→Attribute

-тип објект

-може да се променат во филтерите

-овозможуваат структурирање на кодот според MVC

-може getAttribute и setAttribute

→Parameter

-тип string

-не може да се променат во филтерите

-овозможуваат структурирање на кодот според MVC

-може getInitAttribute и не може set

	Атрибут	Параметар
Опсег на важност	ServletContext Request Session	ServletContext ServletConfig Request
Тип на податок	Object	String
Метод за читање	getAttribute(String ime)	getInitParameter(String ime)
Метод за поставување	setAttribute(String ime, Object vrednost)	Не може да се постави во код, туку во web.xml или преку анотации

→Според тоа од каде може да им се пристапи на параметрите и атрибутите во рамките на една веб апликација се дефинираат следните области на важење:

Основи на веб програмирање со користење на Spring рамката

• Сервлет - важи само за параметри и тоа на ниво на сервлет

• Контекст - важи и за параметри и за атрибути и на ниво на цела апликација

• Барање - важи и за параметри и за атрибути на ниво на барање

• Сесија - важи само за атрибути на ниво на цела апликација, но само за еден ист корисник

→**@Scope** се користи за дефинирање на животен век на 1 бин

-“singleton” –се креира бин еднаш и се користи секаде каде што треба

-“prototype”-се креира бин за секое барање

-“request”/”context”-се креира бин важечки само за вредности на HTTP request

-“session”- се креира бин важечки само за вредности на во рамки на 1 сесија

→AOP-Aspect Oriented Programming

-место во извршувањето на апликацијата каде се вклучува аспектот -> JOIN POINT

-дефинира каде може да се изврши аспектот -> POINT CUT

-дефинира која задача и кога ќе се изврши -> ADVICE

Credits:mi40

-Во **web.xml** има <context-param> и <init-param>

→**<context-param>** -> може да се менува и земе преку context со context.getInitParameter()

-Не може директно да се промени со ServletConfig, туку само со ServletContext, преку setAttribute()

-важат за сите корисници (глобално)

→**<init-param>** -може да се земе со config.getInitParameter(), но не да се менува

→**Config=getServletConfig()** -> вака може само getInitParam() -> ќе се земе вредноста од web.xml ако има init-param или од @WebInitParam

→**Context=getServletContext()** -> вака може и getAttribute() и setAttribute()

-промена кај context -> значи генерална промена кај сите корисници

→**Session=getSession()** -> може и set get со тоа што кога има get мора кастирање

-промена кај сесија не значи промена кај сите корисници туку само кај корисникот кој ја направил промената

→**DispatcherToServletB.forward(req,resp)**

-може да се пристапи до параметри како и сите видови на атрибути(request,session)

-контејнерот ке го повика сервлетот мапиран на servletB.do

→**Response.sendRedirect("/servletC.do")**

-може да се пристапи до само session атрибутите

- контејнерот ке го повика сервлетот мапиран на servletC.do

-клиентот ќе направи ново барање до /servletC.do

@PathVariable -> students/5 "{/id}"

@RequestParam -> како query students?id=5

Request and Response

→ако клиентот клика на линк и се враќа модифицирана истата страна

-GET е барањето,ContentType/Length/Body во request нема

-во response->ContentType=text/html, ContentLength=нумеричка вредност,Body е низа од бајти

→ако клиентот пополнува форма и клика на копче,а потоа е пренасочен на друга страна

-POST е барањето, ContentType=application/x-www-form-urlencoded,ContentLength/Body=низа од бајти во request

-во response 302 Redirect ,кога има редиректирање нема ContentType/ContentLenght/Body,има Location

→ако клиентот клика на копче upload а претходно одбрал pdf датотека

- POST е барањето, ContentType=multipart/form,ContentLength=нум вредност,Body=низа од бајти во request

Credits:mi40

- во response->ContentType=application/pdf, ContentLength= нум вредност ,Body е низа од бајти

→ако клиентот клика на копче Download на pdf

- GET е барањето, ContentType/ContentLength/Body=нема во request

- во response->ContentType=application/pdf, ContentLength= нум вредност ,Body е низа од бајти

