

Решенија и објаснувања за прв колоквиум теорија веб прог од мацо 🐱  
Најдобар и најбрз начин да добиеш 30% за да го положиш овој дел 😎  
Лајк енд субскрајб 👍

J2EE								Spring		Spring MVC				
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Question 1

Not yet answered

Marked out of 1.00

Flag question

Доколку серверот успешно враќа JSON како резултат, кои од следните тврдења се точни?

Select one or more:

- ☐ a. Одговорот е со Status Code 404
- ☐ b. Клиентот го детектира типот на податоците од содржината во телото на одговорот
- ☐ c. Одговорот е со Status Code 500
- ☒ d. Одговорот е со Status Code 200
- ☐ e. Одговорот содржи ContentType параметар со вредност application/json
- ☒ f. Одговорот содржи ContentType заглавје со вредност application/json

Give your reasons

↶ A ▾ B I ≡ ≡ 🔍 🔗 🖼️

Точно е заокружено

Објаснување: Кога серверот успешно вrne нешто има статус код 200, а и ц автоматски отпаѓаат. Под е) е грешно оти ContentType не е параметар туку е заглавје. Под б) е грешно оти клиентот ги детектира типот на податоците во заглавјето на одговорот т.е дали се html, xml, json итн.

Question 3

Not yet answered

Marked out of 1.00

Flag question

Кој додава HTML во одговорот?

Select one or more:

- ☐ a. Web Server
- ☐ b. Container
- ☒ c. Servlet

Give your reasons

↶ A ▾ B I ≡ ≡ 🔍 🔗 🖼️

Точно е заокружено

Servlet генерира html темплјет што го става во body-то на одговорот. Другите две не враќаат html.

4

ed  
out of  
question

Доколку сакате да се де-регистраirate за добивање на пораки на одреден **Publisher** на податоци, во кој метод од сервлетот ќе го сторите тоа?

Select one:

- ☐ a. `init`
- ☐ b. Конструктор
- ☐ c. `service`
- ☒ d. `doPost`
- ☐ e. `doGet`
- ☐ f. `destroy`

Give your reasons

Точно е заокружено

`doPost()`: Се користи за работење со POST requests.

Логично кога размислеш ако сакаш да се де-регистраиш од нешто, треба да пратеш POST request до серверот, па во серверлетот ќе ја користеш `doPost` методата.

2

ed  
out of  
question

Кои од следните настани треба да се извршат за пред да може да се креираат Request и Response објектите кај една апликација во J2EE Container?

Select one or more:

- ☒ Креирање на Servlet Context
- ☐ Повикување на `service()` од сервлетите
- ☒ Поставување на параметри во Servlet Context
- ☒ Читање на `web.xml`
- ☒ Иницијализација на сервлетите

Give your reasons

Точно е заокружено

За разлика од другите второто не треба да се изврши пред да може да се креираат објектите за Req и Res.

Question 10

Поврзете ги дефинициите со концептите од Aspect Oriented Programming (AOP):

Место во извршувањето на апликацијата каде се вклучува аспектот

Дефинира која задача и кога ќе се изврши

Дефинира каде може да се изврши аспектот

Give your reasons

1. Join Point
2. Advice
3. Pointcut

Question 15

Нека во веб апликација е поставена следната конфигурацијана view-resolver:

```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="prefix" value="/WEB-INF/views/">
  <property name="suffix" value=".html">
</bean>
```

Koja ќе биде датотеката која ќе се избере од ViewResolver-от за приказ (view template) при повик на question() методот дефиниран во продолжение?

```
@Controller
@RequestMapping("/exam")
public class ExamController {

    @RequestMapping(value = "/question", method = RequestMethod.POST)
    public String question() {
        return "view";
    }
}
```

Select one:

☐ a. src/main/resources/views/question.html

☐ b. Нема да се селектира ниту една датотека, ќе се изврши редирекција

☐ c. src/main/resources/views/view.html

☒ d. src/main/resources/WEB-INF/views/view.html

☐ e. src/main/resources/WEB-INF/views/question.html

☐ f. Нема да се селектира ниту една датотека, ќе се генерира JSON

Give your reasons

Точно е заокружено

Гледаш ги горе префиксите и што методата враќа па ги спојуваш заедно во една релативна патека.

**БИТНО!** Задачата подоле се повторува 300 милијарди илјади стотици и 5 пати. Гледај да научеш објаснетата постапка, а не на памет. - мацо

13

Доколку имаме потреба кон клиентот да вратиме Header-от "user" во рамките на малиран метод во Spring MVC, во кој од следните методи е можно да го поставиме?

