Spring Annotations Cheat Sheet

@Component

Indicates that a class is a Spring-managed component.

```
@Component
public class MyService {
    public void serve() {
        System.out.println("Service is running...");
    }
}
```

@Controller

Marks a class as a Spring MVC controller.

```
@Controller
public class MyController {
    @RequestMapping("/home")
    public String home() {
        return "home";
    }
}
```

@Service

Denotes a service layer component.

```
@Service
public class UserService {
    public String getUser() {
        return "User";
    }
}
```

@Repository

Indicates a data access component (DAO) in the persistence layer.

```
@Repository

public class UserRepository {

    public List<String> findAllUsers() {

        return Arrays.asList("John", "Jane");

    }

}
```

@Autowired

Injects a dependency automatically.

```
@Service

public class UserService {

    @Autowired

    private UserRepository userRepository;

}
```

@ComponentScan

Scans for Spring components within the specified package.

```
@Configuration

@ComponentScan(basePackages = "com.example")

public class AppConfig {}
```

@Configuration

Indicates that a class declares one or more @Bean methods.

```
@Configuration

public class AppConfig {

    @Bean
```

```
    public MyService myService() {
        return new MyService();
    }
}
```

@Bean

Declares a Spring bean.

```
@Configuration
public class AppConfig {
    @Bean
    public MyService myService() {
        return new MyService();
    }
}
```

@Scope

Defines the scope of a Spring bean (singleton, prototype, etc.).

```
@Component
@Scope("prototype")
public class MyPrototypeBean {}
```

@Qualifier

Specifies which bean to use when multiple candidates are available.

```
@Service
public class UserService {
    @Autowired
    @Qualifier("userRepositoryV2")
    private UserRepository userRepository;
}
```

@RequestMapping

Maps HTTP requests to handler methods.

```
@Controller

@RequestMapping("/user")

public class UserController {

    @RequestMapping("/list")

    public String listUsers() {

        return "userList";

    }

}
```

@GetMapping, @PostMapping, @PutMapping, @DeleteMapping

Specialized versions of @RequestMapping for specific HTTP methods.

```
@RestController

public class UserController {

    @GetMapping("/user/{id}")

    public String getUser(@PathVariable String id) {

        return "User: " + id;

    }

}
```

@PathVariable

Binds a method parameter to a URI template variable.

```
@GetMapping("/user/{id}")

public String getUser(@PathVariable("id") String userId) {

    return "User ID: " + userId;

}
```

@RequestParam

Binds a method parameter to a query parameter.

```
@GetMapping("/user")
public String getUser(@RequestParam("name") String name) {
    return "User Name: " + name;
}
```

@RequestBody

Binds the body of a POST request to a method parameter.

```
@PostMapping("/user")
public String addUser(@RequestBody User user) {
    return "User added: " + user.getName();
}
```

@ResponseBody

Indicates that the return value of a method should be bound to the HTTP response body.

```
@GetMapping("/user")
@ResponseBody
public String getUser() {
    return "User";
}
```

@RestController

Combines @Controller and @ResponseBody (useful for RESTful web services).

```
@RestController
public class MyRestController {
    @GetMapping("/hello")
    public String hello() {
        return "Hello, World!";
```

```
    }
}
```

@Transactional

Manages transactional behavior at the method or class level.

```
@Service
public class AccountService {
    @Transactional
    public void transferMoney() {
        // transfer logic
    }
}
```

@EnableAutoConfiguration

Enables Spring Boot's auto-configuration mechanism.

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

@Value

Injects values from properties files.

```
@Component
public class AppConfig {
    @Value("${app.name}")
    private String appName;
```

```
    public String getAppName() {
        return appName;
    }
}
```