# SPRING REST USING SPRING BOOT

HTTP Request Response

What is HTTP Request and Response?

HTTP Request is the message sent by the client (usually a browser) to the server to ask for a resource.

It contains:

 Request line:

 Method: The action to perform (e.g., GET, POST)

 Resource: The URI/path being requested (e.g., /hello.txt)

 HTTP version: Protocol version used (e.g., HTTP/1.1)

 Headers: Additional info about the client or request (e.g., User-Agent, Host, Accept- Language)

 Body: Optional, mainly in POST or PUT requests (not in GET usually)

HTTP Response is the server’s reply to the client’s request. It contains:

 Status line:

 HTTP version (e.g., HTTP/1.1)

 Status code (e.g., 200 means OK/success)

 Reason phrase/message (e.g., OK)

 Headers: Metadata about the response (e.g., Date, Server type, Content-Type, Content- Length)

 Body: The actual content requested (e.g., HTML, JSON, image bytes)

Example from your text:

Request:

makefile

Copy code

GET /hello.txt HTTP/1.1

User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3 Host: [www.example.com](http://www.example.com/)

Accept-Language: en, mi

 Method = GET

 Resource = /hello.txt

 HTTP Version = HTTP/1.1

 User-Agent = identifies client software  Host = the server’s domain

Response:

yaml

Copy code

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT ETag: "34aa387-d-1568eb00"

Accept-Ranges: bytes Content-Length: 51 Vary: Accept-Encoding

Content-Type: text/plain

Hello World! My payload includes a trailing CRLF.  Status = 200 OK (success)

 Content-Type = text/plain (plain text data)

 Body = "Hello World! My payload includes a trailing CRLF."

How to view HTTP Request/Response in browser:

 Open Chrome DevTools (F12)  Go to Network tab

 Reload page or perform action

 Click on any request to see:

 General info (method, URL, status)  Request headers (sent by browser)  Response headers (sent by server)  Response body/content

Hello World RESTful Web Service

pom.xml

<project [xmlns="http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0) ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>1.0.0</version>

<parent>

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

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

<version>3.0.6</version> <!-- Use any compatible Spring Boot version -->

<relativePath/> <!-- lookup parent from repository -->

</parent>

<dependencies>

<!-- Spring Boot Web Starter -->

<dependency>

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

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

</dependency>

<!-- SLF4J Logger -->

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot Maven Plugin -->

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

# SpringLearnApplication.java

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

HelloController.java

package com.cognizant.spring\_learn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

@RestController

public class HelloController {

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

@GetMapping("/hello") public String sayHello() {

LOGGER.info("Start: sayHello()");

String message = "Hello World!!"; LOGGER.info("End: sayHello()"); return message;

}

}

# output

### Hello World!!

REST - Country Web Service

country.xml

<?xml version="1.0" encoding="UTF-8"?>

<beans [xmlns="http://www.springframework.org/schema/beans"](http://www.springframework.org/schema/beans) [xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance)

xsi:schemaLocation="

<http://www.springframework.org/schema/beans>

[http://www.springframework.org/schema/beans/spring-beans.xsd">](http://www.springframework.org/schema/beans/spring-beans.xsd)

<bean id="india" class="com.cognizant.spring\_learn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

</beans>

Country.java

package com.cognizant.spring\_learn.model;

public class Country {

private String code; private String name;

public Country() {}

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

this.name = 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;

}

}

# CountryController.java

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.model.Country; import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext; import org.springframework.web.bind.annotation.RequestMapping;

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

@RestController

public class CountryController {

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

@RequestMapping("/country") public Country getCountryIndia() {

LOGGER.info("Start: getCountryIndia()");

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml"); Country india = context.getBean("india", Country.class);

LOGGER.info("End: getCountryIndia()");

return india;

}

}

# SpringLearnApplication.java

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

## output

{

"code": "IN",

"name": "India"

}

REST - Get all countries

Country.java

package com.cognizant.springlearn.model;

public class Country { private String code; private String name;

public Country() {}

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

this.name = 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 + "]";

}

}

# country.xml

<?xml version="1.0" encoding="UTF-8"?>

<beans [xmlns="http://www.springframework.org/schema/beans"](http://www.springframework.org/schema/beans) [xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance)

xsi:schemaLoca[tion="http://www.springframework.org/schema/beans](http://www.springframework.org/schema/beans)

[http://www.springframework.org/schema/beans/spring-beans.xsd">](http://www.springframework.org/schema/beans/spring-beans.xsd)

<bean id="in" class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<bean id="us" class="com.cognizant.springlearn.model.Country">

<property name="code" value="US"/>

<property name="name" value="United States"/>

</bean>

<bean id="jp" class="com.cognizant.springlearn.model.Country">

<property name="code" value="JP"/>

<property name="name" value="Japan"/>

</bean>

<bean id="de" class="com.cognizant.springlearn.model.Country">

<property name="code" value="DE"/>

<property name="name" value="Germany"/>

</bean>

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="in"/>

<ref bean="us"/>

<ref bean="jp"/>

<ref bean="de"/>

</list>

</constructor-arg>

</bean>

</beans>

# CountryController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

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

import org.springframework.context.support.ClassPathXmlApplicationContext; import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController; import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.List;

@RestController

public class CountryController {

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

@GetMapping("/countries")

public List<Country> getAllCountries() { LOGGER.info("START: getAllCountries");

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml"); List<Country> countries = (List<Country>) context.getBean("countryList");

LOGGER.info("END: getAllCountries"); return countries;

}

}

# output

[

{ "code": "IN", "name": "India" },

{ "code": "US", "name": "United States" },

{ "code": "JP", "name": "Japan" },

{ "code": "DE", "name": "Germany" }

]