Select one or more:

☐ a.

```
@Controller
@RequestMapping("/exam")
public class ExamController {

    @RequestMapping("question")
    public String question(HttpSession session, ServletContext context) {
        ...
    }
}
```

☐ b.

```
@Controller
@RequestMapping("/exam")
public class ExamController {

    @RequestMapping("question")
    public String question(@Header String session) {
        ...
    }
}
```

☐ c.

```
@Controller
@RequestMapping("/exam")
public class ExamController {

    @RequestMapping("question")
    public String question(HttpServletRequest request) {
        ...
    }
}
```

☒ d.

```
@Controller
@RequestMapping("/exam")
public class ExamController {

    @RequestMapping("question")
    public String question(@RequestHeader(name="user") String user) {
        ...
    }
}
```

☐ e.

```
@Controller
@RequestMapping("/exam")
public class ExamController {

    @RequestMapping("question")
    public String question(HttpServletResponse context) {
        ...
    }
}
```

Give your reasons

1 A B I [ ] [ ] [ ] [ ] [ ]

Точно е заокружено

- а) никаде во методот нема за хеадер
- б) треба да вратиме хеадер со името user
- ц) Не работеме со сервлети мај дудс
- д) тру енд рил одговор 👍
- е) Ит ис еј сервлет метод мај дудс

**БИТНО!** Задачата подале се повторува 300 милијарди илјади стотици и 5 пати. Гледај да научеш објаснетата постапка, а не на памет. - мацо

14  
id  
out of  
question

Доколку сакаме барањето кон нашиот сервис да биде  
GET http://localhost:8080/exam/answer/12?answer=xyz  
и истото треба да го имплементираме во контролерот:  

```
@Controller("exam")  
@RequestMapping("/exam")  
public class ExamController {  
  
}
```

Кое од следните е валидна дефиниција на метод кој ќе се справи со ова барање?

Select one:

☐ a.  

```
@RequestMapping(value = "/exam/answer/{id}?answer={answer}", method = RequestMethod.GET)  
public String question(@RequestParam Long id, @RequestParam String answer) {  
...  
}
```

☐ b.  

```
@RequestMapping(value = "/answer/{id}?answer={answer}", method = RequestMethod.GET, produces = "application/json")  
public String question(@RequestParam Long id, @RequestParam String answer) {  
...  
}
```

☐ c.  

```
@RequestMapping(value = "/answer/{id}?{answer}", method = RequestMethod.GET, produces = "application/json")  
public String question(@PathVariable Long id, @PathVariable String answer) {  
...  
}
```


☐ d.  

```
@RequestMapping(value = "/exam/answer/{id}", method = RequestMethod.GET, produces = "application/json")  
public String question(@PathVariable Long id, @RequestParam String answer) {  
...  
}
```

☒ e.  

```
@RequestMapping(value = "/answer/{id}", method = RequestMethod.GET, produces = "application/json")  
public String question(@PathVariable Long id, @RequestParam String answer) {  
...  
}
```

Give your reasons



Точно е заокружено

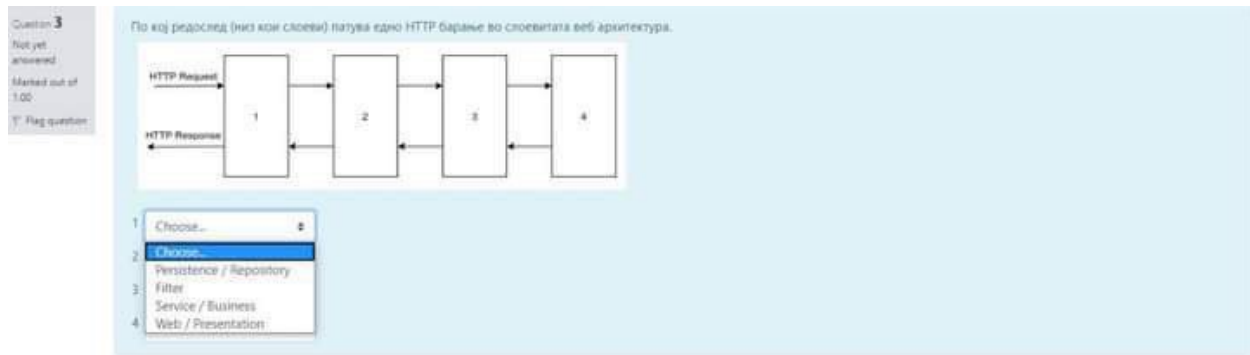
Најбитно да се гледа е @RequestMapping аотацијата.

Прво гледаме во ExamController таму е /exam значи доле сите што имат /exam/answer...

одма испаѓат бидејќи мапирањето ќе испадне /exam/exam/answer т.е а) и д) испаѓат.

