Implementation of the authentication / authorizaton

This document explains how the developer team implemented security and authorization in the Todo list project, on symfony 4.4 LTS

Security

we first create a security in the security.yaml configuration, the access_control key allows anonymous access only for the login route

The other route is reserved for members :

MEMBER
- /tasks
- /tasks/create
- /tasks/finished
- /tasks/{id}/edit
- /tasks/{id}/delete
- /tasks/{id}/toggle ;
- /logout

ADMIN
- /users
- /users/create
- /users/{id}/edit

```
access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/users, roles: [ROLE_ADMIN] }
    - { path: ^/, roles: [ROLE_USER, ROLE_ADMIN]}
```

We also associated the tasks of the application to members, to be able to have a hierarchy in the rights compared to the tasks, so we know to which user a task belongs, which allows to have the rights of deletion or not on a task
If a task have not author : this task is “anonymous task”

- Admin can delete anonymous tasks and their own tasks
- Member can delete only their own tasks

authentication

authentication is simple in this project. We have set up a User entity, which logs in (unique username) and a password. This user is stored in the database, we record his first name, email, and roles.
The role is important because it is he who decides the rights.

The login and logout route are automatically generated in this project thanks to the User interface, you can access the connection information and manipulate the members thanks to this interface

Authorization

To allow members to delete a task or not, this is how we proceed

1. We will recover the task in question and the connected User object.
2. We check its role
3. If it is an ADMIN and the task is his or anonymous, he can delete the task
4. Else If it's a classic USER and the stain is his, he can delete the stain
5. Else we do not display the delete button, or we display a message to say that we do not have the rights

```php
/**
 * @Route("/tasks/{id}/delete", name="task_delete")
 */
public function deleteTaskAction(EntityManagerInterface $em, $id)
{
    $taskRepository = $this->getDoctrine()->getRepository( persistentObject: Task::class);
    $task = $taskRepository->findOneBy(array('id' => $id));
    $role = $this->getUser()->getRoles();

    if($role[0] == 'ROLE_ADMIN' && $task->getUser() == $this->getUser() || $task->getUser() == null){

        $em->remove($task);
        $em->flush();
        $this->addFlash( type: 'success',  message: 'Task is deleted');
    }
    elseif($role[0] == 'ROLE_USER' && $task->getUser() == $this->getUser()){

        $em->remove($task);
        $em->flush();
        $this->addFlash( type: 'success',  message: 'Task is deleted');
    }

    else{
        // sinon , impossible de supprimer la tache
        $this->addFlash( type: 'success',  message: 'Impossible with your right');
    }
    return $this->redirectToRoute( route: 'task_list');
}
```

Resume :

- users are stored in the database with a role (ADMIN OR MEMBER)
- For change right : you can change a security.yaml or change a function in a controller
- If you do not have access to the application page, you are automatically redirected to the login page

If you have additional questions about the contribution, in general or in project security, you can go see the readme of the github project, or contact me with my email address.