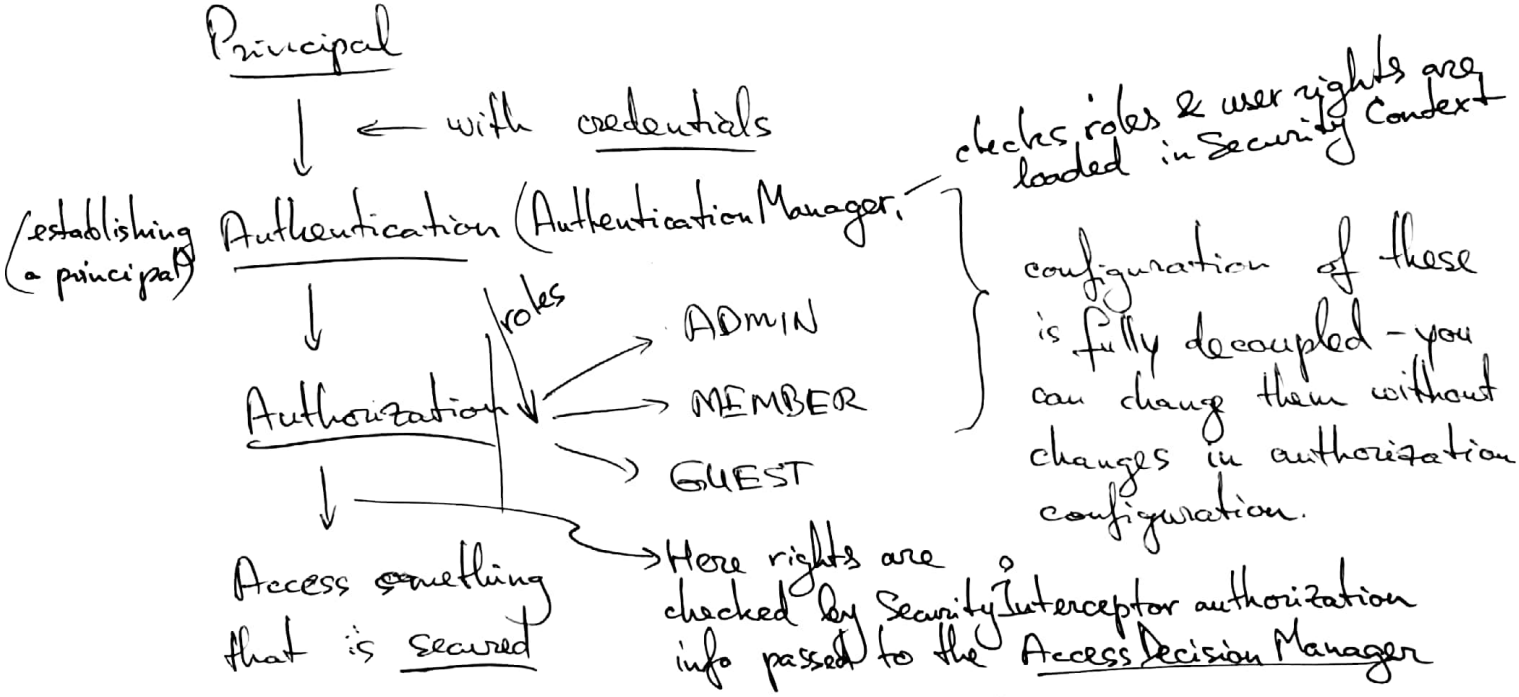
Spring Security capabilities

- Authentication
- Authorizing web requests
- Authorizing method calls
- Authorizing access to individual domain objects

Principal – someone / something that performs some action in application



??? Because Security is cross-cutting it is implemented using AOP

Configure S: Security

1. Declare security filter
2. Define Spring Security context
3. Configure Authentication & Authorization

XML Config:

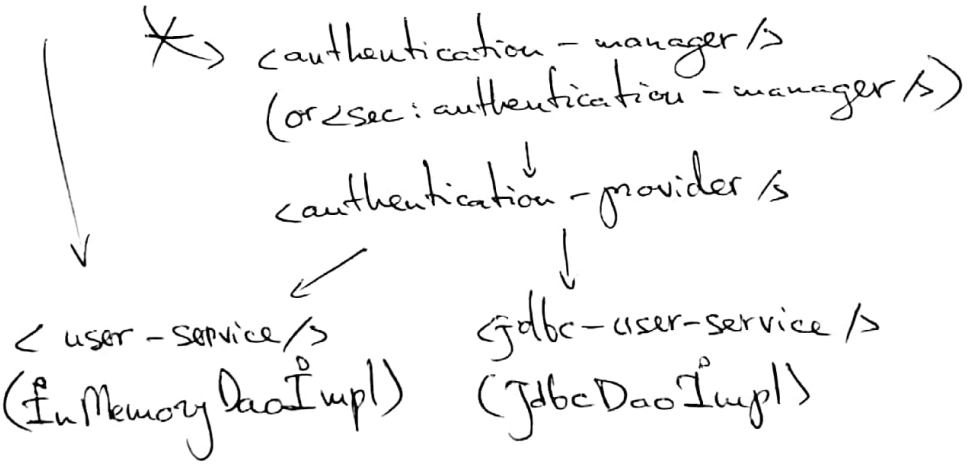
Requires: ① Include security filter

② Make spring security context the root context.

