TD4

October 2019

Ce TD s’appuie sur le cours intitul´e Framework Play! d’ Alexandre Collignon et Fr´ed´eric Drouhin. En cas de soucis (notamment les erreurs de compilation), ce document apporte une aide pr´ecieuse. Il est disponible sur le moodle.

# Base d’une application

|  |
| --- |
| **Objectifs :**   * G´erer les redirection ; * Maˆıtriser le CRUD. |

Ce TD s’appuie sur le dossier formulaire disponible sur le Moodle (section TD4). Vous vouvez changer la route comme suit pour ´eviter le message d’erreur

:

#GET / controllers.HomeController.index(fname : String) GET / controllers.HomeController.all()

## Comprendre le fonctionnement d’une redirection

### Utilit´e

La redirection permet d’acc´eder `a ces quatre actions :

**Ajout d’un ´el´ement (Create)** L’on souhaite ajouter une personne. Il faut donc un lien qui pointe vers le formulaire d’enregistrement de cette personne. Ce lien est par exemple *routes.HomeController.sayhelloform*.

**Lecture de l’´element (Read)** Ensuite, il est n´ecessaire de pouvoir acc´eder aux informations de cette personne.

**Mise `a jour (Update)** Permet de modifier un des ´el´ements et de l’enregistrer.

**Suppression (Delete)** Il peut s’av´erer n´ecessaire de supprimer un ´el´ement.

### Exercices

|  |
| --- |
| public Result helloworldform(Http.Request request) { final Form<Person> pForm = personForm.bindFromRequest(  request) ;  Person a = pForm.get(); a.save();  return redirect(routes.HomeController.all()) ;  } |

1

2

3

4

5

6

Listing 1: Redirection

Que fait le code *Redirect* (0.5pt) ? et ou` est il situ´e (*dossier/fichier*) ?

**Le code Redirect redirige la requete vers la route**

**Il se situe dans le dossier HomeController**

## CRUD

Une fois la redirection comprise, ce TD am`ene `a ´etudier et impl´ementer chacune des parties du CRUD. Ces parties seront li´ees par des redirections. Le fichier **formulaire** avec un programme basique sert de base `a ce TD.

### Affichage de tous les ´el´ements

Tr`es pratique pour la suite, cette partie vise `a afficher la totalit´e des ´el´ements enregistr´es. Il est n´ecessaire pour impl´ementer cet outil ”all” de cr´eer une nouvelle action+vue+route.

**Action** Pour le faire fonctionner correctement, il est n´ecessaire de rajouter le code not´e *codeAction* et d´etaill´e ci-dessous.

|  |
| --- |
| public Result all(){  List<Person> liste = Person.find.all(); return ok(all.render(liste));  } |

1

2

3

4

Listing 2: codeAction

Dans quel dossier/fichier est rajout´e *codeAction* ? (0.5 points**): Dans le dossier HomeController**

**Vue** Pour le faire fonctionner correctement, il est n´ecessaire de rajouter le code not´e *codeV ue* et d´etaill´e ci-dessous.

Quefaitlecode

*CodeVue*

(

1

pt

)

?

Dansqueldossier/fichierestrajout´e

*codeVue*

?

(0.5

points

)

:

|  |
| --- |
| **Dans le dossier all.scala**  @(liste: List[Person])  @main("Liste des personnes") {  <Title>Liste des @liste.size() personnes</Title>  <ul> |

1

**Le code vue permet de modifier la page localhost en lui ajoutant un titre par ex**

|  |  |
| --- | --- |
| } | @for(p <- liste) {  <li>@p.firstname, @p.age years old</li>  } </ul> |

5

6

7

8

9

Listing 3: codeVue

**Route** Une ligne de code est aussi rajout´ee dans conf/routes R´e´ecrivez la route correspondante (0.5 points): **GET / controllers.HomeController.all**

*(Aide : cette route a d´ej`a ´et´e donn´ee au d´ebut du TD)*

**Mise en pratique** L’outil *All* doit bien suˆr ˆetre disponible et utilisable.

### Le CRUD

**Sauvegarde et affichage** La partie Create entraˆıne l’enregistrement des´el´ements.

il suffit de rajouter la ligne de Code *codeCreate*.

