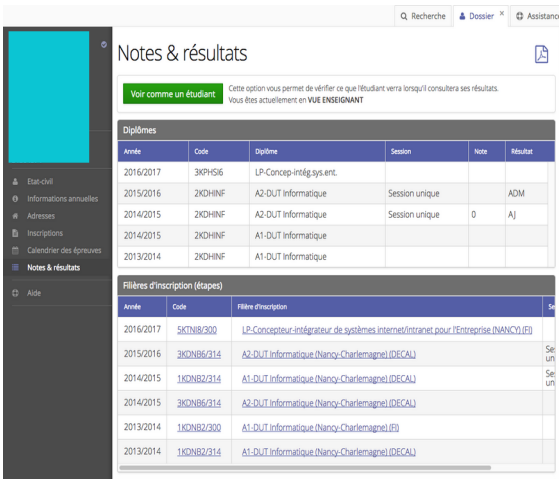


Server side rendering,
Wrapper html,
templates

Server-side rendering



Notes & résultats

Voir comme un étudiant

Cette option vous permet de vérifier ce que l'étudiant verra lorsqu'il consultera ses résultats. Vous êtes actuellement en VUE ENSEIGNANT

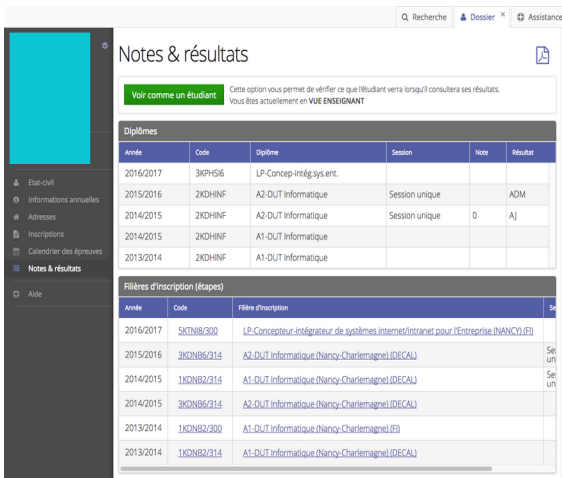
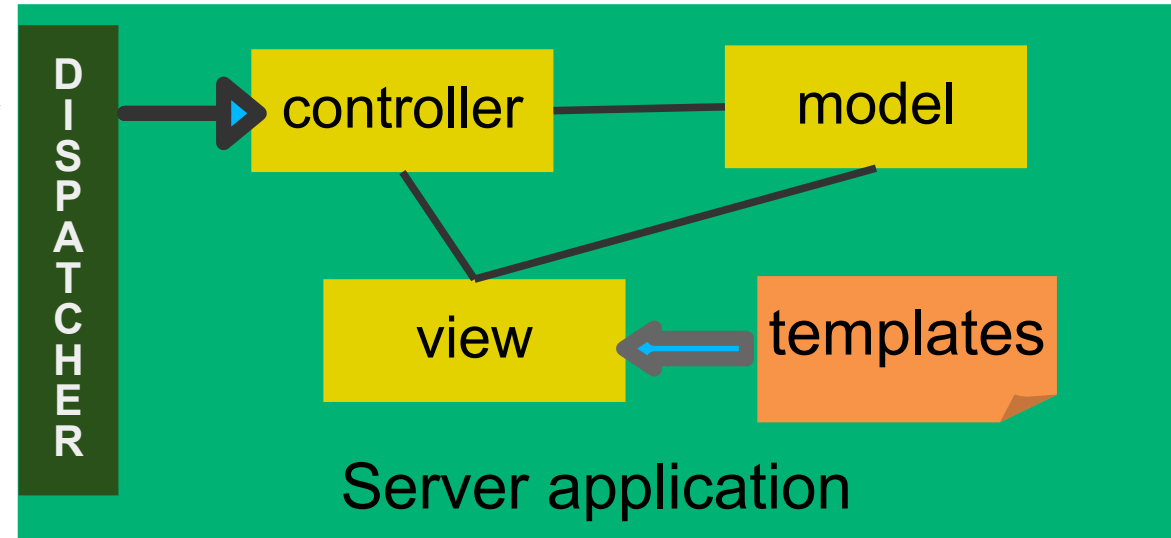
Année	Code	Diplôme	Séance	Note	Résultat
2016/2017	3KPH5G6	LP-Concept-intégr.syst.ent.			
2015/2016	2KDHNIF	A2-DUT Informatique	Séance unique		ADM
2014/2015	2KDHNIF	A2-DUT Informatique	Séance unique	0	AJ
2014/2015	2KDHNIF	A1-DUT Informatique			
2013/2014	2KDHNIF	A1-DUT Informatique			