filter name must be: springSecurityFilterChain
(of type DelegatingFilterProxy)

is simplified by usage of special namespace: spring-security

UserDetailsService — used to provide credentials and authorities.



@Configuration
@EnableWebSecurity
} + extends WebSecurityConfigurerAdapter

1

→ creates servlet filter chain

+

without existing Spring

+ extends AbstractSecurityWebApplicationInitializer

2

→ registers servlet filter chain

with Spring MVC

+ extends AbstractAnnotationConfigDispatcherServletInitializer

+ getRootConfigClasses()

3

→ here the SecurityConfig class must be passed

→ initialises the dispatcher servlet

+ 2

Conclusion:

For MVC: 1 + 2 + 3 — 3 sets 1 as rootConfigClass

For non-MVC: 1 + 2 — 2 receives 1 as constructor param.

Config is done in 1 by:

- configure(HttpSecurity)

or AbstractDispatcherServletInitializer

AuthenticationManager

AuthenticationProvider

UserDetailsService / (+) passwordEncoder // jdbc-user-service

AuthenticationProvider (I) ⟹ DAOAuthenticationProvider (C)

- setUserDetailsService ()
- setPasswordEncoder ()

@Bean BCryptPasswordEncoder

@Bean UserDetailsService (I)

(C) InMemoryUserDetailsManager
(C) JdbcDaoImpl
(C) LdapUserDetailsManager

@EnableGlobalMethodSecurity (securedEnabled = true)

enables @Secured

NON-JSR250

enables method security.
May be combined with
@EnableWebSecurity

equivalent: <global-method-security secured-annotations = "enabled" />

The same, but the JSR 250 way: !!!

@EnableGlobalMethodSecurity (jsr250enabled = true)

enables @RolesAllowed ("ROLE_ADMI

equivalent:

<global-method-security jsr250-annotations = "enabled" />

Method annotations that:

- support use of expressions
- filter submitted collection arguments
- filter return values

→ @PreAuthorize
@PostAuthorize
@PreFilter
@PostFilter → filters returned collection

↓ !!! can access annotated method's parameters and use them in expressions

SecurityContextHolder

↳ ThreadLocal → SecurityContext

↳ Authentication → current principal

↳ currently logged user

what is stored depends on AuthenticationManager implementation

Authentication (i) request → ProviderManager (c) (AuthenticationManager (i)) —iterates the request→ list of AuthenticationProvider

① Checks for a AuthenticationProvider that can process the required type of Authentication

② Obtains a fully authenticated object including credentials

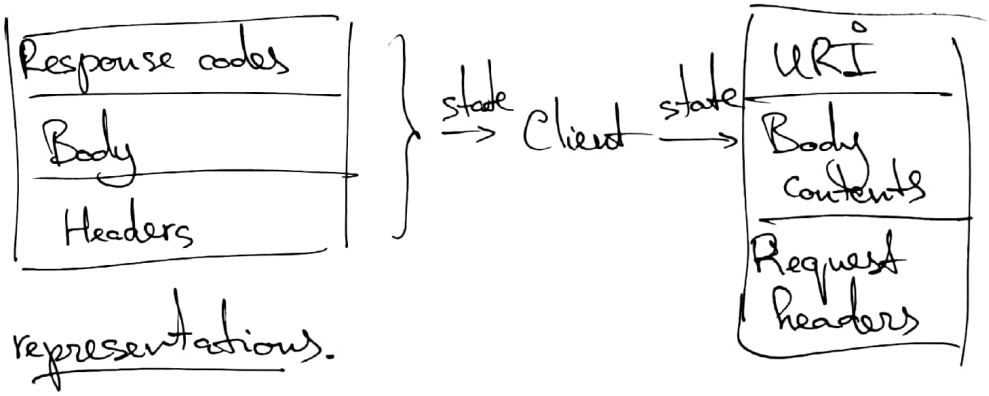
REST CRUD:

GET - READ → safe (doesn't change anything), idempotent (always same result)
POST - Create → non-safe, non-idempotent
PUT - Update → non-safe, idempotent
DELETE - Delete → non-safe, can be considered idempotent (on request to delete something that doesn't exist will always return 404 error.

