Spring Boot Cheat Sheet

A quick reference guide to Spring Boot, covering essential annotations, properties, and best practices for rapid application development.

Core Annotations

Application Setup

`@SpringBootApplication`: A convenience annotation that combines `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan`. Marks the main class of a Spring Boot application.

Example:

```
@SpringBootApplication
public class MyApplication {
    public static void main(String[] args) {

SpringApplication.run(MyApplication.class,
args);
    }
}
```

Component Management

`@Component`	Marks a class as a Spring-managed component. Generic stereotype for any Spring-managed component.
`@Service`	Specialization of `@Component` for classes that hold business logic.
`@Repository`	Specialization of `@Component` for classes that provide data access (e.g., interacting with databases).
`@Controller`	Marks a class as a controller, handling incoming web requests.
`@RestController`	Combination of `@Controller` and `@ResponseBody`, indicating that the method's return value should be bound to the web response body.

Dependency Injection

`@Autowired`	Injects dependencies into a class. Can be used on fields, constructors, or setter methods.
	Example:
	`@Autowired`
	`private MyService myService;`

`@Qualifier`	Used with `@Autowired` to specify which bean to inject when there are multiple candidates.
	Example:
	`@Autowired`
	`@Qualifier("myServiceImpl")`
	`private MyService myService;`

`@Value`	Injects values from properties files or environment variables.
	Example:
	`@Value("${my.property}")`
	`private String myProperty;`

Web Development

Request Mapping

`@RequestMapping`	Maps web requests onto controller methods. Supports various attributes like `path`, `method`, `params`, `headers`, `consumes`, and `produces`.
	Example:
	`@RequestMapping(value = "/users", method = RequestMethod.GET)`
	`public List<User> getUsers() { ... }`

`@GetMapping`	Shortcut for `@RequestMapping(method = RequestMethod.GET)`.
	Example:
	`@GetMapping("/users/{id}")`
	`public User getUser(@PathVariable Long id) { ... }`

`@PostMapping`	Shortcut for `@RequestMapping(method = RequestMethod.POST)`.
	Example:
	`@PostMapping("/users")`
	`public User createUser(@RequestBody User user) { ... }`

`@PutMapping`	Shortcut for `@RequestMapping(method = RequestMethod.PUT)`.
`@DeleteMapping`	Shortcut for `@RequestMapping(method = RequestMethod.DELETE)`.
`@PatchMapping`	Shortcut for `@RequestMapping(method = RequestMethod.PATCH)`.

Request Parameters

`@PathVariable`	Extracts values from the URI path.
	Example:
	`@GetMapping("/users/{id}")`
	`public User getUser(@PathVariable Long id) { ... }`

`@RequestParam`	Extracts values from query parameters.
	Example:
	`@GetMapping("/users")`
	`public List<User> getUsers(@RequestParam(defaultValue = "10") int limit) { ... }`

`@RequestBody`	Binds the request body to a method parameter. Often used to receive JSON or XML data.
	Example:
	`@PostMapping("/users")`
	`public User createUser(@RequestBody User user) { ... }`

Response Handling

`@ResponseBody`	Indicates that a method's return value should be bound to the web response body. Typically used in `@RestController` classes.

`ResponseEntity<>`	Allows fine-grained control over the response, including status code, headers, and body.
	Example:
	`@GetMapping("/users/{id}")`
	`public ResponseEntity<User> getUser(@PathVariable Long id) { ... }`

`@ResponseStatus`	Sets the HTTP status code of the response.
	Example:
	`@ResponseStatus(HttpStatus.CREATED)`
	`@PostMapping("/users")`
	`public User createUser(@RequestBody User user) { ... }`

Data Management

JPA Repositories

`@Entity`: Marks a class as a JPA entity, representing a table in the database.

`@Id`: Specifies the primary key of an entity.

`@GeneratedValue`: Configures the strategy for generating primary key values (e.g., `IDENTITY`, `SEQUENCE`, `AUTO`).

`JpaRepository<T, ID>`: Provides basic CRUD operations and pagination for JPA entities. Extend this interface to create a repository for your entity.

Example:

```
public interface UserRepository extends
JpaRepository<User, Long> {
    List<User> findByLastName(String
lastName);
}
```

Data Source Configuration

Spring Boot automatically configures a data source if you provide the necessary properties in `application.properties` or `application.yml`.

Example properties:

```
spring.datasource.url=jdbc:mysql://localhost:3
306/mydb
spring.datasource.username=user
spring.datasource.password=password
spring.datasource.driver-class-
name=com.mysql.cj.jdbc.Driver
```

Transactions

`@Transactional`: Marks a method or class as transactional. Ensures that database operations are executed within a transaction. Supports various attributes like `propagation`, `isolation`, `timeout`, and `readOnly`.

Example:

```
@Transactional
public void transferFunds(Long
fromAccountId, Long toAccountId,
BigDecimal amount) { ... }
```

Advanced Features

Profiles

Spring Profiles allow you to configure different parts of your application for different environments (e.g., development, testing, production).

`@Profile`: Marks a component as being active only when a specific profile is active.

Example:

```
@Component
@Profile("dev")
public class DevDatabaseConfig { ... }
```

Activate profiles using the `spring.profiles.active` property in `application.properties` or `application.yml`.

Actuator

Spring Boot Actuator provides endpoints for monitoring and managing your application, such as health checks, metrics, and audit events.

Common endpoints include `/health`, `/metrics`, `/info`, `/beans`, and `/env`. These can be accessed over HTTP or JMX.

Enable Actuator by adding the `spring-boot-starter-actuator` dependency to your project.

Testing

`@SpringBootTest`: Loads the full application context for integration tests.

`@WebMvcTest`: Tests only the web layer, loading only the necessary components (e.g., controllers, filters, interceptors).

`@DataJpaTest`: Tests only the JPA components, setting up an in-memory database.

`@MockBean`: Mocks a bean in the application context, allowing you to isolate the component being tested.