Spring Boot Cheat Sheet

Spring Boot 4.1.x / Spring Framework 6.2.x on Java 17/21+
Verified against docs.spring.io

GETTING STARTED & AUTO-CONFIG

```
curl start.spring.io/starter.zip \
  -d dependencies=web,data-jpa \
  -o demo.zip
mvn spring-boot:run
java -jar target/app.jar
```

@SpringBootApplication = @SpringBootConfiguration + @EnableAutoConfiguration + @ComponentScan

Conditions: **@ConditionalOnClass**, @ConditionalOnMissingBean, @ConditionalOnProperty

Boot 3: jakarta.* namespace. Boot 4: starter renames, e.g. spring-boot-starter-webmvc (was -web).

BEANS & DEPENDENCY INJECTION

@Component / @Service / @Repository / @RestController -- component-scanned beans

```
@Service
public class OrderService {
  private final OrderRepo repo; // ctor
  public OrderService(OrderRepo r) {
    this.repo = r;              // injection
  }                              // (recommended)
}
```

@Configuration + @Bean; scopes: singleton (default) | prototype | request | session

@PostConstruct / @PreDestroy lifecycle callbacks

@Aspect + @Around -- proxies back @Transactional/@Cacheable

CONFIGURATION & PROFILES

Precedence (high→low): CLI args > env vars > application-{profile}.yml > application.yml > @PropertySource > defaults

```
@ConfigurationProperties(prefix="app")
@Validated
public record AppProps(String name) {}

@Value("${app.name}")
private String name;
```

spring.profiles.active=dev @Profile("dev")

additional-spring-configuration-metadata.json → IDE autocompletion for custom properties

SPRING DATA -- REPOSITORY & TEMPLATE PAIRS

Store	Repository	Template/Client
SQL	JpaRepository	JdbcClient/JdbcTemplate
MongoDB	MongoRepository	MongoTemplate
Couchbase	CouchbaseRepository	CouchbaseTemplate
Neo4j	Neo4jRepository	Neo4jClient

Derived: findByLastName(...); @Query -- JPQL / Mongo JSON / N1QL / Cypher, per store

Pageable/Sort paging; @EnableJpaAuditing (+ Mongo/Neo4j equivalents) for @CreatedDate/@LastModifiedBy

CACHING

```
@EnableCaching
@Cacheable(value="items", key="#id",
    condition="#id>0", unless="#result==null")
T find(Long id) { ... }

@CachePut(value="items", key="#i.id")
@CacheEvict(value="items", allEntries=true)
```

Local: ConcurrentMapCacheManager or Caffeine (spring-boot-starter-cache + caffeine)

Distributed: RedisCacheManager; spring.cache.redis.time-to-live=10m

REST & gRPC APIs

```
@RestController
@RequestMapping("/orders")
class OrderController {
  @GetMapping("/{id}")
  Order get(@PathVariable Long id) {...}
  @PostMapping
  Order create(@Valid @RequestBody CreateOrder r){...}
}
```

@ExceptionHandler / @ControllerAdvice → ProblemDetail

WebFlux: RouterFunction<ServerResponse> vs annotated controllers

Spring gRPC: **@GrpcService** (server), @ImportGrpcClients (typed client injection)

WEB UI FRAMEWORKS

spring-boot-starter-thymeleaf -- natural templating

```
<p th:text="${msg}">placeholder</p>
<tr th:each="o : ${orders}">...</tr>
<form th:object="${form}">
  <input th:field="*{name}"/>
</form>
<div th:insert="~{frag :: header}"/>
```

Vaadin -- richer component-based UI (see Vaadin Reference). JSF via JoinFaces -- legacy only.

