# 2. spring-rest-handson

# HTTP Request and Response

HTTP (Hypertext Transfer Protocol) is the foundation of data communication for the World Wide Web.   
It works as a request-response protocol between a client and server.  
- HTTP Request includes: Request Line (URL, Method), Headers (Content-Type, User-Agent), and Body (optional).  
- HTTP Response includes: Status Line (Status Code), Headers, and optional Body.  
  
Request Example:  
GET /hello HTTP/1.1  
Host: localhost:8083  
User-Agent: Chrome  
  
Response Example:  
HTTP/1.1 200 OK  
Content-Type: text/plain  
Hello World!!

# Need and Benefits of RESTful Web Services

REST (Representational State Transfer) is an architectural style that uses HTTP protocol for communication.  
Benefits:  
- Lightweight and stateless  
- Scalable and maintainable  
- Supports multiple formats (JSON, XML)  
- Easy to implement using Spring Boot  
- Based on standard HTTP methods: GET, POST, PUT, DELETE

# Hello World RESTful Web Service

URL: http://localhost:8083/hello

Method: GET

Response: Hello World!!

## Code - HelloController.java

@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;  
 }  
}

**REST - Country Web Service**

URL: http://localhost:8083/country

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

## Code - CountryController.java (Static)

@RequestMapping("/country")  
public Country getCountryIndia() {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = (Country) context.getBean("in");  
 return country;  
}

# REST Web Service - Get Country by Code

URL: http://localhost:8083/countries/in

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

## Code - CountryController.java (Dynamic)

@GetMapping("/countries/{code}")  
public Country getCountry(@PathVariable String code) throws Exception {  
 return countryService.getCountry(code);  
}

## Code - CountryService.java

public Country getCountry(String code) throws Exception {  
 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 Exception("Country Not Found"));  
}

# 6. country.xml Configuration

<bean id="in" class="com.cognizant.spring\_learn.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"/>  
 </list>  
 </constructor-arg>  
</bean>

# 7. REST API Testing with MockMvc

Test using @SpringBootTest and @AutoConfigureMockMvc for both `/hello` and `/countries/{code}` endpoints.

## Code - HelloControllerTest.java

@Test  
public void testSayHello() throws Exception {  
 mockMvc.perform(get("/hello"))  
 .andExpect(status().isOk())  
 .andExpect(content().string("Hello World!!"));  
}

## Code - CountryControllerTest.java

@Test  
public void testGetCountryIndia() throws Exception {  
 mockMvc.perform(get("/country"))  
 .andExpect(status().isOk())  
 .andExpect(jsonPath("$.code", is("IN")))  
 .andExpect(jsonPath("$.name", is("India")));  
}