Indiquez le dossier/fichier correspondant (0.5 points): **Dans le dossier HomeController**

|  |
| --- |
| Person a = pForm.bindFromRequest().get() ; a.save() ; |

1

2

Listing 4: codeCreate

**Affichage d’un seul ´el´ement (Read)** Pour cela, il est n´ecessaire cette fois ici, d’impl´ementer dans les trois parties : action+vue+route

La **route** permettant de prendre en compte la valeur de l’ID (GET) est elle rajout´ee ?

|  |  |  |
| --- | --- | --- |
| GET | /show/:id | controllers.HomeController.show(id: Long) |

1

Listing 5: codeReadRoute

L’**action** correspondante (*codeReadAction*) est la suivante. Indiquez le dossier/fichier correspondant (0.5 points): **Dans le dossier HomeControlers**

|  |
| --- |
| public Result show(Long id) { Person p = Person.find.byId(id) ; return ok(show.render(p)) ;  } |

1

2

3

4

Listing 6: codeReadAction

Enfin, pour y acc´eder, une redirection est rajout´ee dans **vue** *show* (exemple ci-apr`es).

|  |
| --- |
| @(liste: List[Person])  @main("Liste des personnes") {  <Title>Liste des @liste.size() personnes</Title>  <a href ="@routes.HomeController.sayhelloform"> Add Person </a> <ul>  @for(p <- liste) { |

1

2

3

4

5

6

|  |
| --- |
| <li>  <a href ="@routes.HomeController.show(p.id)"> @p.firstname, @p.age years old </a> </li>  }  </ul>  } |

7

8

9

10

11

Listing 7: show.scala.html

Dans show.scal.html, indiquez le code de cette redirection (0.5 points): **<a href="@routes.HomeController.show(p.getId())">**

**Mise `a jour d’un ´el´ement** Le code ci-dessous *codeUpdateAction* permet la mise `a jour d’un formulaire pr´e-rempli, en remplac¸ant les ´el´ements manquants.

manquant1 (0.25 points):  **id**

manquant2 (0.25 points**): personne**

|  |
| --- |
| public Result update(Long id) {  Person p = Person.find.byId(/\*manquant1\*/) ;  Form</\*manquant2\*/> pForm = formFactory.form(/\*manquant2\*/. class) ;  pForm = pForm.fill(p); return ok(update.render(pForm, p)) ;  } |

1

2

3

4

5

6

Listing 8: codeUpdateAction

Lors de votre programme, une action de validation du formulaire est r´ealis´ee (indice : *codeCreate* avec appel de la m´ethode *update* `a la place de *save*).

**Suppression d’un ´el´ement (delete)** La suppression d’un ´el´ement retrouve le mˆeme sch´ema action+vue+route que les pr´ec´edents outils.

|  |
| --- |
| public Result delete(Long id) { Person p = Person.find.byId(Id) ; p.delete();  return ok(routes.HomeController.delete(id)) ;  } |