→**Инверзија на контрола** овозможува декларативно означување на зависностите и нивно имплицитно поврзување во процесот на креирање на објектите

-Секоја инстанца од класа која е зачувана во контектстот се нарекува **bean**

-Интерфејсот **BeanFactory** ги дефинира напредните механизми за конфигурација, кои овозможуваат управување со сите типови на објекти

-поставување на бинови преку web.xml или јава конфигурација со анотации

→**автоматско вметнување на зависностите** на својствата анотирани со **@Autowired** (оваа анотација дава инструкција на контејнерот да го пронајде најсоодветниот бин и да го вметне како вредност на анотираното својство) и се користи во следните случаи:

• при вметнување на зависности анотирани со @Autowired анотацијата,

• при вметнување на аргументи на конструкторот за компоненти од кои се креираат бинови и

• при вметнување на аргументи на методите анотирани со @Bean

-**@ComponentScan** -му кажува на Spring IoC контејнерот дека треба да ги скенира сите класи кои се анотирани @Component или некоја изведена од неа како : @ControllerAdvice, @Controller, @Repository, @JsonComponent, @TestComponent, @Service, @Configuration...

-@Component-анотација наменета за означување на класи за кои Spring IoC контејнерот имплицитно ќе креира и регистрира бинови.

-@Bean-анотација наменета за означување на методи чиј резултат ќе биде експлицитно креиран бин кој ´ке се регистрира

-доколку во Spring IoC контејнерот имаме 3 бинови од потребниот тип со имиња "x1 "x2"и "x3 а имаме потреба да ја разрешиме зависноста **@Autowired** X x1, тогаш ´ке се искористи бинот со име "x1"од контекстот како еднозначен кандидат

-Во случаите кога го знаеме специфичниот бин кој сакаме да го вметнеме, но сакаме да искористиме друго име на променливата, се препорачува означување на зависноста со анотацијата @Qualifier("beanId"), преку која експлицитно дефинираме кој бин сакаме да се искористи како зависност. Во погорниот пример, зависноста би требало да биде анотирана на следниот начин: @Autowired **@Qualifier("x1")** X myX

-кога бинот кој сакаме да го вметнеме зависи од околината во која ја стартуваме апликацијата, се препорачува користење на **@Profile**("client-x1") анотацијата при декларацијата на компонентата @Profile("client-x1")@Component

@Value анотацијата во Spring овозможува вметнување на вредности во биновите од различни извори. Се користи за вчитување вредности од property фајлови, системски променливи, аргументи на командната линија итн

Credits:mi40

Прашања за на interview:

1.Што е Spring Boot и како се разликува од Spring Framework?

a) Spring Boot е алатка за управување со сервери.

b) Spring Boot е фрејмворк кој обезбедува автоматска конфигурација и го поедноставува развојот на Spring апликации.

c) Spring Boot е база на податоци за Java апликации.

Точен одговор: b)

2.Објасни ја улогата на @SpringBootApplication анотацијата.

a) Ја стартува апликацијата и ги вклучува сите потребни компоненти.

b) Ја конфигурира базата на податоци.

c) Ја прикажува веб-страната на клиентот.

Точен одговор: a)

3. Што е Dependency Injection и како се имплементира во Spring Boot?

a) Метод за управување со бази на податоци.

b) Концепт каде што Spring автоматски управува со создавање и поврзување на зависности.

c) Алатка за тестирање на апликации.

Точен одговор: b)

4.Што прави анотацијата @RestController и како се разликува од @Controller?

a) @RestController враќа JSON или XML податоци директно, додека @Controller враќа HTML преку view.

b) @RestController враќа HTML, а @Controller враќа JSON.

c) Нема разлика помеѓу нив.

Точен одговор: a)

5. Објасни ја употребата на @RequestMapping и како се разликува од @GetMapping, @PostMapping, итн.

a) @RequestMapping е генерален мапер за HTTP методи, додека @GetMapping, @PostMapping, итн. се специјализирани за одредени HTTP методи.

b) @RequestMapping е само за GET барања.

c) @RequestMapping автоматски ги обработува сите методи.