Filtres d'inscription (étapes)

Année	Code	Filtre d'inscription
2016/2017	SKTNB200	LP-Concepteur-intégrateur de systèmes internet/terminal pour l'entreprise (NANCY) (F)
2015/2016	3KDNB6314	A2-DUT Informatique (Nancy-Charlemagne) (DECAU)
2014/2015	1KDNB2314	A1-DUT Informatique (Nancy-Charlemagne) (DECAU)
2014/2015	3KDNB6314	A2-DUT Informatique (Nancy-Charlemagne) (DECAU)
2013/2014	1KDNB2300	A1-DUT Informatique (Nancy-Charlemagne) (F)
2013/2014	1KDNB2314	A1-DUT Informatique (Nancy-Charlemagne) (DECAU)

GET , POST

<html>



Notes & résultats

Voir comme un étudiant

Cette option vous permet de vérifier ce que l'étudiant verra lorsqu'il consultera ses résultats. Vous êtes actuellement en VUE ENSEIGNANT

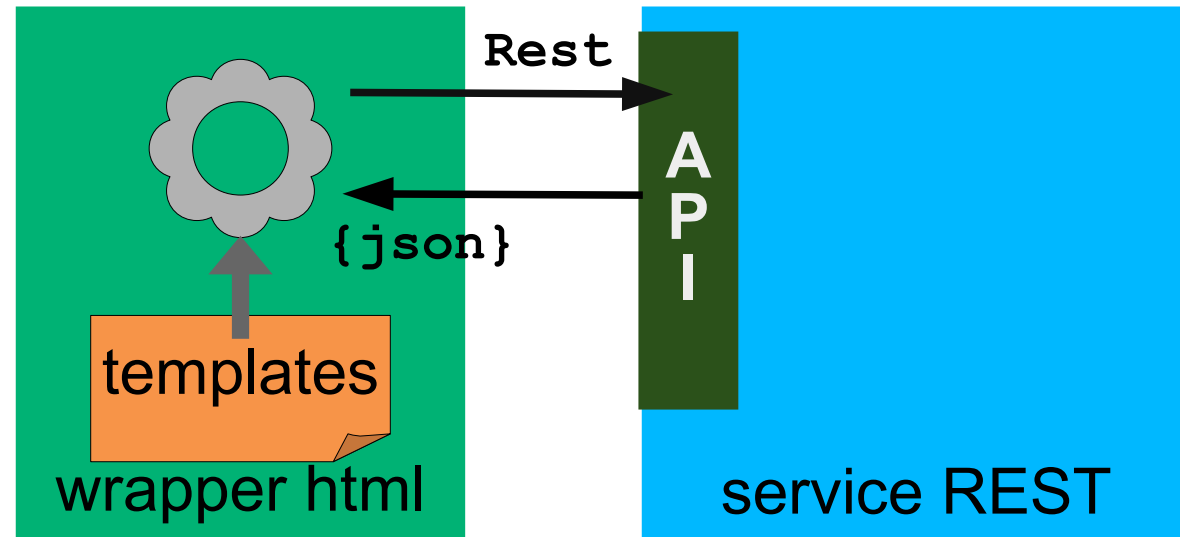
Année	Code	Diplôme	Séance	Note	Résultat
2016/2017	3KPH5G6	LP-Concept-intégr.syst.ent.			
2015/2016	2KDHNIF	A2-DUT Informatique	Séance unique		ADM
2014/2015	2KDHNIF	A2-DUT Informatique	Séance unique	0	AJ
2014/2015	2KDHNIF	A1-DUT Informatique			
2013/2014	2KDHNIF	A1-DUT Informatique			

Filtres d'inscription (étapes)

Année	Code	Filtre d'inscription
2016/2017	SKTNB200	LP-Concepteur-intégrateur de systèmes internet/terminal pour l'entreprise (NANCY) (F)
2015/2016	3KDNB6314	A2-DUT Informatique (Nancy-Charlemagne) (DECAU)
2014/2015	1KDNB2314	A1-DUT Informatique (Nancy-Charlemagne) (DECAU)
2014/2015	3KDNB6314	A2-DUT Informatique (Nancy-Charlemagne) (DECAU)
2013/2014	1KDNB2300	A1-DUT Informatique (Nancy-Charlemagne) (F)
2013/2014	1KDNB2314	A1-DUT Informatique (Nancy-Charlemagne) (DECAU)

