IUT La Rochelle, Département Informatique

Programmation Web côté Serveur

2ère année DUT - 2020/2021

Module **PWS3** **TD 2 Découverte symfony suite**

Reprenez votre projet jeuxdicode débuté la semaine dernière.

L'application Web Jeuxdicode est une application qui permettra de faire connaitre les conférence Jeuxdicode. Comme son nom l'indique les jeuxdicode sont des évènements de type conférences qui ont lieu les jeudi. Un évènement est proposé par toute personne souhaitant présenter un thème dans le domaine du développement, de la gestion de projet, de l'ux, des tests ou autres.
Un évènement peut se décomposer en plusieurs sessions de présentation.
Notre application sera décomposée en deux parties un front office pour les participants leur permettant de connaitre les futurs évènements et de s'y inscrire et une partie backoffice qui permettra à un admin jeuxdicode de consulter les évènements à venir et les évènements passés et de stocker les propositions de sessions à venir.
Nous allons nous intéresser plus particulièrement à la partie backoffice pour l'admin.

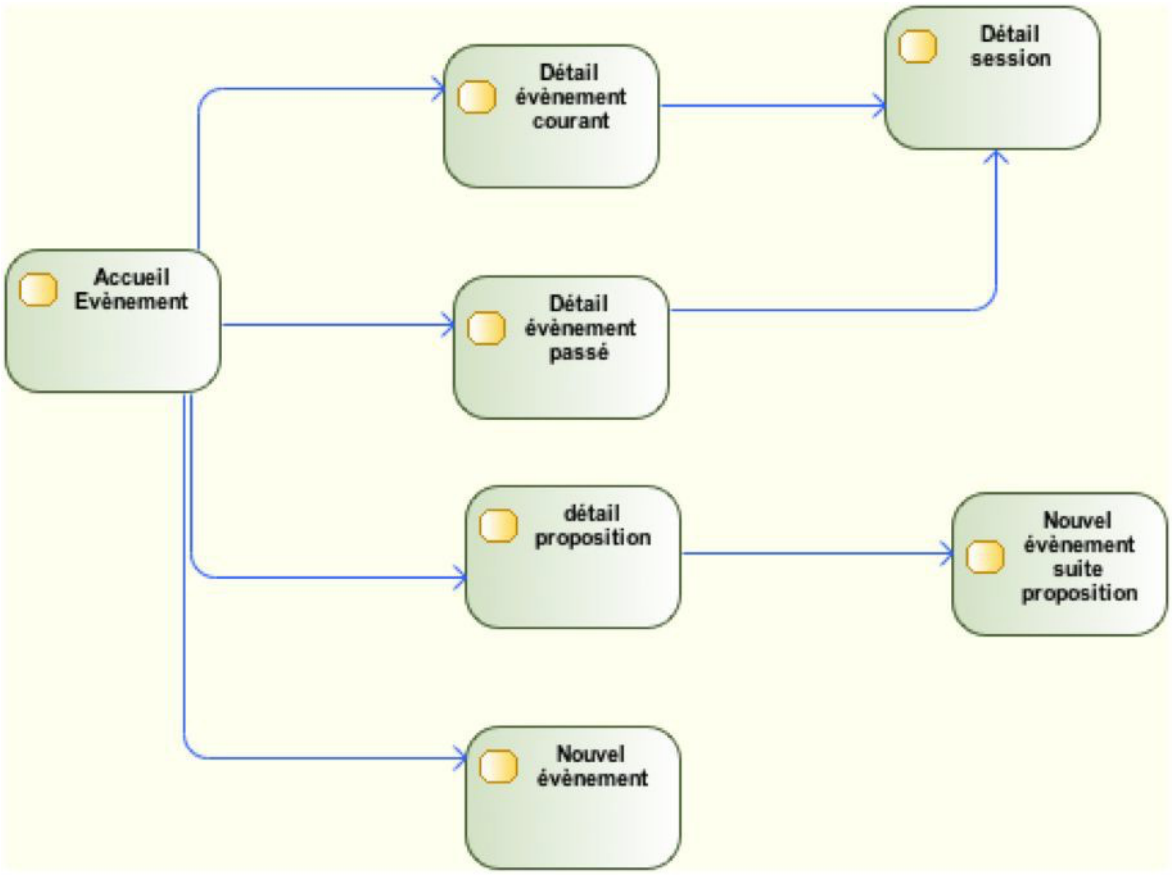
Une étude a été faite qui a conduit à mettre en place les fonctionnalités suivantes par pages.

