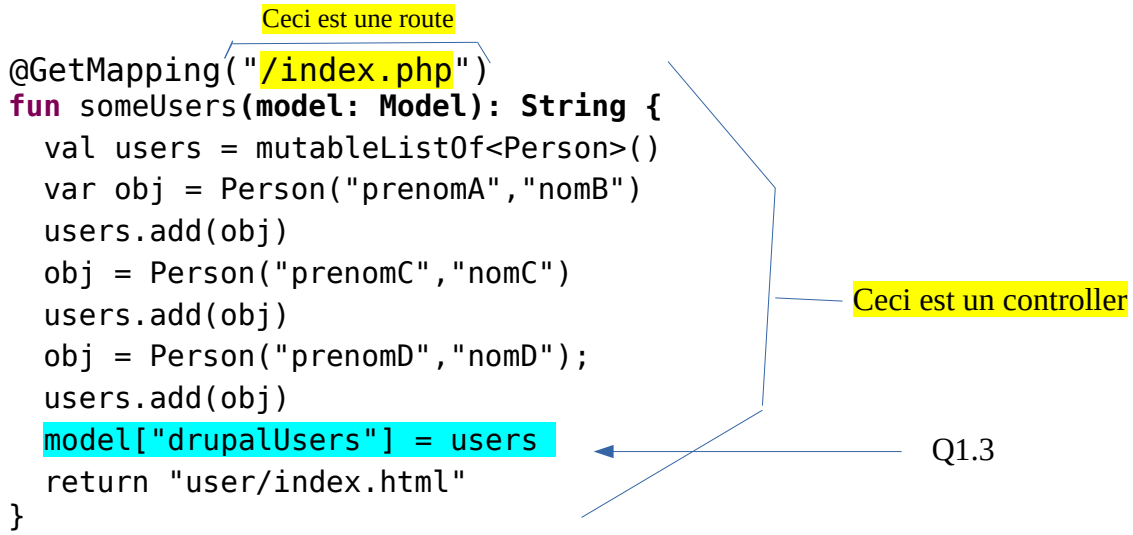
CONTRÔLE DES CONNAISSANCES N° 3 Mercredi 16 novembre 2022
DURÉE : 3/4h

PARTIE 1 - MVC Web - **33 points**

Q1. - Voici un extrait de code (un contrôleur) de la classe `UserController :`



Q1.1 Combien d'éléments contient la liste référencée par users ? (2 points)

User est une liste avec trois (3) users.add.

Q1.2 Où le code de la vue est-il situé dans l'arborescence du projet ? (2 points)

La vue se situe dans le dossier resources/templates

Q1.3 D'après la documentation technique de Thymeleaf ci-dessous, donner l'instruction côté vue (`user/index.html`) qui donne le nombre d'utilisateurs transmis par le contrôleur (2 points)

Documentation Thymeleaf

#lists : utility methods for lists

Exemple dans une vue Thymleaf qui donne la taille d'une liste référencée par la variable aList

```
${#lists.size(aList)}
```

On passe des choses à la vue grace à au model (MVC) dans le contrôleur). La liste des utilisateurs sera donc accessible dans la vue via l'identifiant nommé **drupalUsers.**

```
${#lists.size(drupalUsers)}
```

Q2. - Un développeur de votre équipe découvre la programmation web. Il souhaite solliciter l'url suivante : `http://localhost:8080/demo/bonjour`

Mais reçoit le message suivant :

Q2.a Expliquez la raison **précise (<u>ne pas répéter la cause de l'erreur</u>)** de cette erreur. **(2 points)**

Il n'y a pas de contrôleur lié à cette route par la méthode GET d'une requête HTTP