GET , POST

<html>



Les mécanismes de templating

- **Un template html** : un fragment de **code html** contenant des **variables** que l'on peut appeler depuis un programme
- un système de **templates** facilite la construction et l'assemblage de code html lors de la construction des vues d'une application web
- côté client en js
- côté serveur en php/node/java ... lorsque l'interface de l'appli est construite par le backend
- Pour **générer une vue** :
 - activer 1 template
 - et lui transmettre des valeurs pour les variables

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="{{ root_uri }}/web/css/main.css">
    <title> {{ titre_liste }} </title>
</head>
<body>

<h1> {{ titre_liste }} </h1>

{% for i in items %}
<div class="item">
    <h3>{{ i.nom }}</h3>
    <p> {{ i.description }}</p>

</div>

{% endfor %}

</body>
</html>
```

Exemple : un template twig pour construire une vue sur le serveur

- Nombreux "Template engines" :
 - Blade, Smartie, **twig** : php
 - Liquid, Haml : ruby
 - handlebars, doT, EJS : js
 - mustache : multi-langage, volontairement réduit

- **En général** :
 - Un langage de définition de templates : extension de html
 - Des fonctions php/js/ruby ... pour manipuler et utiliser ces templates

- **intérêts** :
 - faciliter la construction/maintenance des vues en
 - Séparant plus clairement HTML et code applicatif
 - Facilitant la réutilisation d'éléments de présentation html
 - Homogénéiser l'application du point de vue présentation
 - Faciliter la séparation des métiers : informaticiens vs. Intégrateur html vs designer graphique

Exemple : Définition d'un template avec twig

```
<!DOCTYPE html>
<head> <title>Oh le beau Template</title> </head>
<body>
<h2> {{ title }}</h2><br />
<p>{{ message }} </p><br />
</body>
</html>
```



variables

ex1.html.twig

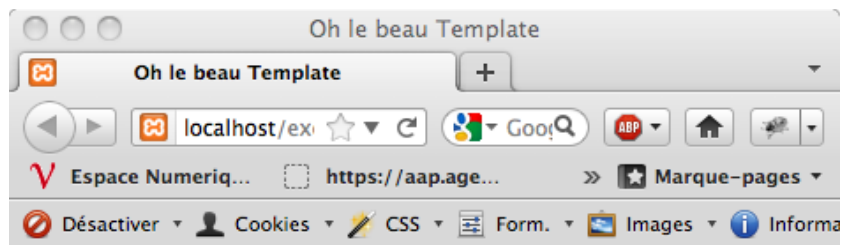
```
<?
include 'vendor/autoload.php' ;

$loader = new Twig_Loader_Filesystem( 'templates' );

$twig = new Twig_Environment($loader,
                                array('debug' => true));

$tmp1 = $twig->loadTemplate('ex1.html.twig');

$tmp1->display( [
    'title' => 'Ah, quel BEAU TITRE',
    'message' => 'Oh, le beau Message'
] ) ;
```

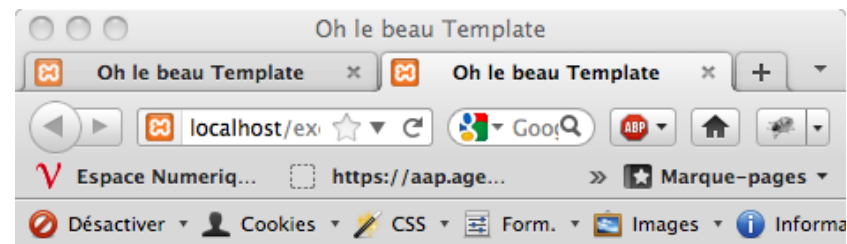


AH, Quel BEAU TITRE

Oh le beau Message



```
title = "Ah Quel BEAU TITRE"  
message = "Oh le beau Message"
```



Ceci est une erreur grave

[Cliquez ici pour sauver votre Ame](#)



```
title = "Ceci est une erreur grave"  
message = "<A href=\"error.php\"> cliquez ici pour sauver votre Ame</A>"
```


twig basics : du côté php

- Pour utiliser twig dans l'application :

