TP5 : EventListeners

Objectif : Comprendre les Events Symfony et implémenter un listener / un subscriber et un Event custom. Utiliser le Dispatcher.

1. A l'aide des mécanismes vu ensemble dans les TP précédents, créer les entités suivantes :
    a. Un entité "ClientTable")représentant les différentes tables du restaurants :
        i. nom de la table (ou numéro de la table)
        ii. nom de la salle (grande salle / petite salle)
    b. Insérez manuellement en base de données (ou via une commande) les différentes tables de votre restaurant.
    c. Une entité "ClientOrder" :
        i. numéro de commande (ID)
        ii. date/heure de prise de commande
        iii. Utilisateur (serveur) à l'origine de la commande (ManyToOne vers User)
        iv. Table concernée par la commande (ManyToOne vers ClientTable)
        v. Plats contenus dans la commande (ManyToMany vers Dish)
        vi. Prix total de la commande
        vii. Status de la commande (Prise / Préparée / Servie / Payée)
2. A l'aide des mécanismes vu dans les TP précédent, créez le formulaire de prise de commande dans le back-office.

3. Créez un EventSubscriber permettant de calculer le montant total de la commande avant que celle ci soit enregistrée en base de donnée.
    a. A lire :
        i. http://symfony.com/doc/3.3/event_dispatcher.html (POUR SF3)
        ii. https://symfony.com/doc/current/doctrine/event_listeners_subscribers.html
    b. A l'aide de la documentation : Créez sous la forme d'un service, un EventSubscriber réagissant aux évènements doctrine : PrePersist et PreUpdate
    c. Au PrePersist, ce service renseignera automatiquement
        i. la date/heure de prise de commande.

ii. le status de la commande à "Prise".

iii. l'utilisateur / serveur ayant pris la commande (utilisateur connecté).

d. Au PrePersist ET au PreUpdate, ce service calculera automatiquement le montant total de la commande en additionnant les prix des plats contenus dans la commande.

4. Créez 2 EventListeners permettant de gérer la gestion de la commande dans le restaurant. Nous utiliserons ici des emails mais cela pourrait être des notifications mobiles ou tout autre système de message.

    a. Vous devrez ici, injecter dans vos services, le service 'Mailer'

    b. Un EventListener réagissant au PostPersist Doctrine sur l'entité Order.

        i. Un email est envoyé aux cuisines indiquant qu'une nouvelle commande a été prise en salle.

    c. Un EventListener réagissant au PostUpdate Doctrine sur l'entité Order

        i. Si la commande est au statut "Préparée" un email est envoyé au serveur pour lui indiquer qu'il peut servir la commande.

        ii. Si la commande est au statut "Servie" un email est envoyé à l'accueil pour indiquer que la commande pourra être encaissée.

        iii. Si la commande est au statut "Payée" un email est envoyé au serveur pour lui indiquer que sa commande a été facturée.

5. Lorsqu'une commande est payée, l'application Restau doit aussi indiquer au système RH que l'argent a été encaissé. Nous pourrions réaliser cela dans les Listeners créés ci dessus, mais nous souhaitons que l'application Restau soit évolutive, et puisse être compatible si le restaurant mets en place d'autres mécanismes Digitaux à l'avenir. Nous allons donc créer notre propre évènement : **order.payed** qui pourra être utilisé par n'importe quel bundle à l'avenir.

6. A l'aide de la documentation, créez cet évènement

    a. Documentation : https://symfony.com/doc/3.3/components/event_dispatcher.html (POUR SF 3.3)

    b. Créez une classe d'évènement dans App/Event qui s'appellera OrderPayedEvent

7. Ajoutez dans le Listener PostUpdate de l'entité Order, le dispatch de l'event order.payed (pour ce faire, vous devrez injecter l'EventDispatcher en dépendance de votre service Listener PostUpdate).

8. En utilisant les mécanismes déjà vu précédemment, créez un listener sur votre propre évènement **order.payed**.

    a. Ce listener devra disposer en dépendance du service d'accès à l'API du système RH développé au TP4. Ce dernier devra contenir une nouvelle méthode **setOrderPayed($order)** . Exemple de code rapide :

```
47      /**
48       * Helper to declare a payed order in RH system
49       */
50      public function setOrderPayed($order)
51      {
52          $url = $this->api_endpoint.
53              "?method=order&order=".$order->getId().
54              "&amount=".$order->getAmount().
55              "&server=".$order->getOwner()->getUsername();
56          return json_decode( file_get_contents($url) );
57      }
58  }
```

b. Lors du déclenchement de l'évènement **order.payed** sur une commande, la méthode **setOrderPayed** sera déclenchée. Et la commande sera donc automatiquement comptabilisée dans l'outil RH de l'établissement.