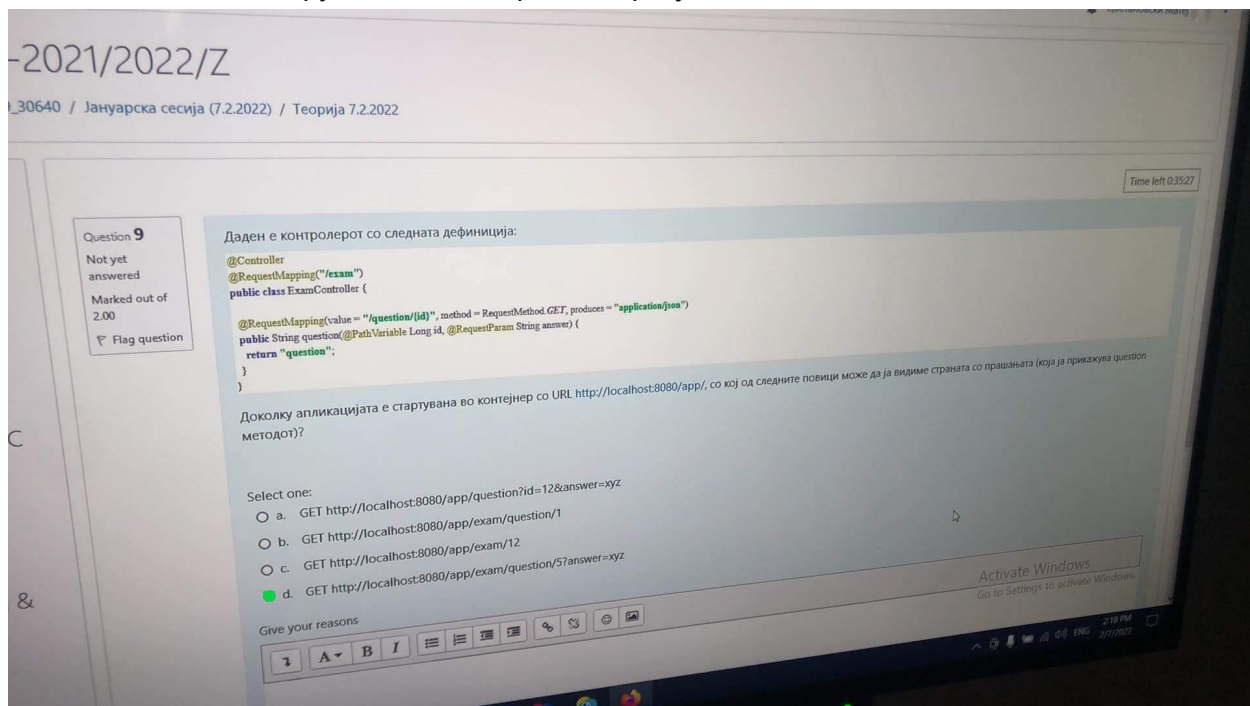
После /answer/ гледаме дека имаме ид бројка па request параметар. Овде е битно да се знае дека ид се става во заграда а request параметрата се става во аргументите на методот. Според ова б) и ц) испаѓат бидејќи answer е request параметра а е ставена во мапирањето наместо аргумент во методата.

Под е) е точниот одговор.



- 1) Filter, прво request-от се соочува со филтрите во апликација.
- 2) Web, после се испраќа кон веб слојот каде што се контролерите
- 3) Service, од веб слојот се испраќа тука каде што е бизнис логиката на апликацијата
- 4) Persistence, на крај се праќа во репозитори слојот каде што податоците се персистнуват и флашнуват во база.

Гледаш дека се повторува? Само пократка верзија. **НАУЧИ ПОСТАПКАТА!!!** 😡



Точно е заокружено

Исто како погоре што објаснив гледаш ја `@RequestMapping` анотацијата и ја следеш во дадениот URI.

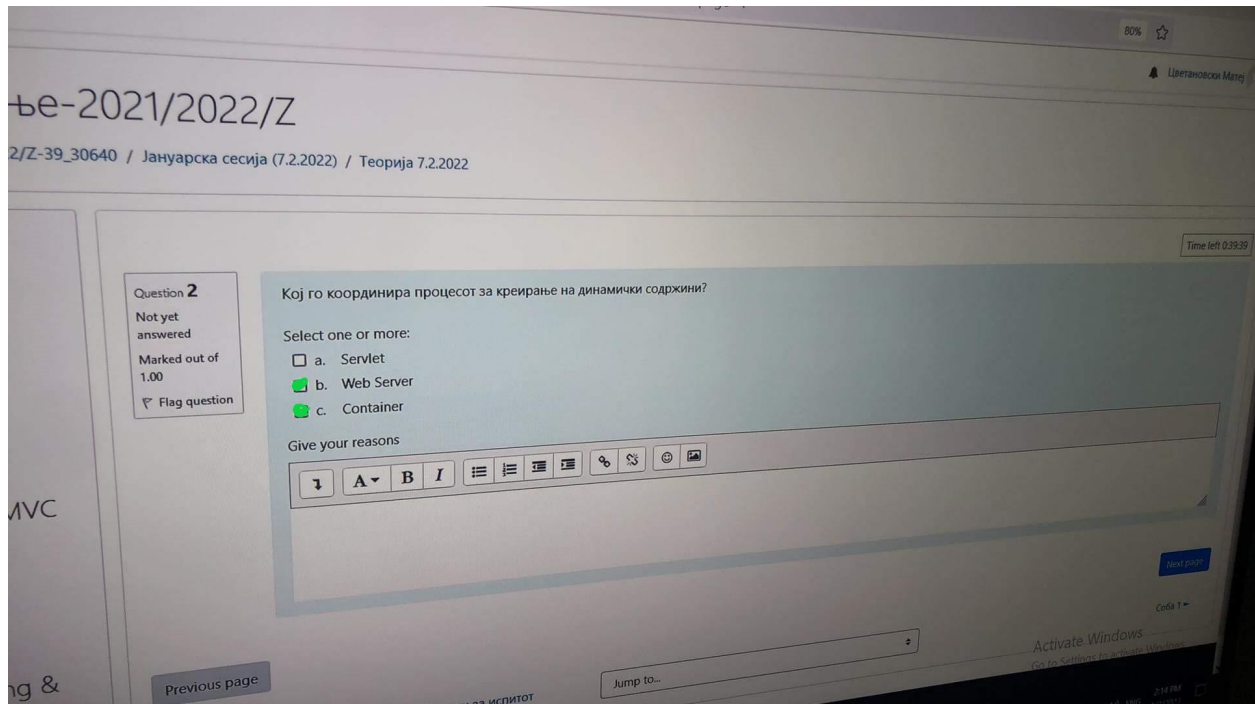
Од `/localhost:8080/app` -> додаваме `/exam` од контролерот