Resource → Request from client → Representation (JSON or XML)

URI = Uniform Resources Identifier

HTTP response :



REST handles representations.

Session is used to preserve the state across many HTTP requests.
→ when working with a container
→ NOT required in REST, because of its stateless characteristics

JAX-RS — JEE standard for REST apps. V.2.0 latest major.

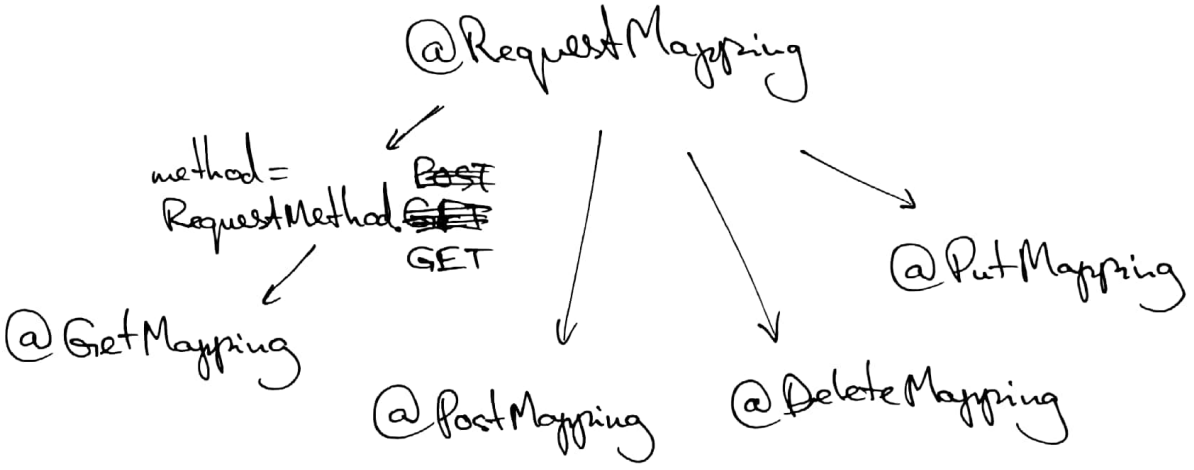
RestTemplate (c) — for building programmatic clients.

SpringMVC — ~~rest~~ REST support on the server side.

@ResponseBody — Notifies DispatcherServlet that the result of execution from controller must NOT be mapped to a view.

@RestController = @Controller + @ResponseBody
→ all methods return values to be bound to response body

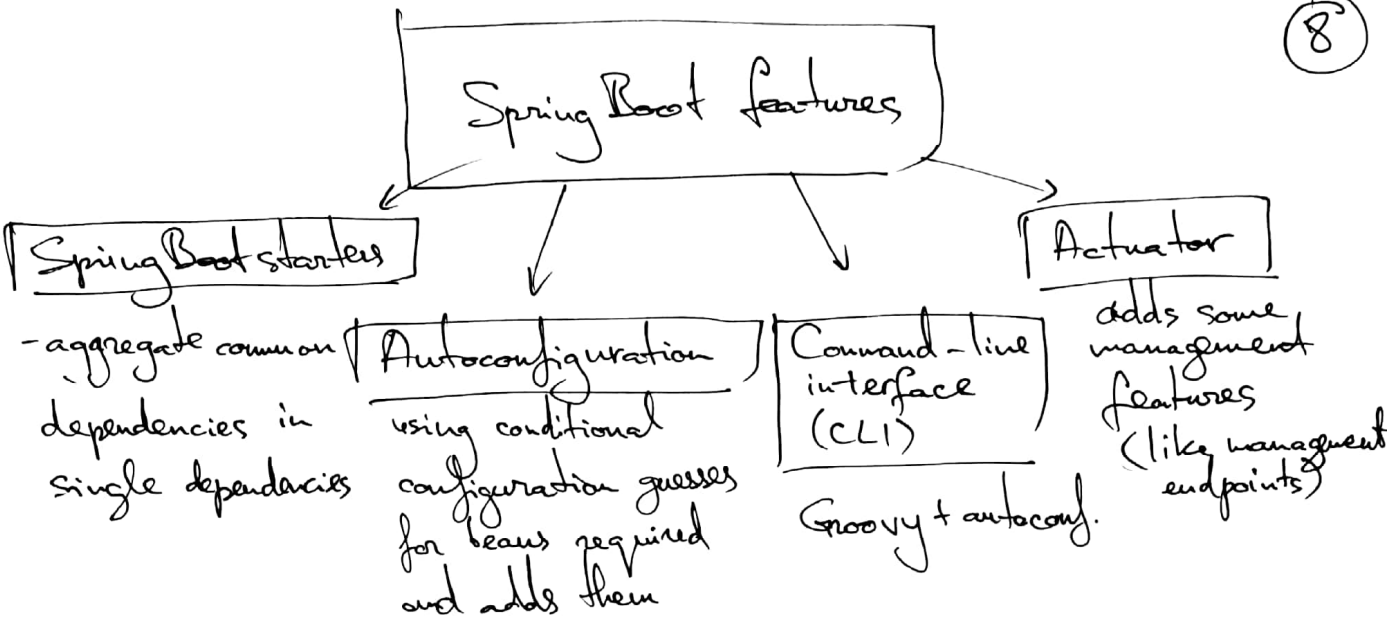
@ResponseStatus — used on controller methods for setting the status of response. (A HttpStatus constant)
- stops DispatcherServlet from finding a view to render.