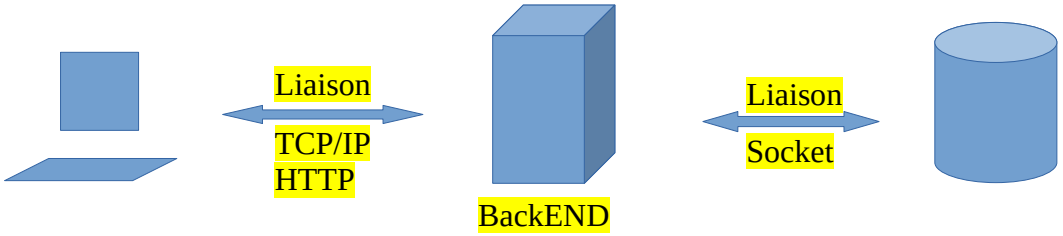
Q2.b Concevoir le code d'un contrôleur (le plus simple possible) qui éviterait cette erreur**.**
(3 points)

```
@GetMapping("/demo/bonjour")
@ResponseBody
fun demo(): String {
   "bonjour"
}
```

@GetMapping désigne la route lié à la méthode
@ResponseBody signifie que le contrôleur ne passe pas par une vue pour répondre (pour simplifier la réponse à cette question)

Q3. Question d'architecture - (3 points)

Q3.1 A partir de cette architecture 3 tiers Web, placer sous chacun des tiers les termes techniques proposés dans la liste ci-dessous, si vous considérez le ou les concepts, derrière ces termes, sont actifs (comprendre exécutés) sur le tiers en question.



Client Web Navigateur	**serveur HTTP**	**SGBDR**
HTML CSS HTTP JavaScript	Java SpringBoot Kotlin HTTP Thymeleaf (`JavaScript avec nodejs`)	SQL

Liste de termes : JavaScript, HTML, HTTP, SQL, Kotlin, Java, CSS, Thymeleaf, Spring Boot
Remarque : une techno peut être active sur plus d'un tiers

Q4 Dans cet extrait de code **(3 points)**

```
@GetMapping("/hello")
fun index(model: Model, request: HttpServletRequest): String {
   val nom: String = request.getParameter("nom") ?: ""
   model["name"] = nom
   return "index/hello"
}
```

index/hello est la vue

PRG `"(redirect:index)"`

Code 3xxx

La route

[X] `@GetMapping("/hello")` est placé obligatoirement avant le nom de la méthode

[X] `@GetMapping("/hello")` est une façon de lier une *route* à un *contrôleur*

[X] `@GetMapping("/hello")` est une façon de lier une requête HTTP à un *contrôleur*

[X] le mot clé return est facultatif en dernière ligne du corps de la méthode

[ ] l'instruction `model["name"] = nom` provoque une erreur de compilation

`Une route désigne la ressource d'une requête HTTP, dans le contexte du code source`

Q5 Dans cet extrait : **(2 points)**

```
@GetMapping("/blog/{slug}")
public function showAction(@PathVariable slug : String): String { … }
```

[X] le paramètre *slug* de la méthode doit avoir le même nom que la partie variable de la route
[ ] le paramètre *slug* de la méthode n'a rien a voir avec les données de l'URL de la route
[X] le paramètre *slug* de la méthode sera automatiquement renseigné par le framework

Il faut que l'argument passe par le controller principal du framework (springboot)

Q6 Soit l'extrait suivant : **(2 points)**

```
@GetMapping("/blog")
fun showAction(@RequestParam(defaultValue = 3) page: Int) : String { … }
 {
    // ...
 }
```

Le type de page

[X] le paramètre *$page* a comme valeur par défaut : 3
[ ] le paramètre *$page* a comme valeur par défaut : "page"
[X] la route */blog?page=0* provoquera l'exécution de la méthode *showAction* avec 0 en argument
[X] la route */blog?page=3* provoquera l'exécution de la méthode *showAction* avec 3 en argument
[ ] la route */blog?page=un* provoquera l'exécution de la méthode *showAction* avec '*un*' en argument

Le "un" de la réponse 5 n'est pas un entier !

