Online Grocery Store API Implementation

---

1. Project Setup

1. **Create a Spring Boot Project using Spring Tool Suite (STS):**
    - **Dependencies:**
        - Spring Web (for RESTful API)
        - Spring Data JPA (for database access)
        - H2 Database (for an in-memory database)
        - Spring Boot Validation (for input validation)

---

2. Dependencies (`pom.xml`)

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

---

3. Database Configuration (`application.properties`)

```
spring.datasource.url=jdbc:h2:mem:grocerydb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
```

---

4. Entity Layer

`GroceryItem.java` Entity

```
package com.example.grocery;
```

```
import jakarta.persistence.Table;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.DecimalMin;

@Entity
@Table(name = "grocery_items")
public class GroceryItem {

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

    @NotBlank(message = "Description is required")
    private String description;

    @NotNull(message = "Price is required")
    @DecimalMin(value = "0.0", message = "Price must be positive")
    private Double price;

    @NotNull(message = "Quantity in stock is required")
    @DecimalMin(value = "0", message = "Quantity must be positive")
    private Integer quantityInStock;

    @NotBlank(message = "Category is required")
    private String category;

    @NotBlank(message = "Name is required")
    private String name;

    // Default constructor
    public GroceryItem() {}

    // Parameterized constructor
    public GroceryItem(String name, String description, Double price,
Integer quantityInStock, String category) {
        this.name = name;
        this.description = description;
        this.price = price;
        this.quantityInStock = quantityInStock;
        this.category = category;
    }

    // Getters and Setters

    public Long getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public Integer getQuantityInStock() {
        return quantityInStock;
    }

    public void setQuantityInStock(Integer quantityInStock) {
        this.quantityInStock = quantityInStock;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }
}
```

---

5. Repository Layer

GroceryItemRepository.java

```
package com.example.grocery;

import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface GroceryItemRepository extends JpaRepository<GroceryItem,
Long> {
    List<GroceryItem> findByNameContaining(String name);
    List<GroceryItem> findByCategory(String category);
}
```

---

6. Service Layer

GroceryItemService.java

```
package com.example.grocery;
```

```
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class GroceryItemService {
    private final GroceryItemRepository groceryItemRepository;

    public GroceryItemService(GroceryItemRepository groceryItemRepository)
{
        this.groceryItemRepository = groceryItemRepository;
    }

    public List<GroceryItem> getAllItems() {
        return groceryItemRepository.findAll();
    }

    public GroceryItem getItemById(Long id) {
        return groceryItemRepository.findById(id).orElseThrow(() -> new
RuntimeException("Item not found"));
    }

    public GroceryItem addItem(GroceryItem item) {
        return groceryItemRepository.save(item);
    }

    public GroceryItem updateItem(Long id, GroceryItem itemDetails) {
        GroceryItem item = getItemById(id);
        item.setName(itemDetails.getName());
        item.setDescription(itemDetails.getDescription());
        item.setPrice(itemDetails.getPrice());
        item.setQuantityInStock(itemDetails.getQuantityInStock());
        item.setCategory(itemDetails.getCategory());
        return groceryItemRepository.save(item);
    }

    public void deleteItem(Long id) {
        groceryItemRepository.deleteById(id);
    }

    public List<GroceryItem> searchItemsByName(String name) {
        return groceryItemRepository.findByNameContaining(name);
    }

    public List<GroceryItem> searchItemsByCategory(String category) {
        return groceryItemRepository.findByCategory(category);
    }
}
```

---

7. Controller Layer

GroceryItemController.java

```
package com.example.grocery;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
```

```
import java.util.List;

@CrossOrigin(origins = "http://localhost:8080")
@RestController
@RequestMapping("/api/grocery-items")
public class GroceryItemController {

    @Autowired
    private GroceryItemService groceryItemService;  // Corrected variable
name

    @GetMapping
    public List<GroceryItem> getAllItems() {
        return groceryItemService.getAllItems();
    }

    @GetMapping("/{id}")
    public ResponseEntity<GroceryItem> getItemById(@PathVariable Long id) {
        GroceryItem item = groceryItemService.getItemById(id);
        return item != null ? ResponseEntity.ok(item) :
ResponseEntity.notFound().build(); // Handle item not found
    }

    @PostMapping
    public GroceryItem addItem(@RequestBody GroceryItem item) {
        return groceryItemService.addItem(item);
    }

    @PutMapping("/{id}")
    public ResponseEntity<GroceryItem> updateItem(@PathVariable Long id,
@RequestBody GroceryItem itemDetails) {
        GroceryItem updatedItem = groceryItemService.updateItem(id,
itemDetails);
        return ResponseEntity.ok(updatedItem); // Return updated item
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteItem(@PathVariable Long id) {
        groceryItemService.deleteItem(id);
        return ResponseEntity.noContent().build();
    }
}
```

---

8. Global Exception Handling

GlobalExceptionHandler

```
package com.example.grocerystore.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(RuntimeException.class)
```

```
    public ResponseEntity<String> handleRuntimeException(RuntimeException
ex) {
        return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT_FOUND);
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleGenericException(Exception ex) {
        return new ResponseEntity<>("An error occurred: " +
ex.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Testing

- **Run the application** and use **Postman** or **cURL** for API testing:
  - **Add a new item:**
    `POST /api/grocery-items`
  - `{`
  - `    "name": "Apple",`
  - `    "description": "Fresh red apples",`
  - `    "price": 3.5,`
  - `    "quantityInStock": 100,`
  - `    "category": "Fruits"`
  - `}`
  - **Update an item:**
    `PUT /api/grocery-items/1`
  - **Search by name:**
    `GET /api/grocery-items/search/name?name=apple`
  - **Search by category:**
    `GET /api/grocery-items/search/category?category=fruits`

Explanation

1. **Entity Layer:** Defines `GroceryItem` with attributes and validation.
2. **Repository Layer:** Provides default CRUD methods and custom queries.
3. **Service Layer:** Contains business logic for handling inventory operations.
4. **Controller Layer:** Exposes RESTful endpoints.
5. **Global Exception Handler:** Manages runtime and generic exceptions.
6. **Validation:** Ensures input correctness with annotations like `@NotBlank` and `@Min`.