Exercise 1: Setting Up RESTful Services

Topic: Setting up a Spring Boot project for an online bookstore.

- **Code:**

```
mvn archetype:generate -DgroupId=com.example.bookstoreapi -DartifactId=BookstoreAPI -Dversion=1.0.0 -DinteractiveMode=false


# Add dependencies in pom.xml
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

Exercise 2: Creating Basic REST Controllers

Topic: Creating and managing RESTful endpoints for books.

- **Code:**

```
@RestController
@RequestMapping("/books")
public class BookController {

  @GetMapping
```

```
    public List<Book> getAllBooks() {
        // Implementation
    }

    @PostMapping
    public Book addBook(@RequestBody Book book) {
        // Implementation
    }

    @PutMapping("/{id}")
    public Book updateBook(@PathVariable Long id, @RequestBody Book book) {
        // Implementation
    }

    @DeleteMapping("/{id}")
    public void deleteBook(@PathVariable Long id) {
        // Implementation
    }
}

@Data
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    private String author;
    private Double price;
    private String isbn;
}
```

Exercise 3: Handling Path Variables and Query Parameters

Topic: Fetching and filtering books using path variables and query parameters.

- **Code:**

```
@GetMapping("/{id}")
public Book getBookById(@PathVariable Long id) {
   // Implementation
}

@GetMapping("/search")
public List<Book> searchBooks(@RequestParam(required = false) String title,
                              @RequestParam(required = false) String author) {
   // Implementation
}
```

Exercise 4: Processing Request Body and Form Data

Topic: Handling JSON request bodies and form data for customer registrations.

- **Code:**

```
@PostMapping("/customers")
public Customer createCustomer(@RequestBody Customer customer) {
   // Implementation
}

@PostMapping("/register")
public String registerCustomer(@RequestParam String name, @RequestParam String email) {
   // Implementation
}
```

Exercise 5: Customizing Response Status and Headers

Topic: Customizing HTTP status codes and headers for book management.

- **Code:**

```
@ResponseStatus(HttpStatus.CREATED)
@PostMapping("/books")
public ResponseEntity<Book> createBook(@RequestBody Book book) {
    HttpHeaders headers = new HttpHeaders();
    headers.add("Custom-Header", "foo");
    return new ResponseEntity<>(book, headers, HttpStatus.CREATED);
}
```

Exercise 6: Exception Handling in REST Controllers

Topic: Implementing global exception handling for REST services.

- **Code:**

```
@ControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(ResourceNotFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public ResponseEntity<String> handleNotFound(ResourceNotFoundException ex) {
        return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT_FOUND);
    }

    // Other exception handlers...
}
```

Exercise 7: Introduction to Data Transfer Objects (DTOs)

Topic: Creating and mapping DTOs for books and customers.

- **Code:**

```
@Data
public class BookDTO {
    private Long id;
    private String title;
    private String author;
```

```
    private Double price;
}

// Example mapping with ModelMapper
public BookDTO convertToDTO(Book book) {
    ModelMapper modelMapper = new ModelMapper();
    return modelMapper.map(book, BookDTO.class);
}
```