If @RestController is NOT used a ResponseEntity being as a return value for some method of a @Controller will inform DispatcherServlet not to look for a view.

@RequestMapping(value="...", method=RequestMethod.PUT,
consumes = MediaType...,
produces = MediaType...)

specifies message converters to be used



S.B. detects JDBC module in classpath ⟹ automatically created JdbcTemplate

— " — H2 ⟹ — " — datasource

For H2: if schema.sql file is placed in resources it will be picked up & H2 is initialized with that schema.

S.B. bootstrap class → uses autoconfiguration to bootstrap a Spring Boot app.

@EnableAutoConfiguration — turns on S.B. autoconfiguration

@SpringBootApplication = @Configuration + @EnableAutoConfigurat + @ComponentScan

↳ has attribute "scanBasePackages" — works like in @ComponentScan

SpringBootServletInitializer — class implementing WebApplicationInitializer

↳ required for making a S.B. app. deployable on any Servlet 3 applicatio server. + configure() must be overriden too.

S.B. config file: - application.properties
- application.yml

@ConfigurationProperties(prefix = "....")

→ maps yml properties to class fields. These fields may be annotated with @NotNull

→ this class is passed to @EnableConfigurationProperties from the class that will require the bean with the properties

@PropertySource is NOT for YML !

@SpringBootTest → for test class.

Regression testing - running "old" tests on new version to see that they work well.

Cohesion - making a class single-purpose.

~~NetFlix Eureka~~

NetFlix OSS Eureka - service registration and discovery technology used mainly in AWS for load balancing.

SpringCloud Netflix - provides Netflix OSS (open source soft) integration with SpringBoot. With annotations can enable and configure:
- Eureka
- Zuul
- Hystrix
- Ribbon

Eureka is incorporated in SpringCloud and makes processes know of each other.

@EnableEurecaServer – registers app as a server.
injects an Eureca server instance in project.

@EnableDiscoveryClient – transforms app into a microservice discoverable by the Eureka instance.

Heartbeat – signal from Eureka clients to the server for informing on availability.

@LoadBalanced – marks a bean to be configured to use a LoadBalancedClient implementation

URL → location, i.e. http//: www.x.com
URI → resource, i.e. http//: www.x.com/something.

- @Configuration_classes MUST have a default /no-arg constructor. — and may not use @Autowired for constructor parameters
- JSR-250 annotations are enabled by adding a dependency — are NOT embedded in Spring.
- <context:component-scan/> = detect stereotypes + enable <context:annotation-config/>
- Pointcut expressions use ||, ~~not~~ && &&, !
- SpringSecurity <intercept-url/> uses Ant NOT Regex patterns by def.
- ResponseStatusExceptionResolver — maps exceptions to Httpcodes using @ResponseStatus.
- @ResponseStatus on @RestController is not supported.
- Spring test has nothing to do with mocks
- Spring AOP only for public methods

only for Servlet classes:
- MockHttpSession
- MockHttpContext

- Destroy-methods of prototype beans are never called.
- BeanPostProcessor creates AOP proxies.
- Spring gives a name for a bean with no id/name set that is made of full reference + a number.

ResultSetExtractor — single object (list ex.) with mapped entities

~~Result~~ RowMapper — single entity mapper

@Scope — define bean scope.

Portlet — session is [different] from global session

Servlet — " — [is the same as] " — .

Force use of CGLib → <proxy-target-class [="true"]/>

Bean lifecycle, @Qualifier — attribute, <meta key="..." value="..."/>

ProceedingJoinPoint → proceed() in @Around may be called [0 or more times].

DataAccessException → convert proprietary ex. to runtime ex.
[Runtime exception = unchecked exception]

DriverManagerDataSource. setDriverClassName
setUrl
setUsername
setPassword

@Autowiring collections

Isolation levels + phantom reads.

