Application des surnoms des enseignants avec Adonis - Step13

Gestion des rôles

Dans cette étape nous allons mettre en place la protection des pages et la gestion des rôles.

Utilisateur non authentifié

Un utilisateur non authentifié peut uniquement voir la liste des enseignants de la homepage.

Aucune action est possible pour lui.

Utilisateur authentifié non admin

Un utilisateur authentifié mais non admin peut uniquement :

- voir la liste des enseignants de la homepage
- voir les détails d'un enseignant

Utilisateur authentifié mais admin

Un utilisateur authentifié et admin peut tout faire sur l'application.

Utilisation des middlewares auth et guest

Par défaut, nous avons plusieurs middlewares à disposition :

- auth() : il faut que l'utilisateur soit authentifié pour accéder à la route en question
- guest() : il faut que l'utilisateur ne soit pas authentifié pour accéder à la route en question

Voilà le fichier des routes mis à jour :

```
/*
|--------------------------------------------------------------------------
| Le fichier des routes
|--------------------------------------------------------------------------
|
| Le fichier des routes a pour but de définir toutes les routes HTTP.
|
*/

import AuthController from '#controllers/auth_controller'
import TeachersController from '#controllers/teachers_controller'
import router from '@adonisjs/core/services/router'
import { middleware } from './kernel.js'

// La route de la homepage est accessible aussi bien à un utilisateur authentifié
que non authentifié
```

```
router.get('/', [TeachersController, 'index']).as('home')

// Route permettant de voir les détails d'un enseignant accessible aussi bien à un
utilisateur authentifié que non authentifié
router.get('/teacher/:id/show', [TeachersController, 'show']).as('teacher.show')

// Toutes les routes de ce groupe sont soumises à l'authentification
router
  .group(() => {
    // Route permettant d'afficher le formulaire permettant l'ajout d'un
enseignant
    router.get('/teacher/add', [TeachersController,
'create']).as('teacher.create')

    // Route permettant l'ajout de l'enseignant
    router.post('/teacher/add', [TeachersController, 'store']).as('teacher.store')

    // Route permettant d'afficher le formulaire permettant la mise à jour d'un
enseignant
    router.get('/teacher/:id/edit', [TeachersController,
'edit']).as('teacher.edit')

    // Route permettant la modification de l'enseignant
    router.put('/teacher/:id/update', [TeachersController,
'update']).as('teacher.update')

    // Route permettant de supprimer un enseignant
    router.delete('/teacher/:id/destroy', [TeachersController,
'destroy']).as('teacher.destroy')
  })
  .use(middleware.auth())

// Route permettant de se connecter accessible uniquement à un utilisateur non
authentifié
router.post('/login', [AuthController,
'login']).as('auth.login').use(middleware.guest())

// Route permettant de se déconnecter accessible uniquement à un utilisateur
authentifié
router.post('/logout', [AuthController,
'logout']).as('auth.logout').use(middleware.auth())
```

Protection des routes

Maintenant que nous avons protégé certaines routes en vérifiant, à l'aide du middleware, que l'utilisateur doit être authentifié, que se passe-t-il si,
sans être authentifié, nous essayons d'accéder à l'URL `http://localhost:3333/teacher/add` ?

Le comportement attendu est d'être redirigé sur la homepage afin de proposer à l'utilisateur de se connecter.

Si nous testons cela, nous obtenons l'erreur :

```
Cannot GET:/login
```

Pourquoi ?

Car dans le fichier `middleware\auth_middleware.ts`, la variable `redirectTo` est définie avec la valeur `'/login'`.

C'est le comportement par défaut d'AdonisJs qui s'attend à trouver une route `/login`.

Il nous suffit simplement de changer en `'/'` afin que la redirection fonctionne dans notre cas.

Voici le code complet de ce fichier :

```
import type { HttpContext } from '@adonisjs/core/http'
import type { NextFn } from '@adonisjs/core/types/http'
import type { Authenticators } from '@adonisjs/auth/types'

/**
 * Auth middleware is used authenticate HTTP requests and deny
 * access to unauthenticated users.
 */
export default class AuthMiddleware {
  /**
   * The URL to redirect to, when authentication fails
   */
  redirectTo = '/'

  async handle(
    ctx: HttpContext,
    next: NextFn,
    options: {
      guards?: (keyof Authenticators)[]
    } = {}
  ) {
    await ctx.auth.authenticateUsing(options.guards, { loginRoute: this.redirectTo
})
    return next()
  }
}
```

Si un utilisateur non connecté tente d'accéder à une route protégée par authentification, nous allons lui afficher un message.

Pour cela, ajoutons à notre partials `flash.edge` le code suivant :

```
@error('E_UNAUTHORIZED_ACCESS')
  <div class="alert alert-danger">
    @if($message === "Unauthorized access")
      Vous devez être authentifié pour accéder à cette page
```

```
    @end
  </div>
@end
```

Vérification du rôle admin pour les méthodes du contrôleur TeachersController

Pour mettre en place la gestion des rôles, nous devons vérifier que l'utilisateur a le droit admin pour les méthodes suivantes :

- create
- store
- edit
- update
- destroy

Pour cela nous allons créer un middleware.

```
$ node ace make:middleware EnsureAdmin
> Under which stack you want to register the middleware? · named
DONE:    create app/middleware/ensure_admin_middleware.ts
DONE:    update start/kernel.ts file
```

Lors de la création de ce middleware, la question suivante vous sera posée :

```
> Under which stack you want to register the middleware? ...  Press <ENTER> to select
  server
  router
> named
```

Il faut choisir `named`.

En fonction de la réponse, le configuration du middleware sera différente.

