate	0 row(s) affected	0.016 sec
10	17:56:03	SELECT * FROM ormlearn.country LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
11	18:10:47	SELECT * FROM springdatajpa.employee LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Query Completed

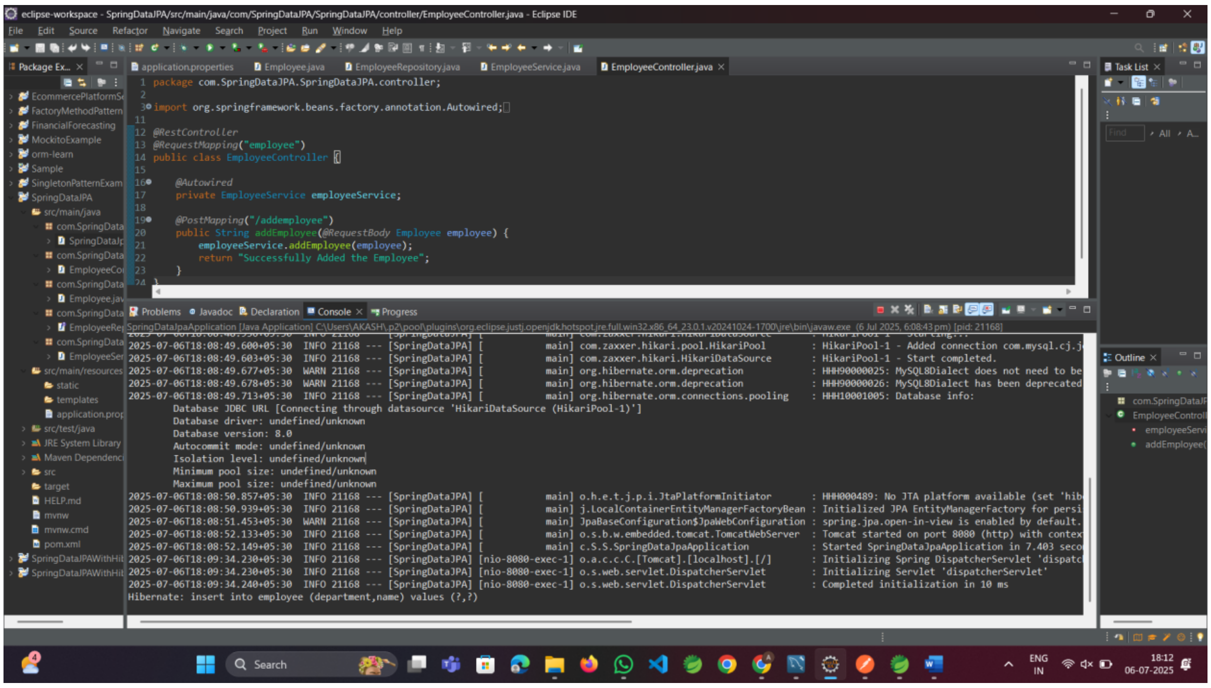


eclipse-workspace - SpringDataJPA/src/main/java/com/SpringDataJPA/SpringDataJPA/controller/EmployeeController.java - Eclipse IDE

```java
package com.SpringDataJPA.SpringDataJPA.controller;

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

@RestController
@RequestMapping("employee")
public class EmployeeController {

    @Autowired
    private EmployeeService employeeService;

    @PostMapping("/addemployee")
    public String addEmployee(@RequestBody Employee employee) {
        employeeService.addEmployee(employee);
        return "Successfully Added the Employee";
    }
}
```

SpringDataJpaApplication [Java Application] C:\Users\AKASH\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.1.v20241024-1700\jre\bin\javaw.exe (6 Jul 2025, 6:08:43 pm) [pid: 21168]

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.3)