Точен одговор: a)

6. Каква е улогата на @Service, @Repository и @Component анотациите?

a) Сите три служат за креирање на бинови, но имаат различна намена.

b) Се користат само во контролери.

c) Се исто, нема разлика.

Точен одговор: a)

7.@SpringBootApplication ги комбинира следниве 3 анотации:

a) @AutoConfiguration,@Service,@Component

b) @ComponentScan,@Component,@Configuration

Credits:mi40

c) @Configuration, @EnableAutoConfiguration,@ComponentScan

Точен одговор: c)

8. Што се профили (Profiles) во Spring?

a) Профилите во Spring се користат за дефинирање различни конфигурации или компоненти (beans) за различни средини, како развој, тестирање или продукција.

b) Профилите во Spring се користат за управување со автентикација на корисници и улоги.

c) Профилите во Spring се механизам за групирање на компоненти (beans) врз основа на нивниот тип.

Точен одговор: a)

9. Што е IOC контејнер во Spring?

a)IOC (Inversion of Control) контејнерот во Spring е одговорен за управување со животниот циклус и зависностите на компонентите (beans) во Spring апликацијата.

b) IOC контејнерот во Spring се користи за контрола на базите на податоци.

c) IOC контејнерот во Spring е алатка за автентикација и авторизација на корисници.

Точен одговор: a)

10. За што е користен Spring?

a) Java Framework

b) Web Development Framework

c) MVC Framework

Точен одговор: a) b) c)

11.Кои од следниве се користени од Maven?

a) Pom.xml

b) Config.xml

c) META-IN

Точен одговор: a)

12. Што е JEE и зошто е потребна?

a) JEE е рамка за развој на фронтенд апликации.

b) JEE е платформа за развој на скалабилни и сигурни бизнис апликации.

c) JEE е само за управување со бази на податоци.

Точен одговор: b)

13. Кој е животниот циклус на еден сервлет?

a) init(), service(), destroy().

b) start(), run(), stop().

c) create(), use(), delete().

Точен одговор: a)

14.Што подразбира принципот на единствена одговорност (Single Responsibility Principle - SRP)?

a) Класата треба да има само една причина за промена.

b) Класата треба да се фокусира на повеќе задачи истовремено.

c) Секој метод треба да се одговорен за повеќе од една задача.

Credits:mi40

Точен одговор: a)

15. Која од следниве изјави е точна во контекст на принципот на отворено/затворено (Open/Closed Principle - OCP)?

a) Класите треба да бидат затворени за модификација, но отворени за екстензија.

b) Класите треба да бидат отворени за модификација и затворени за екстензија.

c) Класите треба да бидат затворени и за модификација и за екстензија.

Точен одговор: a)

16. Што подразбира принципот на заменливост на Лисков (Liskov Substitution Principle - LSP)?

a) Поткласите треба да можат да бидат заменети со свои суперкласови без да се наруши функционалноста.

b) Поткласите треба да имаат различно однесување во споредба со своите суперкласи.

c) Поткласите треба да имаат повеќе методи отколку суперкласите.

Точен одговор: a)

17. Која е разликата помеѓу принципот на интерфејс на сегрегација (Interface Segregation Principle - ISP) и принципот на единствена одговорност?

a) ISP се фокусира на тоа дека клиентите не треба да зависат од интерфејси кои не ги користат, додека SRP се фокусира на тоа дека класите треба да имаат една причина за промена.

b) ISP се фокусира на тоа дека класите треба да имаат повеќе од една одговорност, додека SRP не дозволува тоа.

c) ISP не се однесува на интерфејсите, туку на имплементацијата на класите.

Точен одговор: a)

18. Што подразбира принципот на инверзија на зависности (Dependency Inversion Principle - DIP)?

a) Модулите на високо ниво не треба да зависат од модули на ниско ниво, туку и двата треба да зависат од апстракции.

b) Модулите на ниско ниво треба да зависат од модули на високо ниво.

c) Зависи од тоа дали модули на ниско ниво се апстракции.

Точен одговор: a)