Spring Boot Annotation Examples

@SpringBootApplication

```java
@SpringBootApplication
public class MyApplication {
    public static void main(String[] args) {
        SpringApplication.run(MyApplication.class, args);
    }
}
```

@Controller w/ @GetMapping

```java
@Controller
public class MyController {
    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, World!";
    }
}
```

@Controller w/ @RequestMapping

```java
@Controller
public class MyController {
    @RequestMapping("/hello")
    public String hello() {
        return "Hello, World!";
    }
}
```

@RestController (Basic Overview)

```java
@RestController
public class MyRestController {
    // REST controller methods here
}
```

@RestController w/ @RequestMapping, @GetMapping, @PostMapping, @PutMapping, and @DeleteMapping

```java
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/products")
public class ProductController {

    private final ProductService productService;

    public ProductController(ProductService productService) {
        this.productService = productService;
    }

    @GetMapping("/{id}")
    public ResponseEntity<Product> getProductById(@PathVariable Long id) {
        Product product = productService.getProductById(id);
        if (product != null) {
            return ResponseEntity.ok(product);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @GetMapping
    public ResponseEntity<List<Product>> getAllProducts() {
        List<Product> products = productService.getAllProducts();
        return ResponseEntity.ok(products);
    }

    @PostMapping
    public ResponseEntity<Product> createProduct(@RequestBody Product product) {
        Product createdProduct = productService.createProduct(product);
        return ResponseEntity.status(HttpStatus.CREATED).body(createdProduct);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody Product product) {
        Product updatedProduct = productService.updateProduct(id, product);
        if (updatedProduct != null) {
            return ResponseEntity.ok(updatedProduct);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteProduct(@PathVariable Long id) {
        boolean deleted = productService.deleteProduct(id);
        if (deleted) {
            return ResponseEntity.noContent().build();
        } else {
            return ResponseEntity.notFound().build();
        }
    }
}
```

@Autowired in a Service class

```
@Service
public class MyService {
    @Autowired
    private MyRepository repository;
}
```

@Autowired in a Controller class

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class MyComponent {
    private MyDependency myDependency;

    @Autowired
    public MyComponent(MyDependency myDependency) {
        this.myDependency = myDependency;
    }

    // Other methods using myDependency
}
```

@Autowired in a Repository class

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

@Repository
public class MyRepository {

    private final JdbcTemplate jdbcTemplate;

    @Autowired
    public MyRepository(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    // Repository methods that use jdbcTemplate
}
```

@Autowired in a SpringBootApplication class

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MySpringBootApplication {

    @Autowired
    private MyService myService;

    public static void main(String[] args) {
        SpringApplication.run(MySpringBootApplication.class, args);
    }

    // Other application-related code here
}
```

@Service (Basic Overview)

```java
@Service
public class MyService {
    // Service methods here
}
```

@Service w/ @Qualifier

```java
import org.springframework.beans.factory.annotation.Qualifier;

@Service
public class MyService {
    @Autowired
    @Qualifier("specialRepository")
    private MyRepository repository;
}
```

@Service w/ @Profile

```
@Service
@Profile("development")
public class DevelopmentService {
    // Development-specific service implementation
}

@Service
@Profile("production")
public class ProductionService {
    // Production-specific service implementation
}
```

@Repository (Basic Overview)

```
@Repository
public class MyRepository {
    // Repository methods here
}
```

@Entity (Basic Overview)

```
import jakarta.persistence.Entity;

@Entity
public class Product {
    // Entity properties and methods
}
```

@Entity w/ @Table, @Id, @GeneratedValue, and @Column
Includes @Data and @Override

```java
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import java.math.BigDecimal;
import lombok.Data;

@Entity
@Table(name = "products") // Define the table name
@Data
public class Product {

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

    @Column(name = "product_name") // Customize column name
    private String name;

    @Column(name = "product_description") // Define the column name
    private String description;

    @Column(name = "product_price") // Define the column name
    private BigDecimal price;

    // Constructors, getters, and setters can be omitted due to Lombok

    // Override toString() for debugging and logging

    @Override
    public String toString() {
        return "Product{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", description='" + description + '\'' +
                ", price=" + price +
                '}';
    }
}
```

@Component (Basic Overview)

```java
import org.springframework.stereotype.Component;

@Component
public class MyComponent {
    // Component methods here
}
```

@Component w/ @Value

```java
import org.springframework.beans.factory.annotation.Value;

@Component
public class MyComponent {
    @Value("${my.property}")
    private String propertyValue;
}
```

@Configuration w/ @Bean

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MyConfig {
    @Bean
    public MyBean myBean() {
        return new MyBean();
    }
}
```

@PropertySource

```java
import org.springframework.context.annotation.PropertySource;

@PropertySource("classpath:application.properties")
public class MyConfig {
    // Configuration properties here
}
```

@SpringBootTest:

```java
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
public class MyIntegrationTest {
    // Integration test code here
}
```

@Configuration w/ @EnableWebSecurity, @Bean, and @Override

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    // Define a custom PasswordEncoder (not recommended for production)
    @Bean
    public PasswordEncoder passwordEncoder() {
        return NoOpPasswordEncoder.getInstance(); // For simplicity (not recommended for production)
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
                .antMatchers("/public/**").permitAll() // Allow access to public resources
                .antMatchers("/admin/**").hasRole("ADMIN") // Require ADMIN role for /admin/**
                .anyRequest().authenticated() // Require authentication for other requests
                .and()
            .formLogin()
                .loginPage("/login") // Specify custom login page
                .permitAll() // Allow access to the login page
                .and()
            .logout()
                .permitAll(); // Allow access to the logout page
    }
}
```

@Configuration w/ @EnableWebMvc

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
@EnableWebMvc
public class CorsConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        // Allow cross-origin requests from any origin for specific endpoints
        registry.addMapping("/api/**")
            .allowedOrigins("*")
            .allowedMethods("GET", "POST", "PUT", "DELETE")
            .allowedHeaders("*");
    }
}
```