Од `/localhost:8080/app/exam` -> додаваме `/question/{id}` од методата

Од `/localhost:8080/app/exam/question/{id}` -> додаваме request параметрите со име `answer`

Краен URI треба да е `/localhost:8080/app/exam/question/{id}?answer=xyz`

Под д) е точниот одговор.



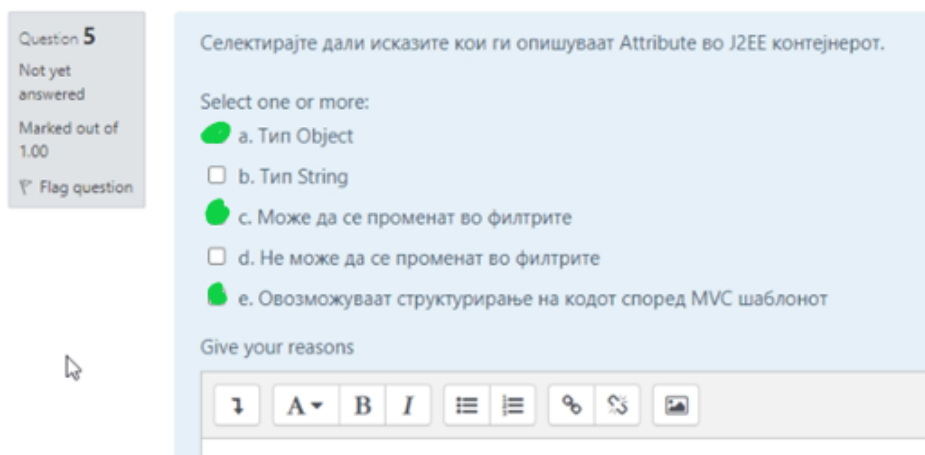
Не е точно заокружено

а) Сервлетот генерира динамични содржини не го води процесот на координација.

б) Web Serverот сервира статични содржини.

ц) **Координаторот** на динамични содржини е Сервлет контејнерот или контејнерот во ова прашање.

Само под ц) е.



Точно е заокружено

За разлика од параметрите, атрибутите се податоци кои може да се уредуваат една штом се постават. Нивниот тип не е ограничен само на текст, туку можат да бидат било кој тип кој може да се серијализира. Тие не се поставуваат Дескрипторот на поставување, туку во самиот код на менаџираните компоненти.

6

ed  
out of  
question

Доколку во ServletA повикаме `dispatcherToServletB.forward(req, resp)`, што од следното е точно:

Select one or more:

- ☒ a. Во ServletB можеме да ги пристапиме параметрите пратени до ServletA
- ☐ b. Контејнерот ќе го повика сервлетот мапиран на `servletB.do`
- ☐ c. Клиентот ќе направи ново барање до `/servletB.do`
- ☒ d. Во ServletB можеме да ги пристапиме Session атрибутите поставени во ServletA
- ☒ e. Во ServletB можеме да ги пристапиме Request атрибутите поставени во ServletA

Give your reasons

Не е точно заокружено

Кога користиме `RequestDispatcher.forward`, контејнерот ќе го пренесе барањето кон ServletB и ќе го повика методот `doGet` или `doPost` на ServletB во зависност од типот на барањето.

Само под ц) е **неточно**, другите се **сите** точни!

7

ed  
out of  
question

Во кои од следните сценарија логично е атрибутите да се чуваат во Request?