1

2

3

4

5

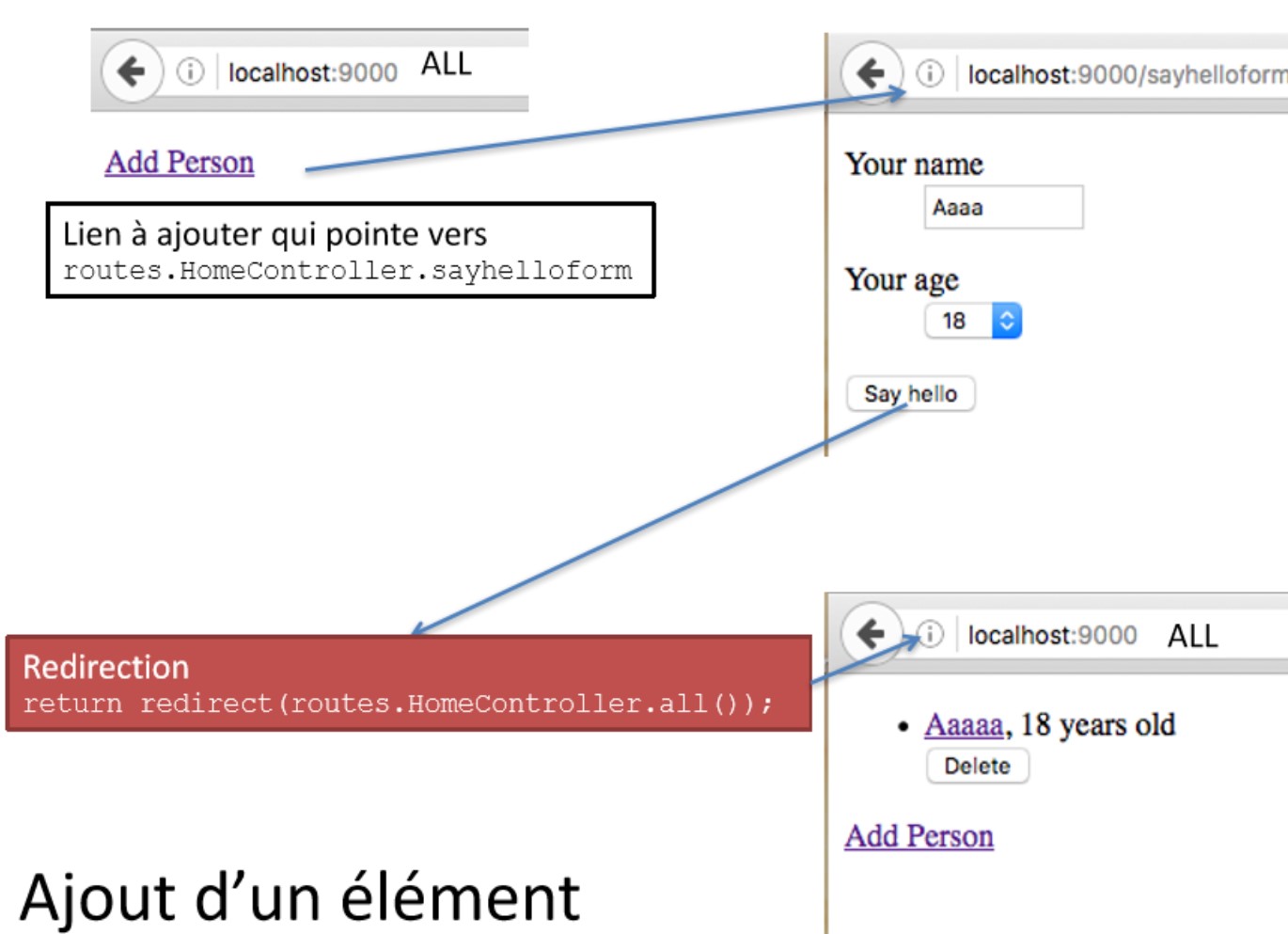
Listing 9: codeDeleteAction

De le mˆeme fa¸con, qu’il est n´ecessaire de rajouter *codeDeleteAction*, il est n´ecessaire d’impl´ementer la route (avec la valeur de l’ID), et la vue.

## Travail sur ordinateur `a r´ealiser

Lors de votre projet final, deux bases de donn´ees doivent ˆetre cr´e´ees. La premi`ere base de donn´ees correspond `a la classe Person (inscription d’un membre et enregistrement des donn´ees relatives `a ce membre). La seconde correspond `a la gestion du contenu. Par exemple dans le cadre du projet biblioth`eque, cette base de donn´ees correspond aux livres. Dans le cadre du projet des jeux, cette base de donn´ees correspond aux parties.

La seconde notation de ce TD s’appuie sur le travail de votre projet. Vous devez faire les programmes suivants :



1. rajouter un statut `a la classe Person. Ce statut est ”membre ou administrateur”, avec la possibilit´e de :
   1. rajouter une personne (create) avec un formulaire et de choisir son statut
   2. lister les personnes (read) avec leur statut
   3. changer le statut d’une peronne (update)
   4. supprimer le statut d’une personne (delete). Le statut devient vide ou par d´efaut
2. rajouter la seconde Base de donn´ees avec le CRUD correspondant;
3. rajouter un petit programme en java parmi ces deux choix :
   * Un programme qui met `a jour l’ˆage de la personne en fonction de la date de naissance. Par exemple : en entr´ee, vous avez ”Date naissance : 1/01/2000” et en sortie ”vous avez 20 ans”. Par exemple

|  |
| --- |
| http://www.quel-age.net/ |

:

* + Un programme qui dit le nombre de personnes dans la liste de personnes (nombre d’inscrits) ou mˆeme (en option) le nombre d’inscrits par statut

## Aide pour le calcul de l’ˆage

|  |
| --- |
| import java.util.Calendar; import java.util.Date;  class Test2  { |

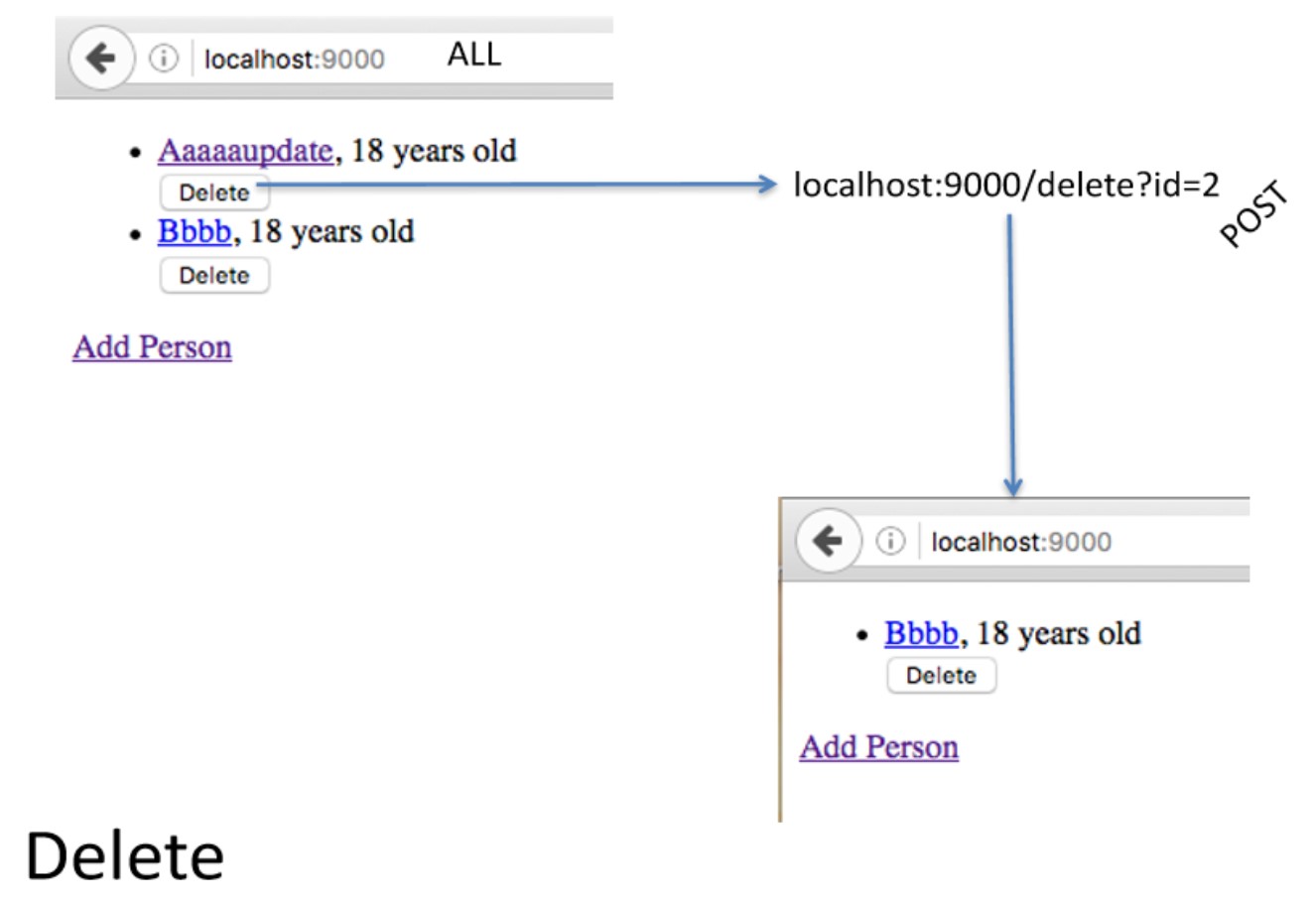
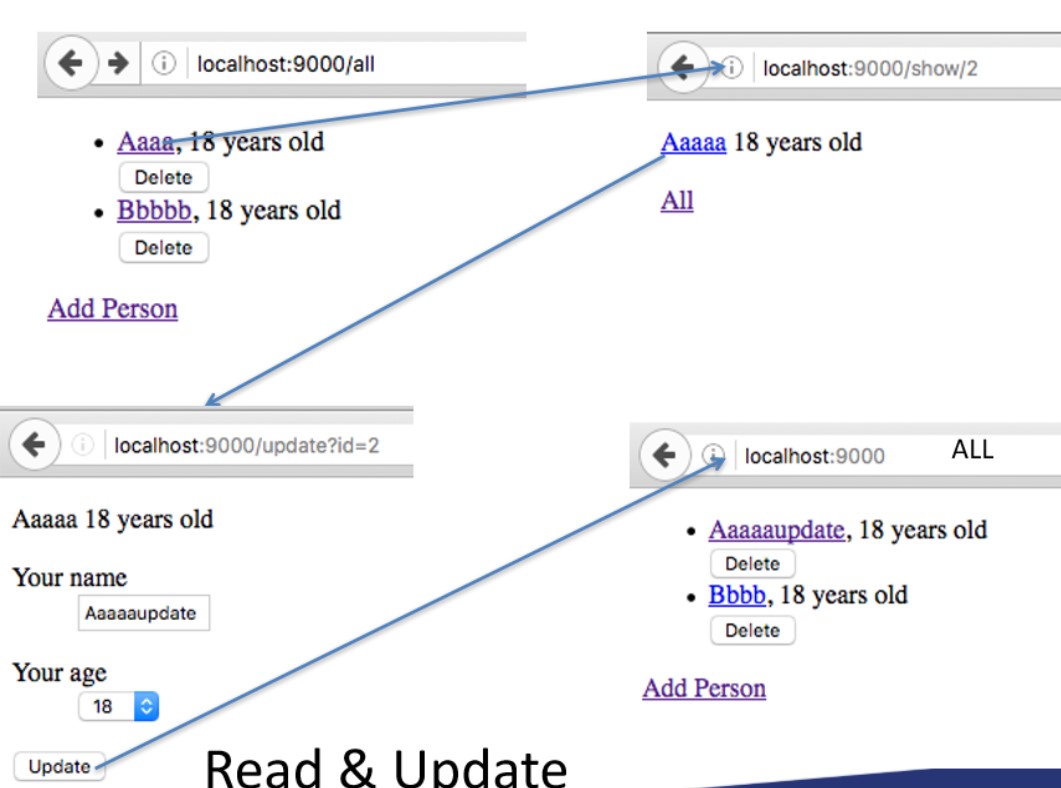
1