	Dirty reads (uncommited)	Non-repeat. reads (same row)	Phantom reads (same search)
read_Uncommited	✓	✓	✓
Read_commited	X	✓	✓
Repeatable_read	X	X	✓
Serializable	X	X	X

Serializable → highest level

~~jdbc~~ JdbcTemplate.update() returns number of rows affected.

EntityManagers → @PersistenceContext EntityManagerFactory → Container Managed Entity Manager !

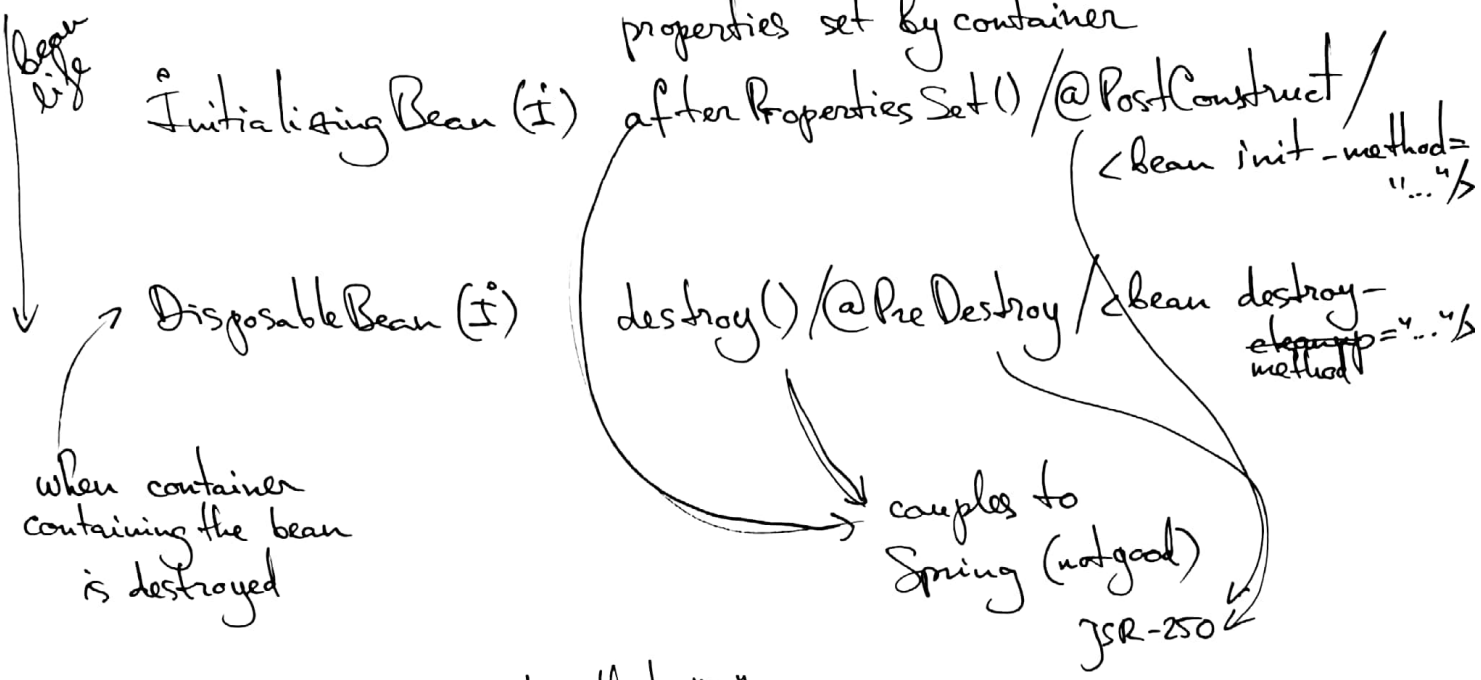
EntityManagers → @PersistenceUnit EntityManagerFactory → App. managed Entity Manager.

REST safe — don't change things

REST idempotent — always return same result.

Weaving — aspects + objects = advised object.

test:
- @TransactionConfiguration (TYPE) (deprecated) → defaultRollback → true by default; → transactionManager
- use ↙
- @Rollback (value=true) (TYPE, METHOD) + @Transactional (transactionManager="...") (TYPE, METHOD)



<beans default-init-method="init"
default-destroy-method="destroy">

@PreDestroy → destroy() of DisposableBean(I) → destroy-method of <bean>

if different method names

! if same name ⇒ called only once.

PropertyEditor — converts string from configuration file to int.