MESSAGING WITH KAFKA

```
kafkaTemplate.send("orders", key, val);

@KafkaListener(topics="orders", groupId="g1")
void onMessage(Order o, Acknowledgment ack) {
  process(o);
  ack.acknowledge();       // manual ack
}
```

Ack modes: MANUAL / MANUAL_IMMEDIATE; @RetryableTopic + dead letter topic (DLT) on exhausted retries

spring.kafka.consumer.* / spring.kafka.producer.* properties

SCHEDULING & SHEDLOCK

```
@EnableScheduling
@Scheduled(cron="0 0 2 * * *", zone="UTC")
@Scheduled(fixedRate=15_000, fixedDelay=30_000)

@EnableSchedulerLock(defaultLockAtMostFor="PT30M")
@SchedulerLock(name="job",
    lockAtMostFor="PT15M", lockAtLeastFor="PT1M")
void nightlyCleanup() { ... }
```

LockProvider impls: JdbcTemplateLockProvider, MongoLockProvider, RedisLockProvider

LOGGING

```
logging.level.root=INFO
logging.level.com.acme=DEBUG
logging.structured.format.console=ecs
logging.pattern.console=...
```

MDC.put("traceId", id) -- correlation across log lines

Prefer stdout over file logging under container orchestration (platform captures/ships stdout)

OBSERVABILITY

Micrometer -- metrics facade; Actuator exposes endpoints

```
management.endpoints.web.exposure.include=\
  health,metrics,prometheus,info
management.tracing.sampling.probability=1.0
```

/actuator/health /actuator/metrics /actuator/prometheus /actuator/info

Micrometer Tracing + OpenTelemetry (OTLP export) → Prometheus scrape → Grafana dashboard

REACTIVE PROGRAMMING (REACTOR)

Type	Emits
Mono<T>	0..1 element
Flux<T>	0..N elements

Operators: map, flatMap, zip, merge, filter, onErrorResume, retry -- assembled lazily, run only on subscribe

```
StepVerifier.create(flux)
  .expectNext(1, 2)
  .verifyComplete();
```

LOMBOK & MAPSTRUCT

@Getter @Setter @Builder @Value @RequiredArgsConstructor @Slf4j

```
@Mapper(componentModel="spring")
interface OrderMapper {
  @Mapping(target="id", ignore=true)
  OrderDto toDto(Order o);
}
```

Maven: order lombok-mapstruct-binding before mapstruct- processor in annotationProcessorPaths

TESTING

```
@ExtendWith(MockitoExtension.class)
class X {
  @Mock Repo repo;
  @InjectMocks Service svc;
  @Test void t() { verify(repo).save(any()); }
}
```

Slices: @WebMvcTest, @DataJpaTest, @DataMongoTest

@SpringBootTest + Testcontainers + @ServiceConnection -- one real container per dependency instead of mocks

PERFORMANCE TESTING (JMETER)

Test plan: Thread Group → Sampler → Assertion

```
${__P(threads,50)}   // parameterized property
mvn jmeter:jmeter jmeter:results
```

Correlate p95/p99 latency and throughput against /actuator/prometheus CPU/memory

API-FIRST DEVELOPMENT

REST: OpenAPI spec → openapi-generator-maven-plugin (codegen); springdoc-openapi (runtime docs, code-first)

gRPC: .proto → protobuf-maven-plugin (stubs)

Messaging: Avro .avsc → avro-maven-plugin; Confluent Schema Registry (backward/forward compat); AsyncAPI spec + @asyncapi/html-template

BUILD & QUALITY (MAVEN PLUGINS)

jacoco-maven-plugin -- prepare-agent, report, report-aggregate (multi-module)

maven-surefire-plugin (*Test, unit) vs maven-failsafe-plugin (*IT, integration-test phase)

maven-checkstyle-plugin, spotbugs-maven-plugin, maven-pmd-plugin -- fail build or report-only; exclude generated sources

ARCHITECTURE GLOSSARY

SOLID -- Single responsibility, Open/closed, Liskov substitution, Interface segregation, Dependency inversion

Hexagonal -- domain at the center; ports = interfaces the domain owns; adapters plug in from outside

DDD -- entity (identity), value object (no identity, immutable), aggregate (consistency boundary), domain event

Outbox -- business row + outbox row, one transaction; relay publishes to Kafka

Listen-to-yourself -- publish, then consume your own event like everyone else

Saga -- orchestration (central coordinator) vs choreography (event chain, no coordinator)