```
composer require "twig/twig :^3.0"
```

- Quand on veut utiliser 1 template :
 - créer un loader : il en existe plusieurs (file, arrays ...)
 - créer un environnement : quelques paramètres possibles

```
$loader = new Twig_Loader_Filesystem( '/path/to/dir' );  
$twig = new Twig_Environment( $loader, $params_array );
```

- charger un template :

```
$template = $twig->loadTemplate('index.html.twig');
```

- générer le rendu en passant des variables

```
echo $template->render([ 'the' => 'variables',  
                        'go'  => 'here' ]);
```

- raccourci :

```
$text= $twig->render('index.html.twig',  
                   [ 'the' => 'variables',  
                     'go'  => 'here' ] );
```

twig basics : du côté html

- Les variables dans un template :

`{{ foo }}`

`{{ foo.bar }}` : `foo` est un objet ou un tableau, contenant l'attribut ou l'entrée `bar`

```
<body>

<p> {{ message }} </p>
<a href="{{ link.href }}"> {{link.text}}</a>

</body>
```

```
$twig->render( 'tpl.html.twig', [
    'message' => 'Un bon moteur de template, twig' ,
    'link'    => [
        'href' => 'http://http://twig.sensiolabs.org/',
        'text' => 'site de twig' ]
    ] ) ;
```

- Filtres sur les variables :
 - un filtre est une fonction qui opère une transformation sur la valeur
 - exemple : mettre en majuscules/minuscules, formater une date, calculer une longueur ...
 - {{ variable | filtre }}

```
<body>
```

```
<p> {{ message | lower }} </p>
<a href="{{ link.href }}"> {{link.text | escape}}</a>

<p> {{ html_content | raw }} </p>
```

```
</body>
```

- Filtrer une partie de code :

```
{% filter upper %}
    Ce texte sera en majuscules, y compris {{ variable }}
{% endfilter %}
```

itérations

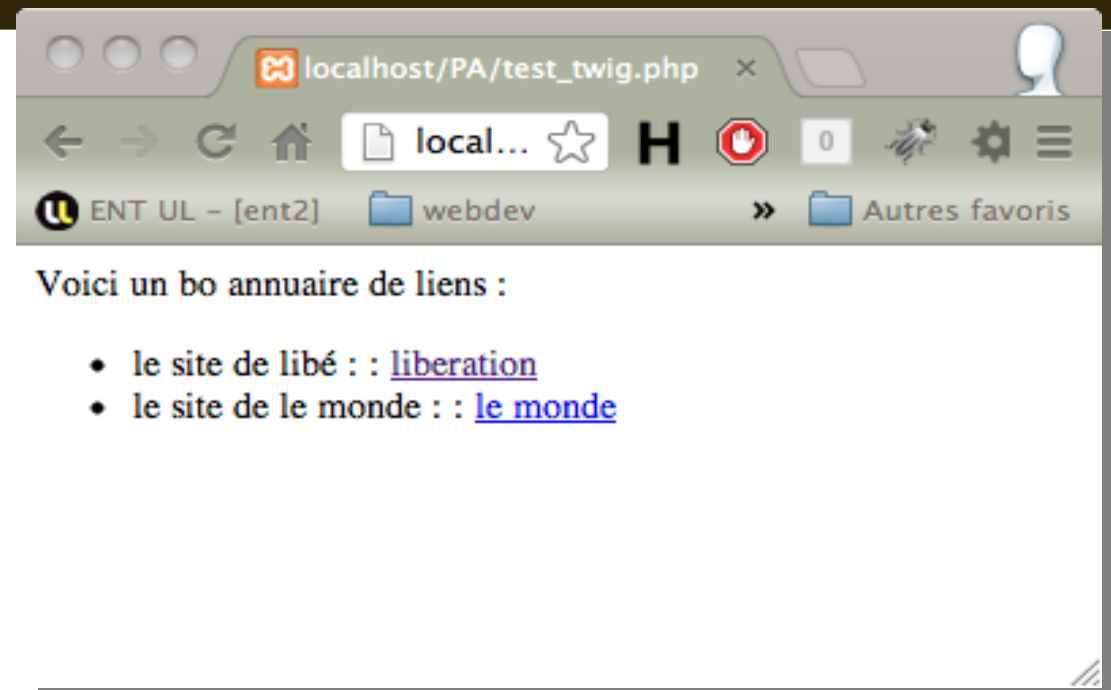
- Répéter une portion d'un template en fonction d'un tableau de valeurs
 - Lignes d'un tableau, éléments d'une liste ...