<context:property-placeholder location="...properties" /> = @PropertySource("...") + @Bean PropertySourcePlaceholderConfigurer

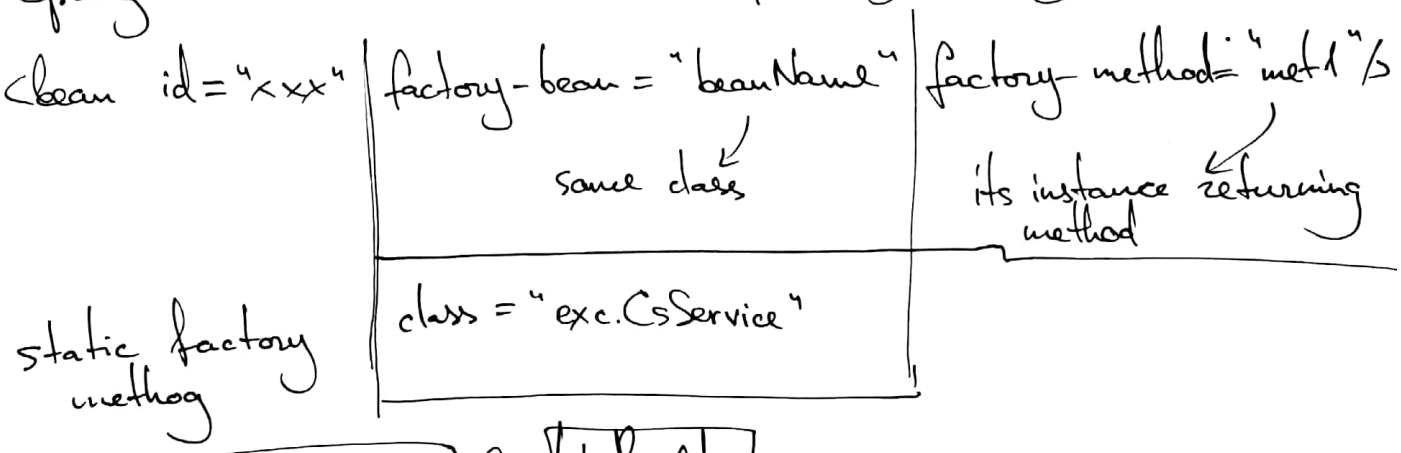
- REST is NOT a protocol ∴
- <null><null> ⇒ null
- empty argument for property = empty string

@Configuration annotated classes must NOT be final.
must have default constructor

@Bean methods must NOT be final.

@Import – imports one config class into another
(TYPE)

Spring beans DON'T HAVE TO follow ~~Java Bean~~ JavaBean spec.



JDK dynamic proxy By default

<intercept-url>
filters ⇒ "none" ⇒ bypass SpringSecurityFilterChain.
similar to
<http security="none" />

SpringBoot → logging → internal → Commons Logging
→ with starters → LogBack

JdbcTemplate can execute ANY SQL statement including DDL.

@Required – property must be given a value in the XML.
(METHOD) – for setters

@EnableAspectJAutoProxy — enables @Aspect → an AspectJ ann.
(TYPE)

```
@Configuration
@EnableAspectJAutoProxy
@ComponentScan
public class ConfigClass
```

→

```
@Component
@Aspect
public class SomeAspect
...
@Pointcut
```

1. Before
2. After
3. After throwing
4. After returning
5. Around

@RestController IS a stereotype annotation.

@Bean CommonAnnotationBeanPostProcessor or <context:annotation-config/>

@Configuration → enable:
- @PreDestroy
- @PostConstruct

HandlerMapping — maps incoming web requests to appropriate handlers.

SpringSecurity filters (ordered; only 3)
1. BasicAuthenticationFilter
2. AnonymousAuthenticationFilter
3. FilterSecurityInterceptor

@Profile — TYPE, METHOD

JpaTransactionManager (C) ← PlatformTransactionManager (I)

↓

single JPA EntityManagerFactory

classpath*: → all matches from classpath jars.