Select one or more:

- ☐ a. Јазикот на корисникот
- ☐ b. Контакт информации за страната
- ☐ c. Резултати од пребарување
- ☐ d. Поддржани валути за плаќање
- ☐ e. ID на тековната сесија

Give your reasons

**НЕ СУМ СИГУРЕН ЗА ОВА ПРАШАЊЕ!**

Атрибутите кои се чуваат во Request треба да бидат оние кои се користат за да се изврши барањето. Следователно, атрибутите кои се чуваат во Request може да бидат:

**б) Контакт информации за страната** - за да се обезбеди начин за контакт со корисникот.

**ц) Резултати од пребарување** - за да се чуваат резултатите од пребарувањето.

**д) Поддржани валути за плаќање** - за да се обезбеди поддршка за различни валути.

**е) ID на тековната сесија** - за да се идентификува тековната сесија на корисникот.

Јазикот на корисникот не е атрибут



9

ed  
out of  
question

Доколку сакаме да креираме bean од кој ќе се креира само еднаш и оваа инстанца ќе се користи секаде каде што треба, тогаш треба да го конфигурираме:

Select one:

- ☐ a. scope="prototype"
- ☐ b. scope="context"
- ☐ c. scope="instance"
- ☒ d. scope="singleton"

Give your reasons

1 A B I [ ] [ ] [ ] [ ] [ ]

Точно е заокружено

@Scope("singleton"): Ова е стандардниот животен век на биновите во Spring. Кога се користи овој животен век, Spring контејнерот креира **само една инстанца** од бинот и ја користи за сите последователни барања за тој бин преку ctx.getBean("singletonName").

11

ed  
out of  
question

Во кој од подолу предложените слоеви е најлогично да биде следната имплементација:

```
public void employeesCards(List<Employee> employees, HttpServletResponse response)
    throws DocumentException, IOException, SQLException {

    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    IDMultiCardsReport report = new IDMultiCardsReport("id_cards_multi", baos, context, employees);
    report.generate();
    new ResponseWriter(response).writePdf(baos, "id_cards");
}
```

Select one:

- ☐ a. Domain Model
- ☐ b. Persistence
- ☒ c. Presentation
- ☐ d. Service
- ☐ e. User Interface

Give your reasons

1 A B I [ ] [ ] [ ] [ ] [ ]

Точно е заокружено

Овој слој се однесува на тоа како информацијата се прикажува на корисниците. Во овој контекст, генерирањето на PDF документи (видливи документи) и одговор на HTTP захтеви конкретно се дел од начинот на прикажување на информацијата кон клиентите (корисниците).

**БИТНО!** Задачата подоле се повторува 300 милијарди илјади стотици и 5 пати. Гледај да научеш објаснетата постапка, а не на памет. - мацо

Question 3

Not yet answered

Marked out of 25.00

Flag question

Time left 0:23:14

Даден е кодот од две класи и од web.config:

```

@WebServlet(name = "Servlet1", value = "/d1",
    initParams = {
        @WebInitParam(name = "day", value = "200"),
    })
public class D_Servlet1 extends HttpServlet{

    public void doGet(HttpServletRequest request, HttpServletResponse response){
        // ги враќа вредностите на count, day, page, size
    }
    ...
}

@WebServlet(name = "Servlet2", value = "/d2",
    initParams = {
        @WebInitParam(name = "day", value = "100"),
    })
public class D_Servlet2 extends HttpServlet{

    public void doGet(HttpServletRequest request, HttpServletResponse response){
        // ја инкрементира вредноста на page за 10 и на size за 5
        // ги враќа вредностите на count, day, page, size
    }
    ...
}

```

Која од наведените URL адреси е валидна за повик на doGet метод од Servlet2?

Нека config=getServletConfig(), context=getServletContext() и session=getSession(). Нека во некој од методите се креира сесиски атрибут size = 10 и нека во contextInitialized() се иницира атрибут page = 20 кој се зачувува во context. Кои од методите се ќе се изврши без грешка?

☐ session.setAttribute("size", 100)
 ☐ context.setAttribute("page", new Date())
 ☐ context.setAttribute("count", 10)
 ☐ session.getAttribute("size")
 ☐ config.setAttribute("day", 100)

Нека config=getServletConfig(), context=getServletContext() и session=getSession(). Нека во некој од методите се креира сесиски атрибут size = 10 и нека во contextInitialized() се иницира атрибут page = 20 кој се зачувува во context. Кои од методите се ќе се изврши без грешка?

☐ session.setAttribute("size", 100)
 ☐ context.setAttribute("page", new Date())
 ☐ context.setAttribute("count", 10)
 ☐ session.getAttribute("size")
 ☐ config.setAttribute("day", 100)
 ☐ config.getInitParameter("day")
 ☐ (Integer)session.getAttribute("size")

Внесете ги вредностите кои му се праќаат на клиент 1 во последниот повик ако клиент 2 ги повика GET /d2, GET /d1, GET /d2, а потоа клиент 1 го повика GET /d1, GET /d2.

count , size , day  и page

Give your reasons

Time left 0:23:12

1. Која од наведените URL адреси е валидна за повик на doGet метод од Servlet2
  - Гледаш ја методата од десно на првиот скриншот, да таа што е со name = "Servlet2" колега/шка.
  - Value е тоа што треба да се спое со адресата за да се повика тој сервлет
  - Бидејќи ја нема dropdown листата на скриншотот точниот одговор треба да е валидна url адреса што го има value /d2
  - Може да биде /d2, http://localhost:8080/shoznam/d2, сфаќаш? **Сфаќаш!** 🎉

2. Овде мора да ги знаеш методите и нивната синтакса. Нема ништо друго посебно. Едино треба да пазеш кои променливи што вршат.