```
<ul>
  {% for it in list_links %}

    <li> {{ it.desc }} : <a href="{{ it.href }}">
      {{ it.text }}</a>
    </li>
  {% endfor %}
</ul>
```

- **list_links** : doit être liée à 1 tableau contenant des objets ou des tableaux

```
$tliens = [  
    [ 'href'=>'http://www.liberation.fr/' ,  
      'desc'=> 'le site de libé :',  
      'text'=>'liberation' ] ,  
  
    [ 'href'=>'http://www.lemonde.fr/' ,  
      'desc'=> 'le site de le monde :',  
      'text'=>'le monde' ]  
];  
  
$tmpl->display( [ 'list_links' => $tliens ] );
```



twig avancé : du côté php

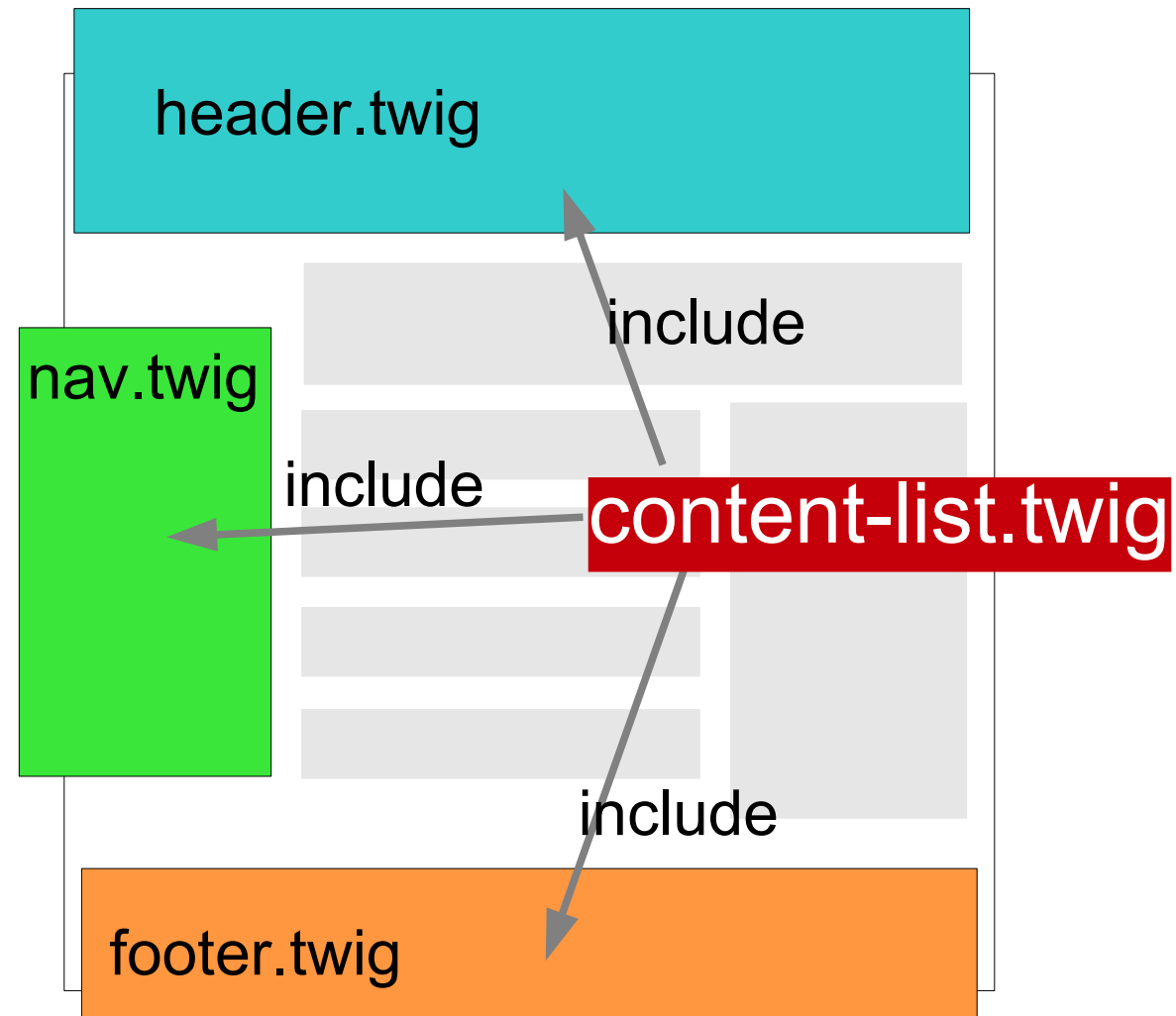
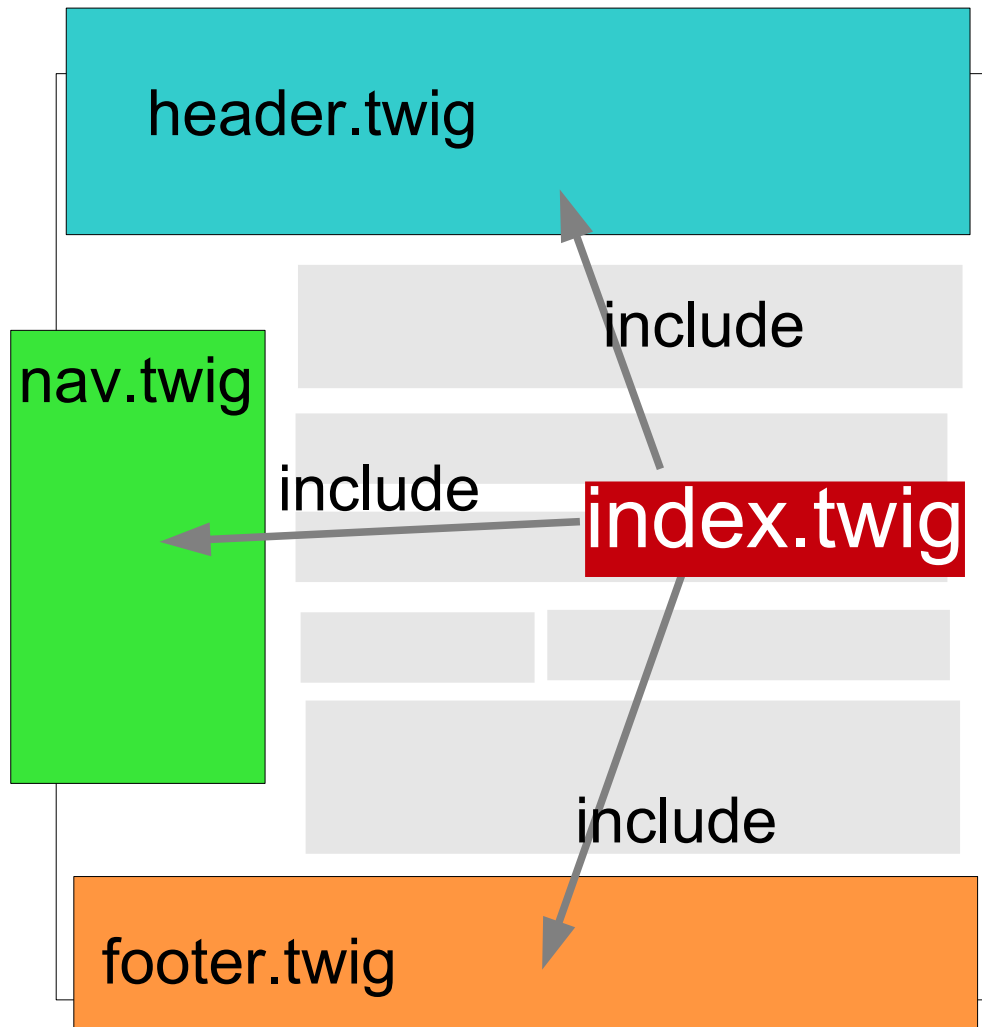