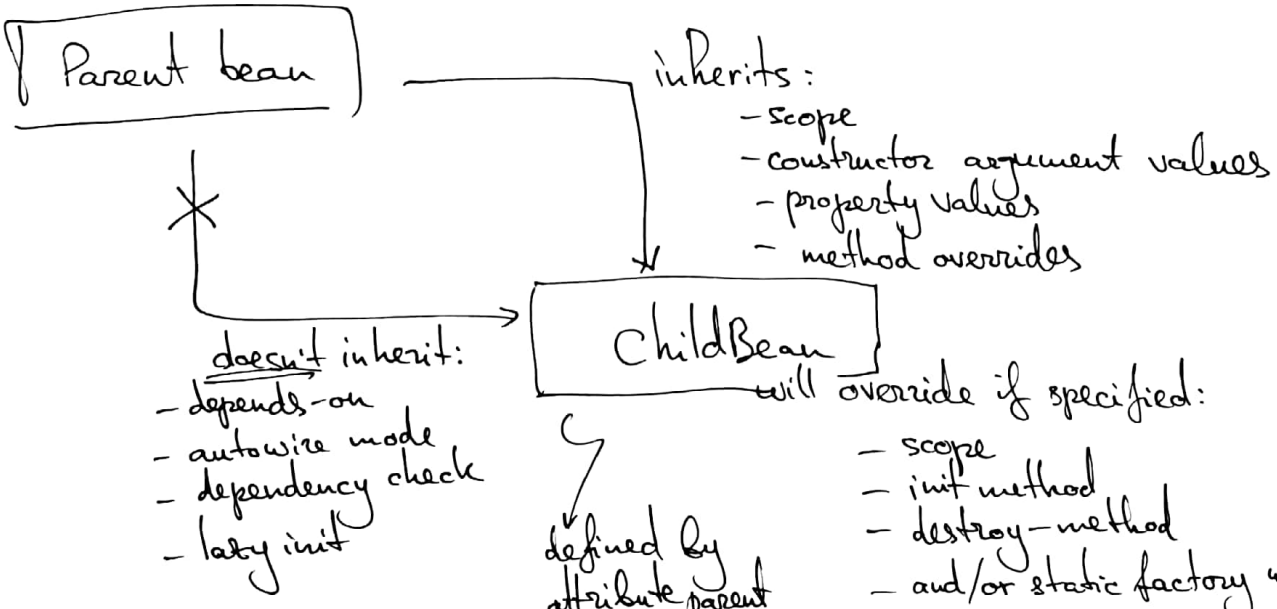
classpath: → one match, first found

TransactionInterceptor is used for applying transactional advice, & in declarative transactions.

@EnableTransactionManagement = <tx:annotation-driven/>

↳ schema tx

Client must release resources prototypes are holding + do the cleanup (destruction)



If parent DOES NOT specify a class explicitly abstract attribute must be set to true. (for parent)

@RequestParam (required = "...")

↳ NOT "mandatory"

For AOP - Spring uses AspectJ annotations.

PlatformTransactionManager

commit() ← | ↓ rollback() | → getTransaction()

getTransaction() → TransactionStatus.

Http Status codes:

1xx	info
2xx	success
~~3xx~~	Redirection
4xx	client errors
~~5xx~~	server errors

Only 1 constructor method with @Autowired can be set required !

XML style named pointcut

@AspectJ style named pointcut

these 2 cannot be combined

! You CAN combine 2 @AspectJ pointcuts

@EnableAspectJAutoProxy = <aop:aspectj-autoproxy>

200 → OK

~~400~~ ~~204~~ 204 → NO content

! Primitives can not be autowired

Access only by HTTPS: <intercept-url requires-channel="https" .... />

DriverManagedDataSource DOES NOT provide connection pooling.

In XML "profile" is applicable only on <beans> NOT on <bean />

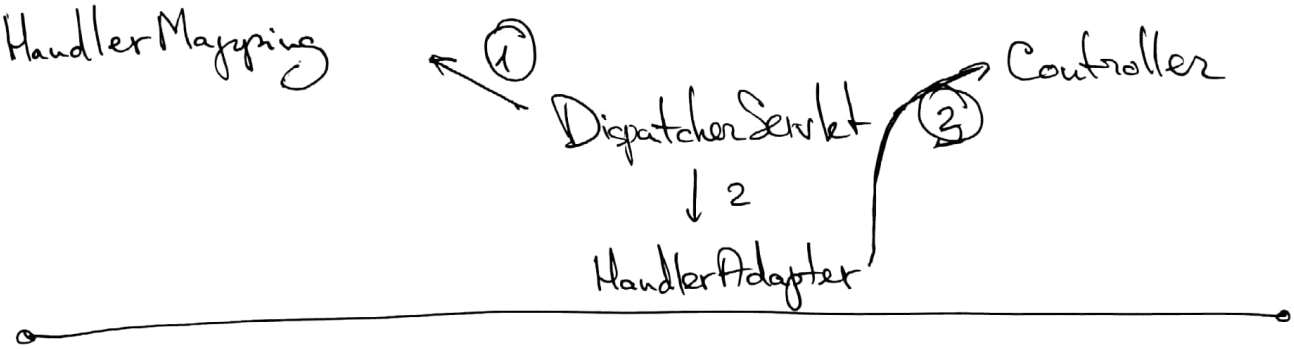
<context:property-placeholder location="..." />

