Country Management Service Implementation
Implementing a complete service layer for managing Country entities using Spring Boot with Hibernate/JPA. This will include:

Entity class

Repository interface

Service layer

Implementation for finding by country code and adding new countries

1. Country Entity

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

import jakarta.persistence.*;

@Entity
@Table(name = "countries")
public class Country {

    @Id
    @Column(name = "country_code", length = 3)
    private String countryCode;

    @Column(name = "name", nullable = false)
    private String name;

    @Column(name = "population")
    private Integer population;

    @Column(name = "continent", length = 25)
    private String continent;

    // Constructors
    public Country() {}

    public Country(String countryCode, String name) {
        this.countryCode = countryCode;
        this.name = name;
    }

    // Getters and Setters
    public String getCountryCode() {
        return countryCode;
    }

    public void setCountryCode(String countryCode) {
        this.countryCode = countryCode;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
    public Integer getPopulation() {
        return population;
    }

    public void setPopulation(Integer population) {
        this.population = population;
    }

    public String getContinent() {
        return continent;
    }

    public void setContinent(String continent) {
        this.continent = continent;
    }

    @Override
    public String toString() {
        return "Country{" +
                "countryCode='" + countryCode + '\'' +
                ", name='" + name + '\'' +
                ", population=" + population +
                ", continent='" + continent + '\'' +
                '}';
    }
}
```

2. Country Repository

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

import com.example.demo.entity.Country;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

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

    // Custom query to find country by code
    Country findByCountryCode(String countryCode);

    // Check if country exists by code
    boolean existsByCountryCode(String countryCode);
}
```

3. Country Service Interface

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

import com.example.demo.entity.Country;
import java.util.List;

public interface CountryService {

    Country findCountryByCode(String countryCode);

    Country addNewCountry(Country country);
    List<Country> getAllCountries();

    // Additional methods can be added here
}
```

4. Country Service Implementation

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

import com.example.demo.entity.Country;
import com.example.demo.repository.CountryRepository;
import com.example.demo.service.CountryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
@Transactional
public class CountryServiceImpl implements CountryService {

    private final CountryRepository countryRepository;

    @Autowired
    public CountryServiceImpl(CountryRepository countryRepository) {
        this.countryRepository = countryRepository;
    }

    @Override
    public Country findCountryByCode(String countryCode) {
        return
countryRepository.findByCountryCode(countryCode.toUpperCase());
    }

    @Override
    public Country addNewCountry(Country country) {
        // Validate country code (3 letters)
        String countryCode = country.getCountryCode();
        if (countryCode == null || countryCode.length() != 3) {
            throw new IllegalArgumentException("Country code must be
exactly 3 characters");
        }

        // Convert to uppercase for consistency
        country.setCountryCode(countryCode.toUpperCase());

        // Check if country already exists
        if
(countryRepository.existsByCountryCode(country.getCountryCode())) {
            throw new IllegalStateException("Country with code " +
country.getCountryCode() + " already exists");
        }

        // Save the new country
        return countryRepository.save(country);
    }
    @Override
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
}
```

5. Country Controller (REST API)

```
package com.example.demo.controller;
import com.example.demo.entity.Country;
import com.example.demo.service.CountryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/countries")
public class CountryController {

    private final CountryService countryService;

    @Autowired
    public CountryController(CountryService countryService) {
        this.countryService = countryService;
    }

    @GetMapping("/{countryCode}")
    public ResponseEntity<Country> getCountryByCode(@PathVariable String
countryCode) {
        Country country = countryService.findCountryByCode(countryCode);
        if (country == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(country);
    }

    @PostMapping
    public ResponseEntity<Country> addNewCountry(@RequestBody Country
country) {
        Country newCountry = countryService.addNewCountry(country);
        return
ResponseEntity.status(HttpStatus.CREATED).body(newCountry);
    }

    @GetMapping
    public ResponseEntity<List<Country>> getAllCountries() {
        List<Country> countries = countryService.getAllCountries();
        return ResponseEntity.ok(countries);
    }
}
```

6. Exception Handling (Optional)

```
package com.example.demo.exception;
```

```
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(IllegalArgumentException.class)
    public ResponseEntity<String>
handleIllegalArgumentException(IllegalArgumentException ex) {
        return ResponseEntity.badRequest().body(ex.getMessage());
    }

    @ExceptionHandler(IllegalStateException.class)
    public ResponseEntity<String>
handleIllegalStateException(IllegalStateException ex) {
        return
ResponseEntity.status(HttpStatus.CONFLICT).body(ex.getMessage());
    }
}
```

Usage Examples:

1. Find a country by country code

Request:
```
GET /api/countries/USA
```

Response:
```
{
    "countryCode": "USA",
    "name": "United States of America",
    "population": 331000000,
    "continent": "North America"
}
```

2. Add a new country

Request:
```
POST /api/countries
Content-Type: application/json

{
    "countryCode": "CAN",
    "name": "Canada",
    "population": 38000000,
    "continent": "North America"
}
```

Response (201 Created):

json
```
{
    "countryCode": "CAN",
    "name": "Canada",
    "population": 38000000,
    "continent": "North America"
}
```

This implementation provides a complete solution for managing countries with:

Proper entity mapping

Repository with custom queries

Service layer with business logic

REST API endpoints

Basic validation and error handling