- Paramètres pour la création de l'environnement
 - `$twig = new Twig_Environment($loader, $params_array) ;`
 - **charset** : le charset utilisé par le template (défaut : utf-8)
 - **cache** : active le cache de templates compilés en donnant 1 chemin
 - **auto_reload** : true/false – pour recompiler les templates en cache en cas de changement dans le code – utile en dev si le cache est activé
 - **strict_variables** : true/false génère une exception/ignore les erreurs sur les variables (true en dev / false en prod), (défaut : false)
 - **autoescape** : false, true, css, url, html – désactive ou active l'auto-échappement sur les variables (défaut : true)

twig avancé : structuration des templates

- **inclusions** : inclure un template dans un autre, se fait dans le contexte courant – le template inclus voit les variable du template incluant

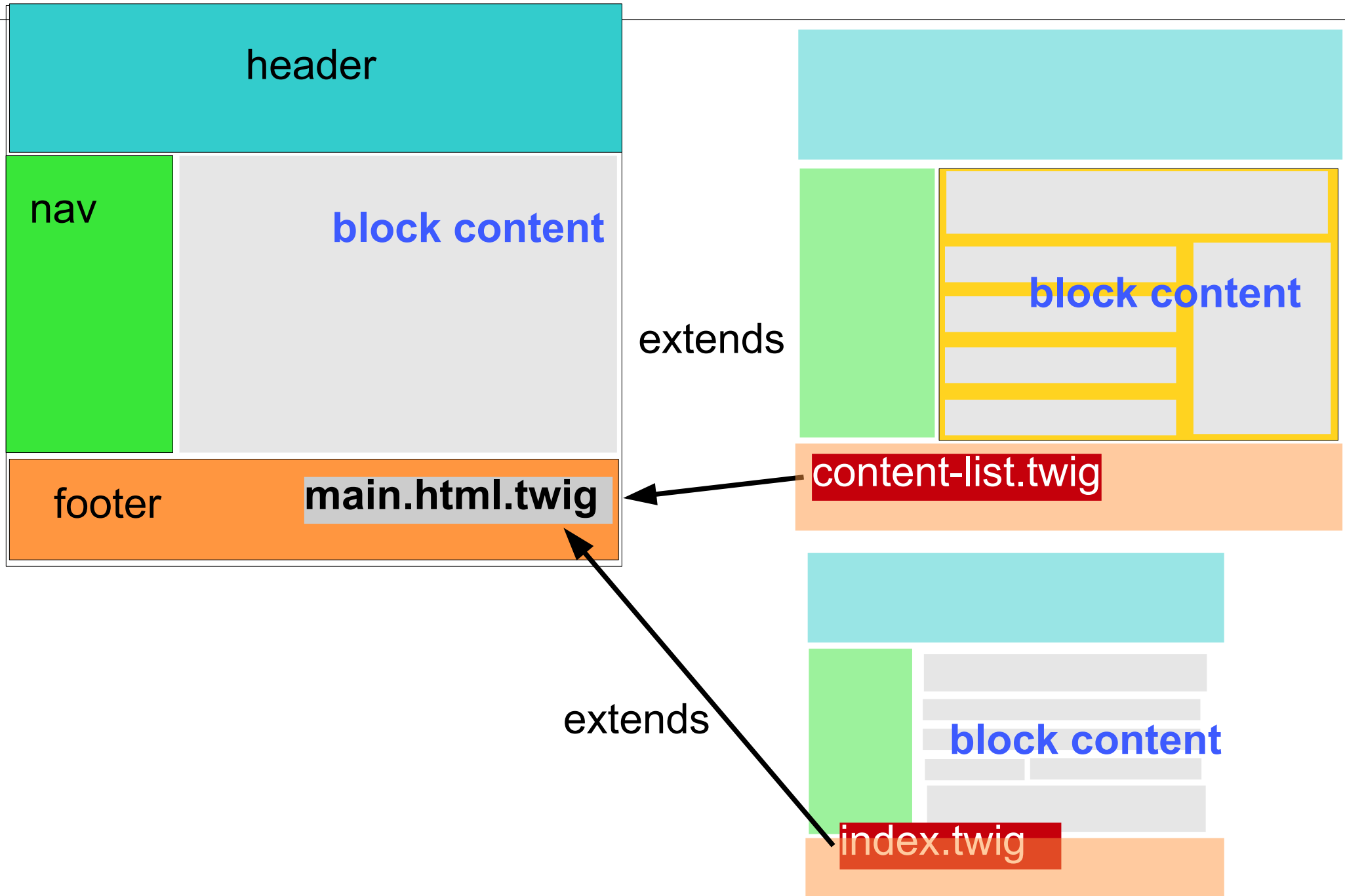
```
<ul>
  {% for it in list_links %}
    <li> {% include li_model.twig %} </li>
  {% endfor %}
</ul>
```


utilisation



- **blocks et extensions de templates**
 - un template peut définir des zones (blocks)
 - un template peut être défini comme une extension d'un autre :
 - Il étend le contenu du template d'origine,
 - et peut redéfinir ou étendre les blocks du template étendu

utilisation