2

3

4

5



|  |
| --- |
| public static int getYears(Date d)  {  Calendar curr = Calendar.getInstance(); Calendar birth = Calendar.getInstance(); birth.setTime(d); int yeardiff = curr.get(Calendar.YEAR) - birth.get(Calendar. YEAR); curr.add(Calendar.YEAR,-yeardiff); if(birth.after(curr))  {  yeardiff = yeardiff - 1;  } return yeardiff;  }  public static void main(String[] args)  {  if(args.length != 3) { System.err.println("Erreur");System.exit  (1); }  Calendar birthdaycal = Calendar.getInstance(); birthdaycal.set(Integer.parseInt(args[0]),Integer.parseInt(args  [1])-1,Integer.parseInt(args[2]));  Date birthday = birthdaycal.getTime();  System.out.println("L’age du capitaine est : "+getYears( birthday));  }  } |

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

Listing 10: Age

# Evaluation

## Auto-´evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| Travail r´ealis´e | Not´e sur | Ma note | Note du professeur |
| R´eponses ´ecrites | 5 |  |  |
| CRUD BDD1 (Person) | 4 |  |  |
| CRUD BDD2 | 4 |  |  |
| Programme | 5 |  |  |
| Code bien ´ecrit (pratique) | 2 |  |  |
| Travail sur le Git | 1 |  |  |
| Auto-´evaluation correcte (point bonus) | 1 |  |  |

Table 1: Evaluation

*Lors de l’´evaluation, l’´etudiant peut ˆetre invit´e `a expliquer `a l’oral son code au professeur.*