Q7 Soit l' instruction : **(2 points)**

```
return "produit/show"
```

Dans une méthode contrôleur, approche MVC, une telle instruction :

[X] signe la fin de l'exécution du corps de la méthode
[X] prend la responsabilité de désigner la vue à appliquer suite à une requête HTTP
[ ] retourne la valeur `"produit/show"` à une vue par défaut
[ ] retourne la valeur `"produit/show"` directement au client HTTP

Le rôle "return" fait 2 choses : retourne une valeur compatible avec le type de défintion de la fonction et permet de quitter immédiatement l'exécution du corps de la méthode

Q8 Avec Thymeleaf je peux logiquement **(2 points)**

[X] générer du HTML
[X] écrire du code java
[X] générer du code CSS
[X] écrire du contenu sous condition alternative
[ ] afficher la valeur de la variable xx avec l'instruction `{{ xx }}`
[X] afficher la valeur de la variable xx avec l'instruction `${ xx }`
[ ] afficher la valeur de la variable xx avec l'instruction `$${ xx }`

Q9 Un message *flash* est **(2 points)**

[ ] un message de notification à destination du serveur
[X] un message de notification à destination du client
[ ] un message à usage unique initié côté serveur
[X] un message à usage unique initié côté client
[X] un message au cycle de vie court (le temps d'un affichage)
[ ] un message au cycle de vie moyen (le temps d'une session utilisateur)

`Voir le redirectAttributes : ↓`

https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/servlet/mvc/support/RedirectAttributes.html

Q10 Une instance de *HttpServletRequest* , me permet **(2 points)**

[X] de récupérer les données d'entrée transmises en GET
[X] de récupérer les données d'entrée transmises en POST
[X] d'accéder aux données de la session du client HTTP(s'il y en a)
[X] d'ajouter des données à la session du client HTTP
[X] d'obtenir des informations sur le client à l'origine de la requête HTTP

Q11 Du point de vue du client HTTP, une « page web » est accessible : **(2 points)**

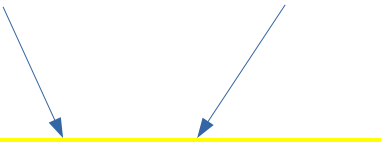
[X] via une *url*
[ ] via un contrôleur principal
[ ] via une vue *thymeleaf*
[ ] via une méthode d'une classe contrôleur

Du point de vue client, on ne peut pas savoir de quelle technologie il s'agit. Voir la route de la question Q1 pour comprendre ce que cela veut dire à partir d'un exemple.

Q15 Dans un fichier *thymeleaf*, une telle commande `${ title }` est l'équivalent : **(2 points)**

[X] d'un *echo* en php
[ ] de la création d'une balise *title* HTML
[X] de l'insertion, en lieu et place de l'instruction, de la valeur détenue par la variable *title*

```
echo $title  ≡  ${title}
```

