Développement web côté serveur avec Symfony

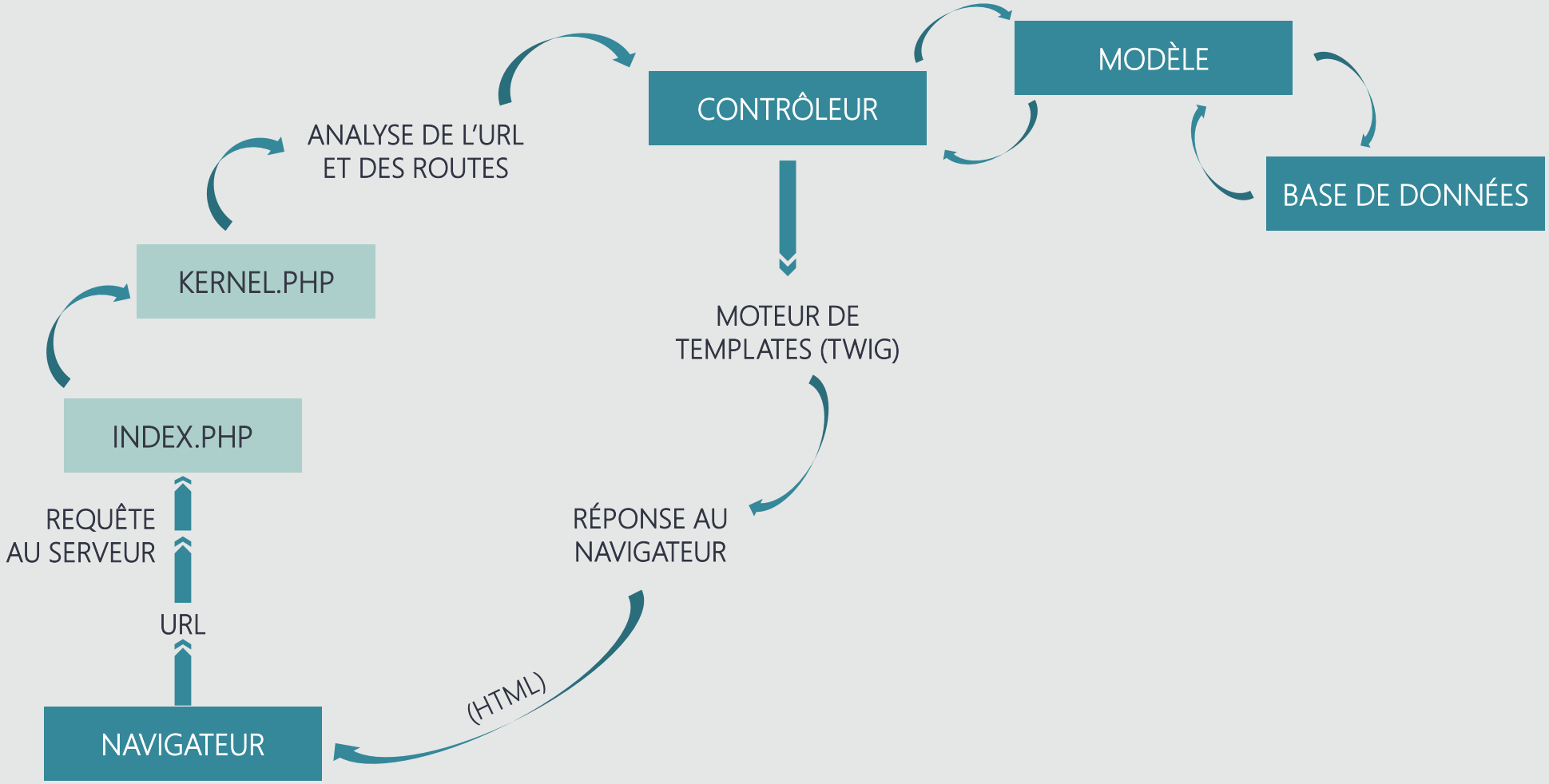
Module 04 - Routage et contrôleurs

Objectifs

- Comprendre et savoir utiliser le système de routage
- Connaître les fonctionnalités des contrôleurs
- Savoir créer un site sans base de données

Routage et contrôleurs

Le parcours d'une requête HTTP sous Symfony



Routage et contrôleurs

Les routes

- Correspondance URL / Fonction
- Permettent de facilement créer de jolies URL
- Système souples

Les routes : configuration

- PHP, YAML, XML ou annotations
- Avantage aux annotations
- PHP8 : remplacement possible des annotations par les *Attributes*

Les routes : les annotations

- Commentaires interprétés
- *Au-dessus* du code PHP associé
- Doivent impérativement avoir cette forme :

```
/**
 * @UneAnnotation(cle="valeur1", cle2="valeur2")
 */
```

