Spring Boot and RESTful Web Services

Q.1) Write a program to create a simple Spring Boot application that prints a message.

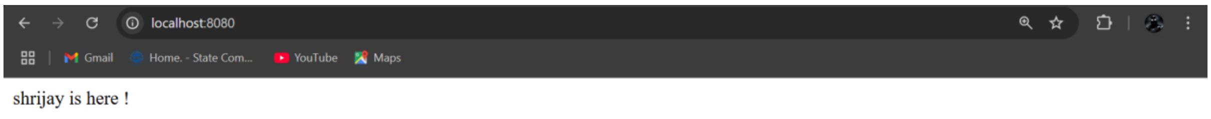
HelloController.java

```
package com.example.demo;


import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class HelloController {
	@RequestMapping("/")
	public String hello()
	{
	return "shrijay is here !";
	}
}
```

Tut1Application.java

```
package com.example.demo;


import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication
public class Tut1Application {


	public static void main(String[] args) {
		SpringApplication.run(Tut1Application.class, args);
	}
}
```

Output :



```
shrijay is here !
```

Q.2) Write a program to demonstrate RESTful Web Services with spring boot.

HelloWorldBean.java

```
package com.example.demo;
public class HelloWorldBean {
        public String message;
        public HelloWorldBean(String message)
        {
        this.message=message;
        }
        public String getMessage()
        {
        return message;
        }
        public void setMessage(String message)
        {
        this.message = message;
        }
        @Override
        public String toString()
        {
        return String.format ("HelloWorldBean [message=%s]", message);
        }
}
```

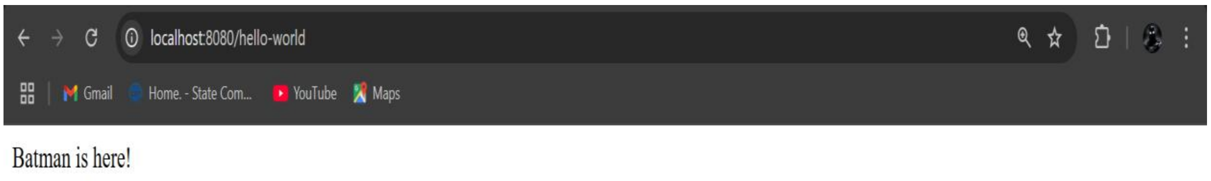
HelloWorldController.java

```
package com.example.demo;

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


@RestController
public class HelloWorldController {
	@GetMapping(path="/hello-world")
	public String helloWorld()
	{
	return "Batman is here!";
	}
	@GetMapping(path="/hello-world-bean")
	public HelloWorldBean helloWorldBean()
	{
	return new HelloWorldBean("How are you fella? xD");  }
}
```

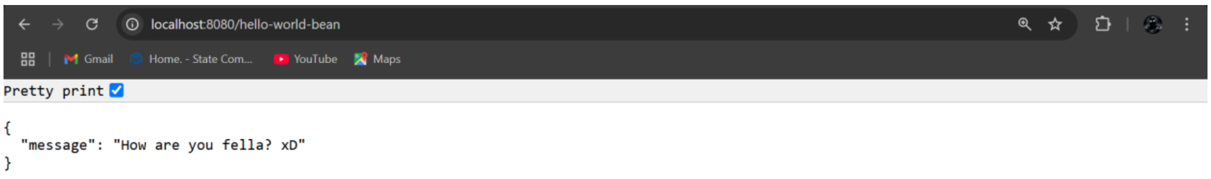
ShrijayApplication.java

```
package com.example.demo;


import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class ShrijayApplication {


	public static void main(String[] args) {
		SpringApplication.run(ShrijayApplication.class, args);
	}


}
```

Output :



Batman is here!



```
{
  "message": "How are you fella? xD"
}
```

Q.3) Database CRUD operations project using spring Boot.

UserController.java

```
package com.example.demo.controller;

import com.example.demo.entity.User;
import com.example.demo.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

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

  @Autowired
  private UserService userService;

  @GetMapping("/register")
  public String showRegistrationForm(Model model) {
    model.addAttribute("user", new User());
    return "register";
  }

  @PostMapping("/register")
  public String registerUser(@ModelAttribute User user) {
    userService.saveUser(user);
```

```java
        return "redirect:/users/" + user.getId();
    }

    @GetMapping("/{id}")
    public String getUserDetails(@PathVariable Long id, Model model) {
        Optional<User> user = userService.getUserById(id);
        user.ifPresent(value -> model.addAttribute("user", value));
        return "account";
    }

    @PostMapping("/{id}/update")
    public String updateUser(@PathVariable Long id, @ModelAttribute User updatedUser) {
        Optional<User> user = userService.getUserById(id);
        user.ifPresent(value -> {
            value.setUsername(updatedUser.getUsername());
            value.setPassword(updatedUser.getPassword());
            userService.updateUser(value);
        });
        return "redirect:/users/" + id;
    }

    @PostMapping("/{id}/delete")
    public String deleteUser(@PathVariable Long id) {
        userService.deleteUser(id);
        return "redirect:/users/register";
    }
}
```

User.java

```
package com.example.demo.entity;

import jakarta.persistence.*;

@Entity
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

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

    @Column(nullable = false)
    private String password;

    // Getters and Setters
    public Long getId() {
        return id;
    }

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

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

UserRepository.java

```
package com.example.demo.repository;

import com.example.demo.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

UserService.java

```
package com.example.demo.service;
```

```
}

import com.example.demo.entity.User;
import com.example.demo.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class UserService {

  @Autowired
  private UserRepository userRepository;

  public User saveUser(User user) {
    return userRepository.save(user);
  }

  public Optional<User> getUserById(Long id) {
    return userRepository.findById(id);
  }

  public void deleteUser(Long id) {
    userRepository.deleteById(id);
  }

  public User updateUser(User user) {
    return userRepository.save(user);
  }
```

```
}
```

resgister.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Register</title>
</head>
<body>
  <h1>Register</h1>
  <form th:action="@{/users/register}" th:object="${user}" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" th:field="*{username}"
required><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" th:field="*{password}"
required><br>
    <button type="submit">Register</button>
  </form>
</body>
</html>
```

account.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Account Details</title>
</head>
<body>
  <h1>Account Details</h1>
```

```
    <table>
        <tr>
            <th>ID</th>
            <td th:text="${user.id}"></td>
        </tr>
        <tr>
            <th>Username</th>
            <td th:text="${user.username}"></td>
        </tr>
        <tr>
            <th>Password</th>
            <td th:text="${user.password}"></td>
        </tr>
    </table>
    <h2>Update Details</h2>
    <form th:action="@{/users/{id}/update(id=${user.id})}" th:object="${user}"
method="post">
        <label for="username">New Username:</label>
        <input type="text" id="username" name="username" th:field="*{username}"
required><br>
        <label for="password">New Password:</label>
        <input type="password" id="password" name="password" th:field="*{password}"
required><br>
        <button type="submit">Update</button>
    </form>
    <form th:action="@{/users/{id}/delete(id=${user.id})}" method="post">
        <button type="submit">Delete Account</button>
    </form>
</body>
</html>
```

application.properties

```
spring.application.name=Login

spring.datasource.url=jdbc:postgresql://localhost:5432/postgres

spring.datasource.username=postgres

spring.datasource.password=root@7571

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true
```