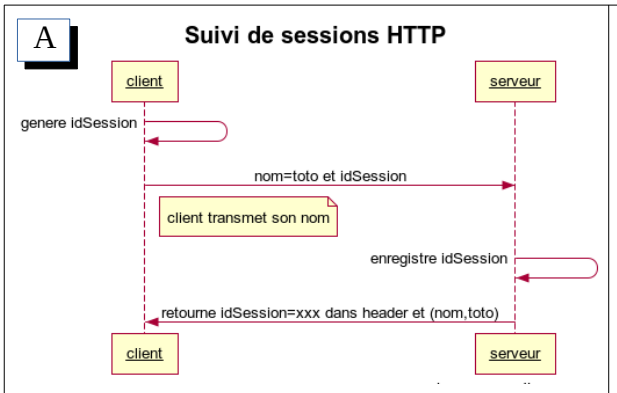


Remplacer par la valeur de la variable title

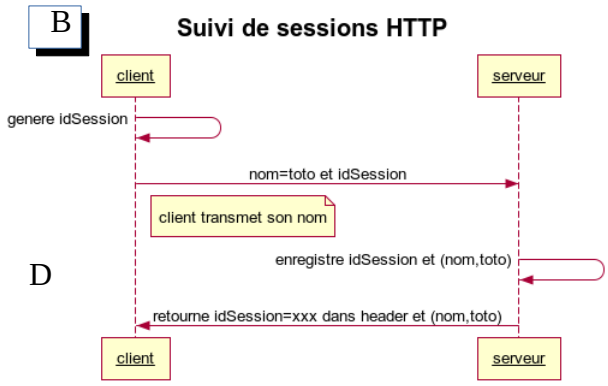
PARTIE 2 – HTTP (6 points)

Q16 Scénario : le client HTTP transmet son nom au système qui le reconnaîtra lors d'une prochaine et proche interaction HTTP.
Déterminez le, ou les, diagramme(s) de séquences qui représente(nt) le mieux la mise en place typique d'un suivi de sessions HTTP par id de session.

