Spring Boot Cheat Sheet

Auto-Config · REST · Data JPA · Actuator

SETUP & STARTER

```
<parent>spring-boot-starter-parent</parent>
<dep>spring-boot-starter-web</dep>
```

Bootstrap via start.spring.io

BOM manages all dependency versions

```
mvn spring-boot:run  /  ./gradlew
bootRun
```

MAIN CLASS

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

Combines

@Config+@EnableAutoConfig+@Scan

REST CONTROLLER

```
@RestController
@RequestMapping("/api/users")
public class UserController {
@GetMapping("/{id}")
public User get(@PathVariable Long
id){...}
@PostMapping
public User create(@RequestBody User
u){...}
```

SPRING DATA JPA

```
@Entity public class User { @Id
@GeneratedValue Long id; }
interface UserRepo extends
JpaRepository<User,Long>{}
List<User> findByEmail(String email);
```

Derives queries from method names

```
@Query("SELECT u FROM User u WHERE
u.active=true")
```

DEPENDENCY INJECTION

```
@Service / @Repository / @Component
@Autowired  // field injection (avoid)
```

Constructor injection preferred:

```
public UserSvc(UserRepo repo){
this.repo=repo; }
@Bean  // manual bean in @Configuration
class
```

CONFIGURATION

```
application.properties / application.yml
server.port=8080
spring.datasource.url=jdbc:postgresql://...
@Value("${app.secret}")  private String
secret;
@ConfigurationProperties(prefix="app")
```

PROFILES

```
application-dev.yml  /
application-prod.yml
spring.profiles.active=dev
@Profile("prod")  class ProdBean {}
-Dspring.profiles.active=prod  (JVM arg)
SPRING_PROFILES_ACTIVE=prod  (env var)
```

EXCEPTION HANDLING

```
@RestControllerAdvice
@ExceptionHandler(UserNotFoundException.class)
public ResponseEntity<Error> handle(ex)
{
return ResponseEntity.status(404)...
}
@ResponseStatus(HttpStatus.NOT_FOUND)
```

VALIDATION

```
@NotNull @Size(min=2,max=50) String
name;
@Email String email;
@Min(0) int age;
public User create(@Valid @RequestBody
User u){}
```

Add spring-boot-starter-validation dep

```
@ControllerAdvice handles
MethodArgumentNotValidException
```

ACTUATOR

```
<dep>spring-boot-starter-actuator</dep>
management.endpoints.web.exposure.include=*
GET /actuator/health  /actuator/metrics
GET /actuator/env  /actuator/beans
```

Custom: implement HealthIndicator interface

LOGGING

```
private static final Logger log =
LoggerFactory.getLogger(MyClass.class);
log.info("User {} created",
user.getId());
logging.level.com.example=DEBUG
```

Uses Logback by default, SLF4J facade

COMMON GOTCHAS

Circular bean deps use @Lazy or constructor refactor

OpenEntityManagerInViewFilter N+1 queries

Transaction must be on public methods only

@SpringBootTest loads full context use slices

Flyway/Liquibase for DB migrations, not auto-ddl in prod