Voir le fichier `start\kernel.ts`

Maintenant que le fichier `ensure_admin_middleware.ts` est créé, il faut le compléter avec le code suivant :

```
import type { HttpContext } from '@adonisjs/core/http'
import type { NextFn } from '@adonisjs/core/types/http'

export default class EnsureAdminMiddleware {
  async handle(ctx: HttpContext, next: NextFn) {
    const { auth, session, response } = ctx

    try {
      // Vérifie si l'utilisateur est connecté
      const isAuthenticated = await auth.check()
      if (!isAuthenticated || !auth.user?.isAdmin) {
        // Affiche un message d'erreur à l'utilisateur
        session.flash('error', 'Vous devez avoir les droits admin pour accéder à
cette page')
```

```
        // Redirige l'utilisateur vers la page d'accueil
        return response.redirect().toRoute('home')
      }

      // Passe au middleware suivant ou à la logique de la route
      await next()
    } catch (error) {
      console.error('Erreur dans EnsureAdminMiddleware :', error)

      // Redirige vers la page d'accueil en cas d'erreur
      session.flash('error', 'Une erreur est survenue')
      return response.redirect().toRoute('home')
    }
  }
}
```

On utilise ce nouveau `middleware` pour protéger nos routes.

```
/*
|--------------------------------------------------------------------------
| Le fichier des routes
|--------------------------------------------------------------------------
|
| Le fichier des routes a pour but de définir toutes les routes HTTP.
|
*/

import AuthController from '#controllers/auth_controller'
import TeachersController from '#controllers/teachers_controller'
import router from '@adonisjs/core/services/router'
import { middleware } from './kernel.js'

// La route de la homepage est accessible aussi bien à un utilisateur authentifié
que non authentifié
router.get('/', [TeachersController, 'index']).as('home')

// Route permettant de voir les détails d'un enseignant
router
  .get('/teacher/:id/show', [TeachersController, 'show'])
  .as('teacher.show')
  .use(middleware.auth())

// Toutes les routes de ce groupe sont soumises à l'authentification
router
  .group(() => {
    // Route permettant d'afficher le formulaire permettant l'ajout d'un
enseignant
    router.get('/teacher/add', [TeachersController,
'create']).as('teacher.create')

    // Route permettant l'ajout de l'enseignant
```

```
    router.post('/teacher/add', [TeachersController, 'store']).as('teacher.store')

    // Route permettant d'afficher le formulaire permettant la mise à jour d'un
enseignant
    router.get('/teacher/:id/edit', [TeachersController,
'edit']).as('teacher.edit')

    // Route permettant la modification de l'enseignant
    router.put('/teacher/:id/update', [TeachersController,
'update']).as('teacher.update')

    // Route permettant de supprimer un enseignant
    router.delete('/teacher/:id/destroy', [TeachersController,
'destroy']).as('teacher.destroy')
  })
  .use(middleware.auth())
  .use(middleware.ensureAdmin())

// Route permettant de se connecter accessible uniquement à un utilisateur non
authentifié
router.post('/login', [AuthController,
'login']).as('auth.login').use(middleware.guest())

// Route permettant de se déconnecter accessible uniquement à un utilisateur
authentifié
router.post('/logout', [AuthController,
'logout']).as('auth.logout').use(middleware.auth())
```

Cacher les icons en fonction de l'utilisateur authentifié ou non, admin ou non

La dernière chose qu'il nous reste à faire est de nous assurer que:

- un utilisateur authentifié peut voir l'icon de l'action `show`
- seul un admin peut voir les icons des actions `create`, `edit` et `destroy` depuis les vues `show` et `home`.

Dans le fichier `views/partials/header.edge` est :

```
<nav>
  <a href="{{ route('home') }}">Accueil</a>  @if(auth.isAuthenticated &&
auth.user.isAdmin)
  <a href="{{ route('teacher.create') }}">Ajouter un enseignant</a>
  @end
</nav>
```

Dans le fichier `views/pages/home.edge` est :

```
<td class="containerOptions">
  @if(auth.isAuthenticated && auth.user.isAdmin)
```

```
  <a href="{{ route('teacher.edit', {id: teacher.id}) }}">
    <img height="20em" src="/img/edit.png" alt="edit" />
  </a>
  {{-- Ajout d'un formulaire pour la suppression d'un enseignant --}}
  <form action="{{ route('teacher.destroy', { id: teacher.id }) }}?_method=DELETE"
method="POST"
    style="display: inline">
    {{ csrfField() }}
    <button type="submit" style="background: none; border: none; padding: 0;
cursor: pointer;">
      <img height="20em" src="/img/delete.png" alt="delete" />
    </button>
  </form>
  @end
  @if(auth.isAuthenticated)
  <a href="{{ route('teacher.show', {id: teacher.id}) }}">
    <img height="20em" src="/img/detail.png" alt="detail" />
  </a>
  @end
</td>
```

Dans le fichier `views/pages/teachers/show.edge` est :

```
<div class="actions">
    @if(auth.isAuthenticated && auth.user.isAdmin)
    <a href="{{ route('teacher.edit', {id: teacher.id}) }}">
        <img height="20em" src="/img/edit.png" alt="edit" />
    </a>
    {{-- Ajout d'un formulaire pour la suppression d'un enseignant --}}
    <form action="{{ route('teacher.destroy', { id: teacher.id }) }}?
_method=DELETE" method="POST"
        style="display: inline">
        {{ csrfField() }}
        <button type="submit" style="background: none; border: none; padding: 0;
cursor: pointer;">
            <img height="20em" src="/img/delete.png" alt="delete" />
        </button>
    </form>
    @end
</div>
```