A

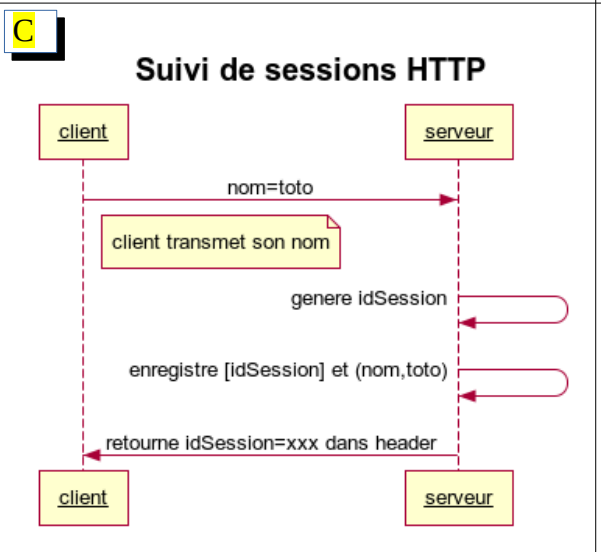


B

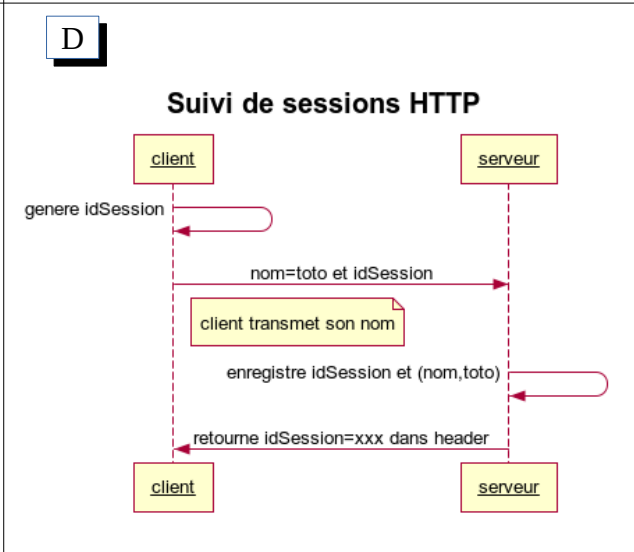


D

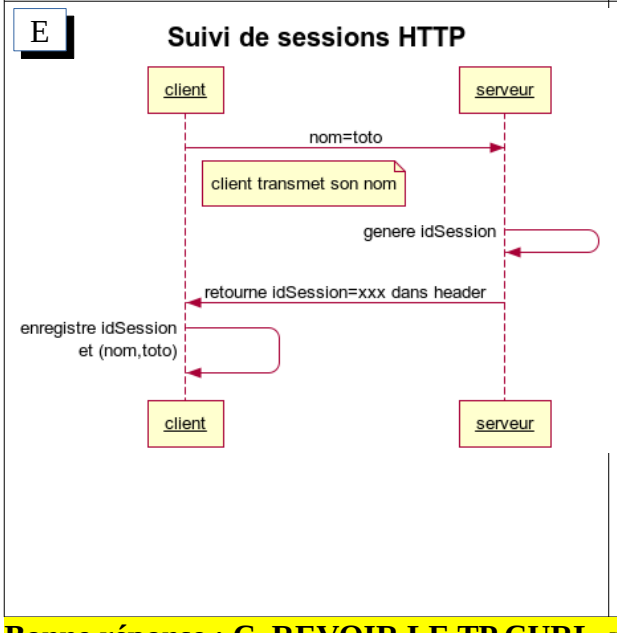
C



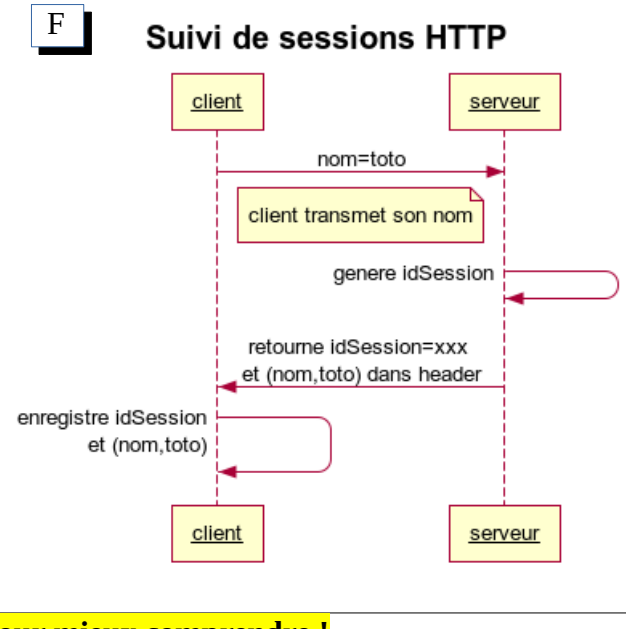
D



E



F



Bonne réponse : C, REVOIR LE TP CURL pour mieux comprendre !

PARTIE 3 – programmation facile (9 points)

Voici un extrait de corps d'une méthode de test : l'objet à tester est de type `Truc` référencé par la variable `obj` :

```
val s: String = obj.m()
assertEquals("sio", s)
```

Q17 D'après le code ci-dessus, déterminer le type de la méthode *m* (2 points)

[ ] Unit -> Unit
[ ] Unit -> Boolean
[ ] Boolean -> Unit
[ ] Truc -> Unit
[ ] Truc -> Boolean
[ ] String -> Unit
[X] Truc -> String
[ ] String -> Truc
[ ] String -> Boolean

Q18 Quel est l'objectif de ce test ? (**3 points**)

Test que l'appel de la méthode m à partir de l'objet de type Truc, référencé par obj, retourne bien la valeur "sio"

Dis autrement (Sekou Traore) :

Test que l'appel de obj.m() retourne bien la valeur "sio"

Et non pas : *vérifie que s est égal à "sio" (no matter variable name) s n'est pas l'objectif du test.*

Q19 En vous aidant de l'extrait de l'API JUnit donné, réécrire ce code en utilisant la méthode `assertTrue` au lieu de `assertEquals`. (**4 points**)

API (extrait) de JUnit

```
assertEquals( expected: String, actual: String )
  // Asserts that two objects are equal.

assertTrue( expressionBool: Boolean )
  // Asserts that expressionBool is true
```

```
val s: String = obj,m()
AssertTrue (s =="sio")
```

s == "sio" en est une expression booléenne, si "sio" est égal à la valeur de s alors l'expression vaut True sinon vaut False