TLDR: **session=getSession()**, ја зима тековната сесијата, можеш да означуваш и да зимаш атрибути, **context=getContext()**, го зима тековниот контекст, можеш да означуваш и да зимаш атрибути, **config.getServletConfig()** ја зимаш конфигурацијата, не постоје **setAttribute** и **getAttribute** методи во оваа класа.

- `session.setAttribute("size", 100)` -> точно
- `context.setAttribute("page", new Date())` -> точно
- `context.setAttribute("count", 10)` -> точно
- `session.getAttribute("size")` -> **не е точно** мора да се кастира пред да се врате
- `config.setAttribute("day", 100)` -> **не е точно**
- `config.getInitParameter("day")` -> точно
- `(Integer)session.getAttribute("size")` -> точно

Само `config.setAttribute("day", 100)` и `session.getAttribute("size")` **имат грешка**

3. Сега треба да гледаш што прават методите.

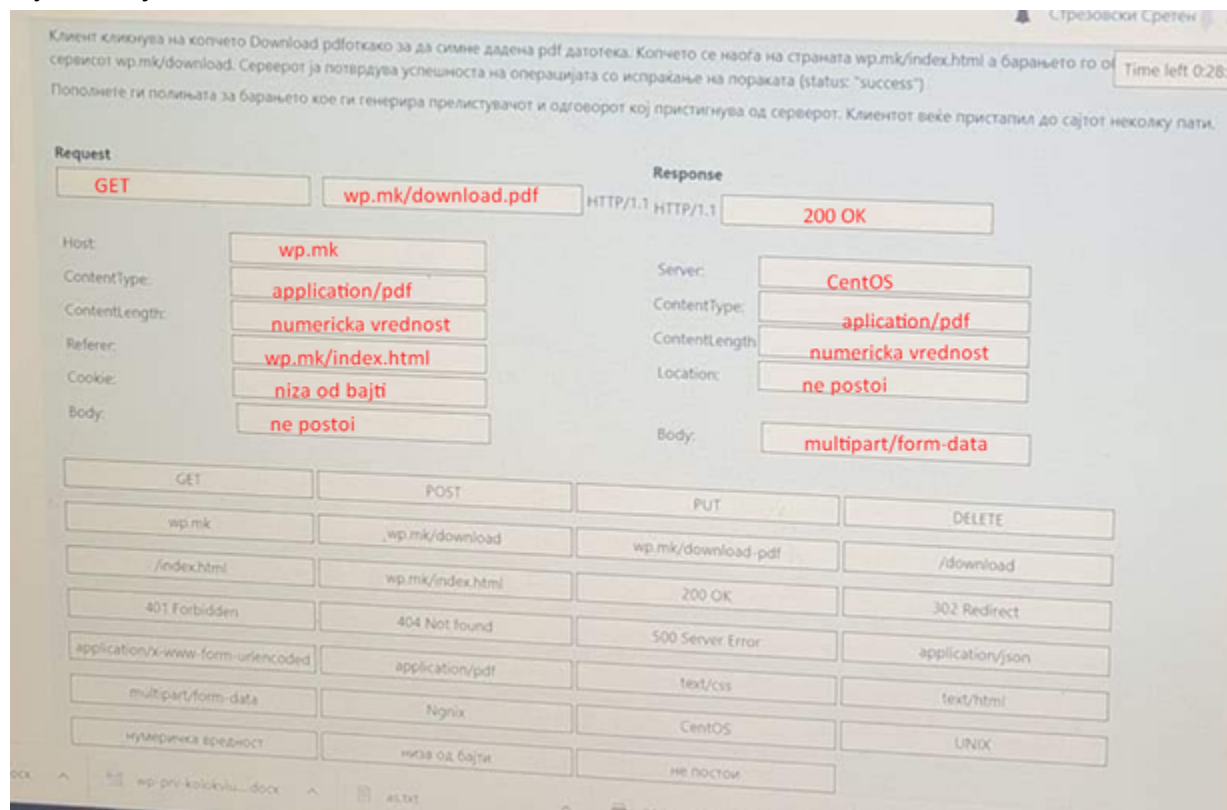
- **Count = ??** не кажува никаде што прави count, ако брое requests тогаш е **5**, ако се гледа во втората точка „`context.setAttribute("count", 10)`“ тогаш ќе е **10**, не знам што ќе е точниот одговор, па пробај си среќата 😊

**Size = 10+5 = 15**, според точка 2 size е сесиски атрибут тоа значи дека ќе е уникатен за клиент. Така што се што ќе повика клиент 2 не е битно и само гледаме што повикува клиент 1. Иницијално е 10 според точка 2 и еднаш се повикува /d2 значи се инкрементира еднаш за 5 според дадена doGet метода.