Les routes en annotation

- Dans le contrôleur :
  - Importer (donner un alias) à la classe de route de Symfony
  - Créer l'annotation au-dessus de chaque méthode

Routage et contrôleurs

Les paramètres de routes

- Premier paramètre : l'URL souhaitée
- Autres paramètres incontournables :
  - name
  - requirements
  - methods

Les paramètres d'annotations de routes : l'URL

- Précise l'URL menant à la méthode
- Doit être unique
- Forme libre !

```
/**
 * @Route("/utilisateurs/profils")
 */
public function listUsers()
{
```

Les paramètres d'annotations de routes : l'URL

Les jokers dans l'URL : une bénédiction

http://monsite.com/utilisateurs/profils/?id=123

http://monsite.com/utilisateurs/profils/123

Les paramètres d'annotations de routes : l'URL

Les jokers dans l'URL : syntaxe

- Entre accolades
- Passés automatiquement à la méthode

```
/**
 * @Route("/utilisateurs/profils/{id}")
 */
public function userProfile($id)
{

}
```

Les paramètres d'annotations de routes : l'URL

Les jokers dans l'URL : précisions

- Expression rationnelle correspondant au paramètre d'URL

```
/**
 * @Route("/utilisateurs/profils/{id}", requirements={"id"="\d+"})
 */
```

- Valeur par défaut

```
/**
 * @Route("/utilisateurs/profils/{page}")
 */
public function listUsers($page = 1)
{

}
```

Routage et contrôleurs

Les routes : *requirements* et autres conditions

- Permettent d'ajouter des conditions au *match* avec la requête
  - `method`
  - `host`
  - `_locale`
  - caractères UTF-8
  - …

Noms des routes

- Très fortement recommandés
- Permettent de générer les URL correspondant aux routes
- Convention : nom du contrôleur + méthode en *snake_case*
- Uniques

```php
// dans un ProductController...

/**
 * @Route("/products/detail/{id}", name="product_detail")
 */
public function detail($id)
{
```

Routage et contrôleurs

Les routes et la console

- Afficher la liste des routes :

```
php bin/console debug:router
```

- Afficher les détails d'une route :

```
php bin/console debug:router nomDeLaRoute
```

- Tester le match entre une URL et une route :

```
php bin/console router:match /url-a-tester
```

Les contrôleurs

- Simple classe PHP
- Hérite du contrôleur de Symfony
- Nombre illimité
- Chaque méthode = une page
- Accès facile à `...HttpFoundation\Request`
- Une obligation : retourner un objet `...HttpFoundation\Response`

Le contrôleur de base de Symfony

- Affichage et réponses :
  - `render()`
  - `json()`
  - `file()`
  - `createNotFoundException()`
- Routage
  - `redirectToRoute()`
  - `generateURL()`
  - `redirect()`
- Sécurité
  - `getUser()`
  - `CreateAccessDeniedException()`
  - `isGranted()`
  - `denyAccessUnlessGranted()`
- Modèle et données
  - `createForm()`
  - `getDoctrine()`
- Divers
  - `addFlash()`

Routage et contrôleurs

Les contrôleurs : préfixes

```php
class ProductController extends AbstractController
{
    /**
     * @Route("/product/", name="product_list")
     */
    public function list(): Response
    {
        return $this->render('product/list.html.twig');
    }

    /**
     * @Route("/product/{id}", name="product_detail")
     */
    public function detail($id): Response
    {
        return $this->render('product/detail.html.twig');
    }
}
```

Devient

```php
/**
 * @Route("/product", name="product_")
 */
class ProductController extends AbstractController
{
    /**
     * @Route("/", name="list")
     */
    public function list(): Response
    {
        return $this->render('product/list.html.twig');
    }

    /**
     * @Route("/{id}", name="detail")
     */
    public function detail($id): Response
    {
        return $this->render('product/detail.html.twig');
    }
}
```

Les contrôleurs et la console

- Créer un nouveau contrôleur :

```
php bin/console make:controller
```

puis donner le nom du contrôleur

- Ou plus directement :

```
php bin/console make:controller Profile
```

pour générer un *ProfileController.php*

Routes et contrôleurs

Démonstration

Débogage avec le VarDumper

- Alternative à `var_dump()` ou `print_r()`
- `dump()` réalise un joli var_dump(). S'affiche dans la debug bar.
- `dd()` réalise un dump(), puis un die()

Débogage avec le Profiler

- Debug Bar affichée au bas du site
- Profiler : encore plus d'informations !
- Inspection du déroulement de la requête

Débogage

Démonstration

Conclusion

- Vous savez (presque) tout sur le routage Symfony
- Vous savez utiliser les contrôleurs et les méthodes fournies par Symfony
- Vous savez comment créer un site sans base de données