2025-07-06T18:08:45.428+05:30  INFO 21168 --- [SpringDataJPA] [           main] c.S.S.SpringDataJpaApplication           : Starting SpringDataJpaApplication using Java 2
2025-07-06T18:08:45.432+05:30  INFO 21168 --- [SpringDataJPA] [           main] c.S.S.SpringDataJpaApplication           : No active profile set, falling back to 1 defau
2025-07-06T18:08:46.684+05:30  INFO 21168 --- [SpringDataJPA] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in
2025-07-06T18:08:46.806+05:30  INFO 21168 --- [SpringDataJPA] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 93
2025-07-06T18:08:47.685+05:30  INFO 21168 --- [SpringDataJPA] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2025-07-06T18:08:47.709+05:30  INFO 21168 --- [SpringDataJPA] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-06T18:08:47.709+05:30  INFO 21168 --- [SpringDataJPA] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.4
2025-07-06T18:08:47.801+05:30  INFO 21168 --- [SpringDataJPA] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationCon
2025-07-06T18:08:47.804+05:30  INFO 21168 --- [SpringDataJPA] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization com
2025-07-06T18:08:48.205+05:30  INFO 21168 --- [SpringDataJPA] [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [nam
2025-07-06T18:08:48.341+05:30  INFO 21168 --- [SpringDataJPA] [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.6.18.F
2025-07-06T18:08:48.389+05:30  INFO 21168 --- [SpringDataJPA] [           main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-07-06T18:08:48.877+05:30  INFO 21168 --- [SpringDataJPA] [           main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class tr
```


eclipse-workspace - SpringDataJPA/src/main/java/com/SpringDataJPA/SpringDataJPA/controller/EmployeeController.java - Eclipse IDE

```java
package com.SpringDataJPA.SpringDataJPA.controller;

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

@RestController
@RequestMapping("employee")
public class EmployeeController {

    @Autowired
    private EmployeeService employeeService;

    @PostMapping("/addemployee")
    public String addEmployee(@RequestBody Employee employee) {
        employeeService.addEmployee(employee);
        return "Successfully Added the Employee";
    }
}
```

SpringDataJpaApplication [Java Application] C:\Users\AKASH\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.1.v20241024-1700\jre\bin\javaw.exe (6 Jul 2025, 6:08:43 pm) [pid: 21168]

```
2025-07-06T18:08:49.600+05:30  INFO 21168 --- [SpringDataJPA] [           main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.j
2025-07-06T18:08:49.603+05:30  INFO 21168 --- [SpringDataJPA] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-07-06T18:08:49.677+05:30  WARN 21168 --- [SpringDataJPA] [           main] org.hibernate.orm.deprecation            : HHH90000025: MySQL8Dialect does not need to be
2025-07-06T18:08:49.678+05:30  WARN 21168 --- [SpringDataJPA] [           main] org.hibernate.orm.deprecation            : HHH90000026: MySQL8Dialect has been deprecated
2025-07-06T18:08:49.713+05:30  INFO 21168 --- [SpringDataJPA] [           main] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
	Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
	Database driver: undefined/unknown
	Database version: 8.0
	Autocommit mode: undefined/unknown
	Isolation level: undefined/unknown
	Minimum pool size: undefined/unknown
	Maximum pool size: undefined/unknown
2025-07-06T18:08:50.857+05:30  INFO 21168 --- [SpringDataJPA] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hib
2025-07-06T18:08:50.939+05:30  INFO 21168 --- [SpringDataJPA] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persi
2025-07-06T18:08:51.453+05:30  WARN 21168 --- [SpringDataJPA] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default.
2025-07-06T18:08:52.133+05:30  INFO 21168 --- [SpringDataJPA] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with contex
2025-07-06T18:08:52.149+05:30  INFO 21168 --- [SpringDataJPA] [           main] c.S.S.SpringDataJpaApplication           : Started SpringDataJpaApplication in 7.403 seco
2025-07-06T18:09:34.230+05:30  INFO 21168 --- [SpringDataJPA] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatc
2025-07-06T18:09:34.230+05:30  INFO 21168 --- [SpringDataJPA] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-07-06T18:09:34.240+05:30  INFO 21168 --- [SpringDataJPA] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 10 ms
Hibernate: insert into employee (department,name) values (?,?)
```