**Day = 100**, гледаш го `@WebInitParam(name="day", value=broj)` кај двете методи, после ги гледаш GET request-ите во точката, day цело време се иницијализира на 100 или 200 зависно во која метода се прате GET request-от, па само го гледаш последниот пратен, тоа е /d2 значи во него го иницијализираме да биде 100.

**Page = 20 + 10 + 10 + 10 = 50**, во точка 2 page=20 е иницијализирано, три пати се повикува /d2, а таа го инкрементира за 10. Page е **context**, а context се **глобално** споделува помеѓу сервлети за разлика од **сесиски** атрибут како size.

**БИТНО!** Задачата подоле се повторува 300 милијарди илјади стотици и 5 пати. Гледај да научеш објаснетата постапка, а не на памет. - мацо



Не се точно напишани

→ Request:

- ◆ **GET** /download, во текстот пишува дека се повикува ова URI
- ◆ **Host:** wp.mk, тоа е хостот нема друг
- ◆ **ContentLength:** не постои, **GET** и **DELETE** request-ови немат ContentLength, ако **не** се овие тогаш ќе е **нумерична вредност** точниот одговор
- ◆ **Content-Type:** не постои, во response ќе се врати, на сликата application/x-www-form-urlencoded се праќа често во **POST** и **PUT** request-ови
- ◆ **Body:** не постои, нема боди во **GET** request
- ◆ **Referer:** wp.mk/index.html
- ◆ **Cookie:** низа од бајти, ако пристапувавме прв пат одговорот ќе е **не постои**.

→ Response:

- ◆ **HTTP/1.1** 200 OK,
- ◆ **Server:** Nginx, Постојат Apache, Nginx, Microsoft IIS како сервери нема други. CentOS на сликата е оперативен систем **лол**
- ◆ **Content-Type:** application/pdf, симнува pdf, не css или html,
- ◆ **ContentLength:** нумеричка вредност, е сега враќа фајл. Тој фајл зафаќа место во меморија т.е бајти меморија.
- ◆ **Location:** не постои, само на redirect има локација
- ◆ **Body:** низа од бајти, нема да е multipart/form-data, бидејќи тоа не се става во response

**БИТНО!** Задачата подоле се повторува 300 милијарди илјади стотици и 5 пати. Гледај да научеш објаснетата постапка, а не на памет. - мацо

12

ed  
out of  
question

REST контролерот ConsultationSlotApi е мапиран на следната локација: `/api/consultations`. Апликацијата е стартувана на порта 8080. Кои од прикажаните методи успешно ќе го обработат PATCH барањето во следната форма:

```
PATCH http://localhost:8080/api/consultations/{slotId}
professorId: petko.petkov
Content-Type: application/x-www-form-urlencoded
```

```
roomName=117&dayOfWeek=2&from=18:00:00&to=20:00:00
```

Како опционален параметар во барањето, може да се прати и date којшто треба да се десеријализира во објект од типот LocalDate.

Select one or more:

☐ a.

```
@RequestMapping(value = "{pathId}", method = RequestMethod.PATCH)
public ConsultationSlot updateSlot(@PathVariable int slotId,
    @RequestHeader String professorId,
    @RequestParam String roomName,
    @RequestParam(value = "dayOfWeek") Integer dayOfWeek,
    @RequestParam(value = "date", required = false) @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate date,
    @RequestParam("from") @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime from,
    @RequestParam("to") @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime to) {
    return consultationSlotService.updateSlot(slotId, professorId, roomName, DayOfWeek.of(dayOfWeek), date, from, to);
}
```

☐ b.

```
@PatchMapping("/{slotId}")
public ConsultationSlot updateSlot(@PathVariable int slotId,
    @RequestHeader String professorId,
    @RequestParam String roomName,
    @RequestParam(value = "dayOfWeek") Integer dayOfWeek,
    @RequestParam(value = "date") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate date,
    @RequestParam("from") @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime from,
    @RequestParam("to") @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime to) {
    return consultationSlotService.updateSlot(slotId, professorId, roomName, DayOfWeek.of(dayOfWeek), date, from, to);
}
```

☒ c.

```
@PatchMapping("/{slotId}")
public ConsultationSlot updateSlot(@PathVariable int slotId,
    @RequestHeader String professorId,
    @RequestParam String roomName,
    @RequestParam(value = "dayOfWeek", required = false) Integer dayOfWeek,
    @RequestParam(value = "date", required = false) @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate date,
    @RequestParam("from") @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime from,
    @RequestParam("to") @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime to) {
    return consultationSlotService.updateSlot(slotId, professorId, roomName, DayOfWeek.of(dayOfWeek), date, from, to);
}
```