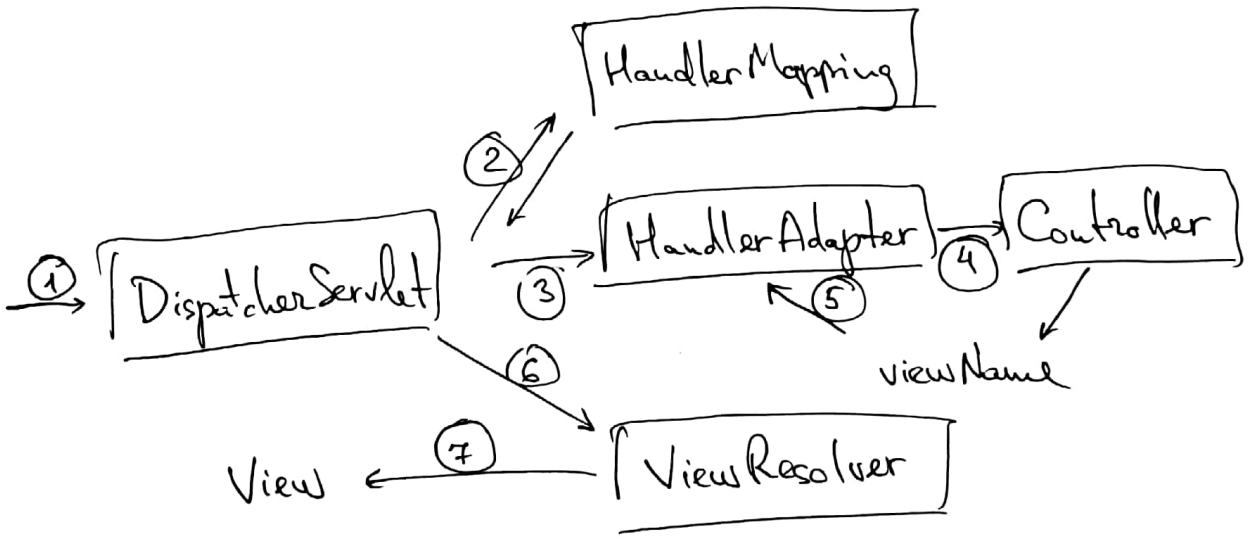
JSR 330 @Qualifier
Spring @Qualifier
} → Both can be used for creating custom qualifiers.

- implement Controller (I)
- extend Controller implementations
} ⇒ Controller class

BeanNameUrlHandlerMapping — default HandlerMapping bean.

More meta-annotations ⇒ Composed annotations.





HandlerAdapter – invokes Handler methods selected by HandlerMapping
↳ selects method
↖ selects controller

@PersistenceUnit injects an EntityManagerFactory
@PersistenceContext injects an EntityManager.

@Transactional → rollBackFor
→ rollbackForClassName
→ noRollBackFor
→ noRollbackForClassName
→ propagation
→ isolation
→ timeout
→ transactionManager / value (alias)
→ readOnly

Best: prototype → stateful
singleton → stateless

@RequestBody (PARAMETER)
@ResponseBody (TYPE, METHOD)

@RequestMapping("/")
public String myMethod (HttpSession h) → will be populated with current HttpSession obj. - automatically

Invocation:
II. setter based DI methods
I. no-argument static factory method. } ~~DO~~

Circular dependency ⟹ BeanCurrentlyInCreationException

In same bean ~~DO NOT~~ DO mix setter-based with constructor-based DIs

JdbcTemplate.queryForObject ⇒ single object, not list of.

Config
⇓
BeanDefinitions ← BeanFactoryPostProcessor (modifies internal BeanFactory) ↖ after context initialization
(ex. PropertyPlaceholderConfigurer - replaces ${...} with values)
⇓
instantiation
⇓
beans (instances) ← BeanPostProcessor (AOP proxies ← ex.) ⇒ IoC Container

Entities (more) ← PersistenceContext

EntityManager - tool for manipulating with Entities.

@Component } @Bean LiteMode - inter @Bean relations won't
@Bean } work, like in AOP

@Named / @ManagedBean ⟺ @Component
@Inject / @Resource ⟺ @Autowired

@ContextConfiguration } ⇒ looks for A-context.xml
public class A

id + idref
early validation of XML

@Entity (TYPE) ⇒ no method annotating

<beans default-lazy-init="true">
<bean lazy-init="true">
!

DelegatingFilterProxy → springSecurityFilterChain

web container — Spring context

XML inheritance ⇒ "parent" attribute

Java –"– ⇒ simple Java inheritance, but no means to create abstract beans

@Transactional — only for public methods. (Because of AOP)

SpEL supports user-defined functions.

Class constructor is called before any DI

Idempotent ⇒ making multiple identical requests has the same effect as making a single request.

<password-encoder/> NOT <encode-password/>

BeanPostProcessors: ① postProcessBeforeInitialization()

② @PostConstruct

③ afterPropertiesSet() of InitializingBean (I)

④ init-method

bean creation ↓

④ postProcessAfterInitialization()

JDK proxy — only public methods

CGLib proxy — public & protected methods

@Transactional properties —

<ref bean="..." parent/>