REST - Get country based on country code

### Country.java

package com.cognizant.springlearn.model;

public class Country { private String code; private String name;

public Country() {}

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

this.name = 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;

}

}

# CountryService.java

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.exception.CountryNotFoundException; import com.cognizant.springlearn.model.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext; import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) throws CountryNotFoundException {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml"); List<Country> countries = (List<Country>) context.getBean("countryList");

return countries.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElseThrow(() -> new CountryNotFoundException("Country not found: " + code));

}

}

# CountryController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.exception.CountryNotFoundException; import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*;

@RestController

public class CountryController {

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

@Autowired

private CountryService countryService;

@GetMapping("/countries/{code}")

public Country getCountry(@PathVariable String code) throws CountryNotFoundException { LOGGER.info("START: getCountry()");

Country country = countryService.getCountry(code); LOGGER.info("END: getCountry()");

return country;

}

}

# output

{

"code": "IN",

"name": "India"

}

REST - Get country exceptional scenario

### CountryNotFoundException.java

package com.cognizant.springlearn.service.exception;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.NOT\_FOUND, reason = "Country not found") public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

CountryService.java

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country; import

com.cognizant.springlearn.service.exception.CountryNotFoundExceptio n;

import org.springframework.context.ApplicationContext; import

org.springframework.context.support.ClassPathXmlApplicationContext; import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) throws CountryNotFoundException {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

List<Country> countries = (List<Country>) context.getBean("countryList");

return countries.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElseThrow(() -> new CountryNotFoundException("Country not found"));

}

}

# CountryController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*;

@RestController

public class CountryController {

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

@Autowired

private CountryService countryService;

@GetMapping("/countries/{code}")

public Country getCountry(@PathVariable String code) throws CountryNotFoundException { LOGGER.info("START: getCountry()");

Country country = countryService.getCountry(code); LOGGER.info("END: getCountry()");

return country;

}

}

# output

{

"timestamp": "2025-07-12T13:14:22.519+0000",

"status": 404,

"error": "Not Found",

"message": "Country not found",

"path": "/countries/az"

}

MockMVC - Test get country service

SpringLearnApplicationTests.java

package com.cognizant.springlearn;

import com.cognizant.springlearn.controller.CountryController;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.boot.test.context.SpringBootTest; import org.springframework.test.web.servlet.MockMvc;

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import static org.junit.jupiter.api.Assertions.assertNotNull;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get; import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

import org.springframework.test.web.servlet.ResultActions;

@SpringBootTest

@AutoConfigureMockMvc

public class SpringLearnApplicationTests {

@Autowired

private CountryController countryController;

@Autowired

private MockMvc mvc;

// ✅ 1. Check if CountryController bean is loaded properly @Test

public void contextLoads() {

assertNotNull(countryController);

}

// ✅ 2. Test the /country endpoint for correct JSON response @Test

public void testGetCountry() throws Exception {

ResultActions actions = mvc.perform(get("/country"));

actions.andExpect(status().isOk()) // HTTP 200

.andExpect(jsonPath("$.code").exists())

.andExpect(jsonPath("$.code").value("IN"))

.andExpect(jsonPath("$.name").exists())

.andExpect(jsonPath("$.name").value("India"));

}

}

# country.xml

<beans [xmlns="http://www.springframework.org/schema/beans"](http://www.springframework.org/schema/beans)

xmlns:conte[xt="http://www.springframework.org/schema/context](http://www.springframework.org/schema/context)" [xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance)

xsi:schemaLoca[tion="http://www.springframework.org/schema/beans](http://www.springframework.org/schema/beans)

[http://www.springframework.org/schema/beans/spring-beans.xsd">](http://www.springframework.org/schema/beans/spring-beans.xsd)

<bean id="in" class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="in"/>

<!-- Add more countries if needed -->

</list>

</constructor-arg>

</bean>

</beans>

# Country.java

package com.cognizant.springlearn.model;

public class Country { private String code; private String name;

// Getters and setters

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; }

}

CountryController.java

@GetMapping("/country")

public Country getCountryIndia() {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

return (Country) context.getBean("in");

}

MockMVC - Test get country service for exceptional scenario

SpringLearnApplicationTests.java

package com.cognizant.springlearn;

import com.cognizant.springlearn.controller.CountryController; import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.context.SpringBootTest; import org.springframework.test.web.servlet.MockMvc;

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import static org.junit.jupiter.api.Assertions.assertNotNull;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get; import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

import org.springframework.test.web.servlet.ResultActions;

@SpringBootTest

@AutoConfigureMockMvc

public class SpringLearnApplicationTests {

@Autowired

private CountryController countryController;

@Autowired

private MockMvc mvc;

@Test

public void contextLoads() {

assertNotNull(countryController);

}

@Test

public void testGetCountry() throws Exception {

ResultActions actions = mvc.perform(get("/country")); actions.andExpect(status().isOk())

.andExpect(jsonPath("$.code").value("IN"))

.andExpect(jsonPath("$.name").value("India"));

}

// ✅ Test for exceptional scenario: country not found @Test

public void testGetCountryException() throws Exception { ResultActions actions = mvc.perform(get("/countries/az"));

actions.andExpect(status().isNotFound()) // HTTP 404

.andExpect(status().reason("Country not found")); // matches @ResponseStatus

}

}

# CountryService.java

public Country getCountry(String code) throws CountryNotFoundException {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml"); List<Country> countryList = (List<Country>) context.getBean("countryList");

return countryList.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElseThrow(() -> new CountryNotFoundException("Country not found"));

}

## output

{

"timestamp": "...", "status": 404,

"error": "Not Found",

"message": "Country not found", "path": "/countries/az"

}