☐ d.

```
@PatchMapping("/{slotId}")
public ConsultationSlot updateSlot(@RequestParam int slotId,
    @RequestHeader String professorId,
    @RequestParam String roomName,
    @RequestParam(value = "dayOfWeek", required = true) Integer dayOfWeek,
    @RequestParam(value = "date", required = false) @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate date,
    @RequestParam("from") @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime from,
    @RequestParam("to") @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime to) {
    return consultationSlotService.updateSlot(slotId, professorId, roomName, DayOfWeek.of(dayOfWeek), date, from, to);
}
```

Give your reasons

Точно е заокружено

а) **Не**, поради pathId во @RequestMapping(), треба ни slotId

б) **Не**, датата треба да е опционална а чим не е специфицирано ставено е true по default

ц) **Да**, изгледа ми океј

д) **Не**, RequestParam int slotId, треба да е **PathVariable!**

Дадена веб апликација е стартувана на `localhost:8080`, контролер класата во која се наоѓа овој метод е анотирана со `@RequestMapping("/api/events")`

Time left 0:18:45

```
@RequestMapping(method = RequestMethod.POST)
public Event createEvent(@RequestHeader String creatorId,
    @RequestParam String location,
    @RequestParam(value = "day_of_week", required = false) Integer dayOfWeek,
    @RequestParam(required = false) @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate date,
    @RequestParam("start") @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime from,
    @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.TIME) LocalTime to) {

    Event result = service.createEvent(creatorId, location, DayOfWeek.of(dayOfWeek), date, from, to);
    return result;
}
```

Пополнете го `http` барањето со кое ќе се повика методот од погоре:

The diagram illustrates an HTTP POST request to the endpoint `/api/events` using the `HTTP/1.1` protocol. The request body is encoded as `application/x-www-form-urlencoded` and contains the following data:

- `creatorId`: 345
- `location`: 1ab8
- `day_of_week`: 28
- `from`: 18:00:00
- `to`: 20:00:00

Како резултат од ова барње ќе се прикаже страницата `createEvent`.html и во моделот ќе се постави својство со име `createEvent` и вредност `result`.

GET		PUT	DELETE	
localhost/api/events	http://localhost:8080	http://localhost:8080/api/events	localhost:8080/api/events	
creator_id		roomName	dayOfWeek	
date	event_date	from		
end		multipart/form-data	text/html	text/css
		content-type	contentType	ContentLength
content-length	contentLength		event	Event

Не се точно напишани

1. POST /api/events -> **точно**, методата е POST
2. creatorId -> **не е точно**, ова е ContentLength
3. ContentType: application/x-www-form-urlencoded -> **точно**, multipart/form-data се користи во по комплицирани request-ови каде што има и фајлови, а во овај има само дати, интеџер, стринг.
4. location, day\_of\_week?, from, to -> **точно**, гледаме ги според RequestParam тука пазете на value вредноста и на името
5. Тука не знам што треба оти никаде не гледам каде пишува какво е името на темплјетот што ќе се генерира, а на сликата тоа што е пишано е сигурно грешно.

Question 2

Not yet answered

Marked out of 10.00

Flag question

Time left 0:23:43

Веднаш по поставувањето на апликацијата составена од servlet1, servlet 2 и servlet 3, корисникот испраќа **post** барање кои го обработува **servlet2**, а потоа две **get** барања кои го обработува **servlet 1**. Потоа администраторот ја прекинува апликацијата (undeploy).

Подредете ги настаните по редослед на извршување.

Забелешка: Ако ви останат вишок полиња, внесете ознака „празно“. Ако недостасуваат полиња, внесете ги само првите настани за кои има место.

container -> load servlet 2class			
servlet 2 -init()			
servlet 2 -> doPost()			
container -> load servlet 1			
servlet 1 -> init()			
servlet1 -> doGet()			
Servlet 1 -> doGet()			
Servlet 1 -> destroy()			
Servlet 2 -> destroy()	container -> load servlet 2 class	container -> load servlet 3 class	servlet 1 -> init()
servlet 2 -> init()	servlet 3 -> init()	servlet 1 -> doGet()	servlet 1 -> doPost()
servlet 2 -> doGet()	servlet 2 -> doPost()	servlet 3 -> doGet()	servlet 3 -> doPost()
servlet 1 -> destroy()	servlet 2 -> destroy()	servlet 3 -> destroy()	празно

Give your reasons

Точно е напишано

Прво испраќаме **POST** барање до сервлет2, тоа значи контејнерот ќе го лоадира, па ќе се иницијализира пред да направи **doPost**.

Потоа испраќаме 2 **GET** барања до сервлет1, пак се лоадира контејнерот, се иницијализира и на крај 2 **doGet** методи.

Кога ќе се прекине апликацијата **destroy** се повикува рекурзивно, тоа значи по обратен редослед ќе се гасат т.е прво сервлет1 па сервлет2.

Гуд лак хав фан колега/шка. Толку од мене, ако паднеш не сум **крив** 🤪