Accueil évènement	Cette page affichera la description résumée de l'évènement avec possibilité d'accéder au détail de l'vènement courant. Elle affichera également la liste des évènements passés avec possibilité d'accéder au détail des évènements passés.(sous forme de tableau) Enfin elle permettra de voir la liste des propositions de futures sessions avec possibilité d'accéder au détail de cette proposition.
Détail événement courant	Cette page affichera le détail de l'événement courant, dont la ou et les descriptions des sessions constituant l'événement Avec la liste des inscrits au moment de la consultation.
Détail session	Cette page affichera le détail d'une session d'un événement
Détail événement passé	Cette page affichera le détail de l'événement passé avec la liste des inscrits à cet évènement

Ajout événement	Cette page permettra d'ajouter un événement suite à une proposition hors système.
Ajout événement à partir d'une proposition	Cette page permettra d'ajouter un événement suite à une proposition stockée dans le système (faite par un membre)
Détail proposition	Cette page permettra d'afficher le détail d'une proposition de session pour un événement jeuxdicode, nous pourrons depuis cette page demander la création d'un événement à partir de cette proposition.

Nous pouvons désormais créer le squelette de notre application pour mettre en place la navigation et les composants pour l'admin.

Spécification de la navigation du BO de l'admin :



Chaque page (vue) sera pilotée par une méthode et les méthodes sont regroupées dans les contrôleurs.

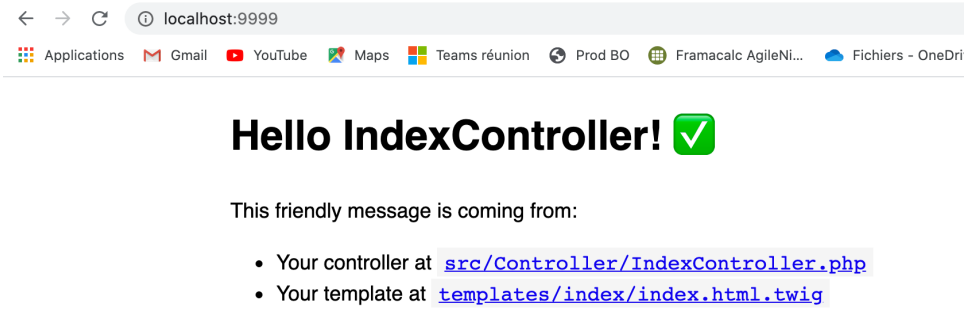
La page d'accueil de la partie admin du site sera gérée par le contrôleur index.

Page	Contrôleur	Méthode
Accueil événement	IndexController	index

Le contrôleur Evenement aura pour l'instant 6 méthodes, qui seront :

Page/Vue	Contrôleur	Méthode	URL
Détail événement courant	EvenementController	detailevt	
Détail événement passé	EvenementController	detailevtpass	
Détail proposition	EvenementController	detailprop	
Détail session	EvenementController	detailsession	
Nouvel évènement	EvenementController	add	
Nouvel événement suite à proposition	EvenementController	addprop	

Etape 1 : Faites un refactoring pour faire que le contrôleur index appelle une vue index.html.twig sur la route /



Etape 2 : Utilisation du CLI pour mettre en place le contrôleur d'évènement - création du squelette de l'application

Complétez les URL dans le tableau pour définir les routes à construire dans le contrôleur.

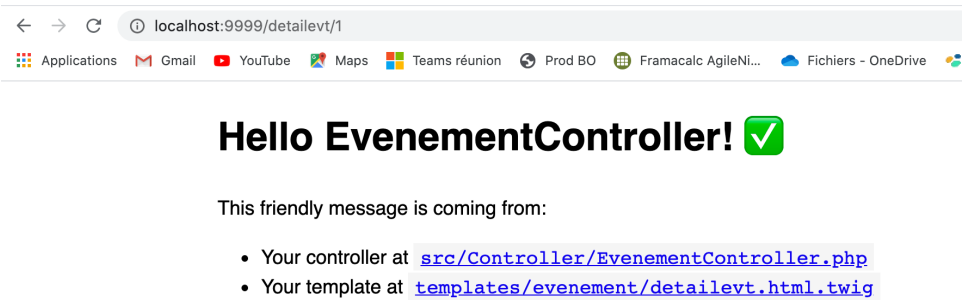
Ajoutez un contrôleur Evenement en utilisant le cli.

Créez les méthodes et les annotations de routes et ajoutez les templates de vues

Exemple pour la page détail évènement :

```php
/**
 * @Route("/detailevt/{id}", name="detailevt")
 */
public function detailevt($id)
{
    return $this->render( view: 'evenement/detailevt.html.twig', [
        'controller_name' => 'EvenementController',
    ]);
}
```

Et sa vue twig.



Naviguez sur l'ensemble des routes mises en place pour les tester.

Etape 2 : le moteur de template Twig

http://twig.sensiolabs.org/documentation

Les templates (vues) ont pour objectif de séparer le code php du code HTML.
Cependant on a toujours besoin de code dynamique pour:

- Faire une boucle pour afficher les données d'une liste
- Créer les conditions pour différentier un affichage

Le moteur de template Twig dispose d'un pseudo-langage.

- {{..}} affiche quelque chose;
- {%..%} fait quelque chose;
- {#..#} syntaxe pour les commentaires

Afficher des variables :

Description	Example Twig
Afficher une variable	{{pseudo}}
Afficher l'index d'un tableau	{{ user['id']}}
Afficher l'attribut d'un objet	{{ user.id }}
Afficher une variable en appliquant un filtre	{{pseudo | upper}}
Afficher une variable en combinant les filtres	{{ news.texte|striptags|title }}
Utiliser un filtre avec des arguments	{{ date | date('d/m/y') }}
concaténer	{{nom~" "~prenom}}

Précision sur la syntaxe {{ objet.attribut }} :

- Elle vérifie si objet est un tableau et si attribut en est un index valide, si c'est le cas, elle affiche objet ['attribut']
- Sinon si objet est un objet, elle vérifie si attribut en est un attribut valide (public). Si c'est le cas elle affiche objet ->attribut()
- Sinon si objet est un objet, elle vérifie si getAttribute() en est une méthode valide. Si c'est le cas elle affiche objet->getAttribut()
- Sinon si objet est un objet, elle vérifie si isAttribute() en est une méthode valide. Si c'est le cas elle affiche objet->isAttribut()
- Sinon elle n'affiche rien et retourne null

Les filtres :

http://twig.sensiolabs.org/doc/filters/index.html

Consultez les filtres utilisables.

Quelques variables globales disponibles:

{{ app.request}}
{{ app.session }}
{{ app.environnement }}

Les structures de contrôle et expression :
Consultez la documentation :

http://twig.sensiolabs.org/doc/tags/index.html

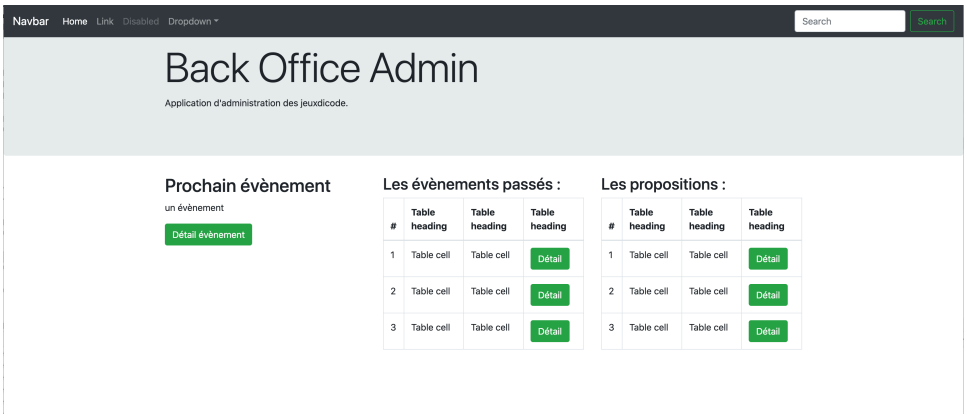
Hériter des templates :

Principe : un template père qui contient le design global du site, ainsi que de blocs et des templates fils qui vont remplir ces blocs.
Regarder la page base.html.twig située à la racine du répertoire “templates” qui est un template père

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}{% endblock %}
        <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" />
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}{% endblock %}
    </body>
</html>
```

Puis regardez dans le fichier index.html.twig comment ce template hérite du template père:
{% **extends** **'base.html.twig'** %}

Mise en oeuvre :
On veut maintenant intégrer bootstrap à notre application pour cela intégrer directement les fichiers bootstrap avec le CDN fourni, à votre template base.html.twig à la racine du répertoire “templates”.
https://getbootstrap.com/docs/4.3/getting-started/introduction/
Utilisons l’exemple Jumbotron.
Nous allons travailler l’héritage sur la page index.html.twig qui correspond à la page d’accueil de notre application.
Nous souhaitons qu’elle suive le modèle suivant :



A vous de jouer.

Dernier travail nous allons simuler dans le contrôleur la récupération de l'événement courant en le stockant dans un tableau pour cet événement on connaîtra son identifiant, sa date et sa description résumée. Mais en affichage dans la vue on récupère uniquement sa description.

A vous de jouer.