```
DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="main-style.css" >
    ...
</head>
<body>
    {% include header.html.twig %}

    <section id= "main">
        {% block content %}
        {% endblock %}

    </section>

    {% include footer.html.twig %}
</body>
```

```
{% extends main.html.twig %}
```

```
{% block content %}  
    <h1> {{ m.titre }}</h1>  
    <p>{{ m.contenu }}</p>  
    <p>{{ m.auteur.nom }}</p>  
{% endblock %}
```

- conditionnelles

```
{% if list_links|length > 0 %}  
  <ul>  
    {% for it in list_links %}  
      <li> {{ it.desc }} :  
        <a href="{{ it.href }}">  
          {{it.text}}      </a>  
      </li>  
    {% endfor %}  
  </ul>  
{% endif %}
```

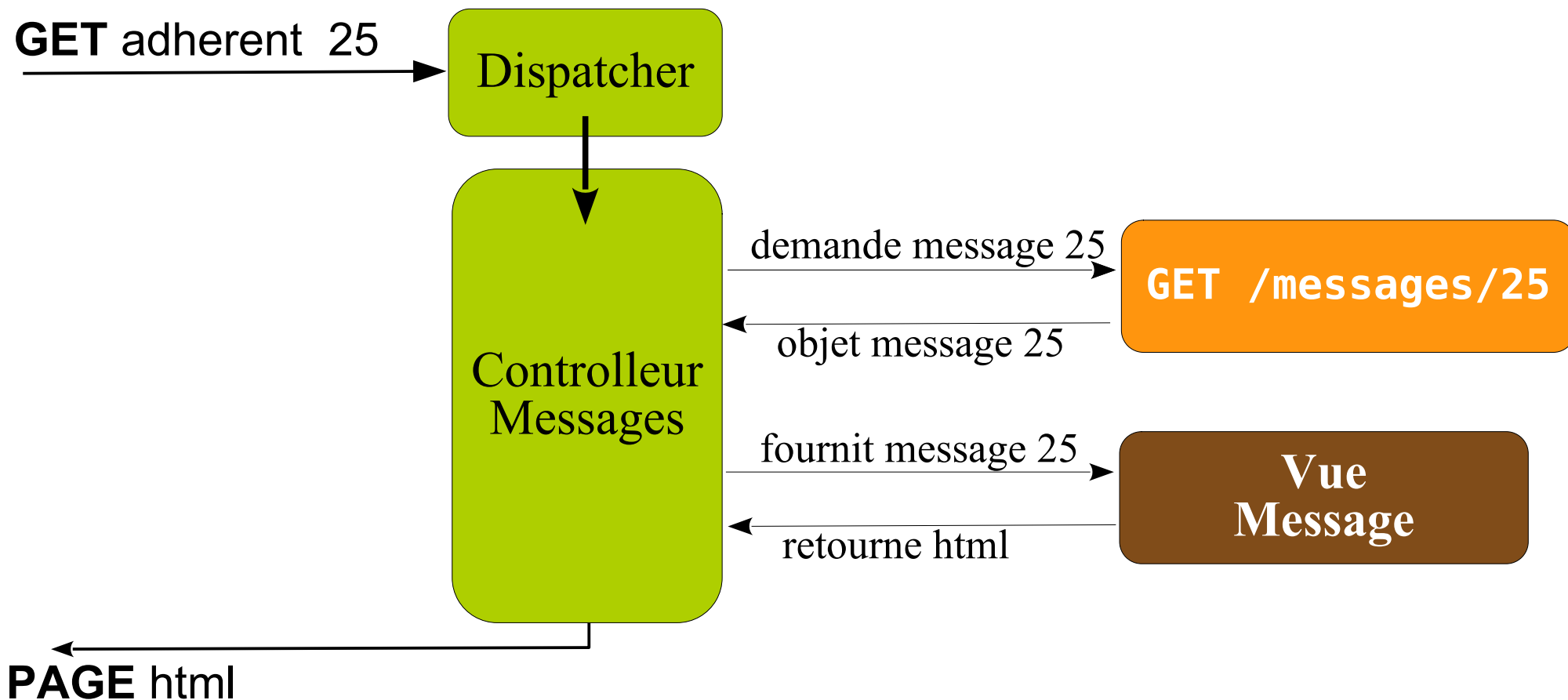
conclusion sur twig

- puissant et simple d'emploi
- le plus utile : variables, itérations, extensions
- utilisé dans symfony

intégration dans Slim

rappel : les vues

- Une vue = une mise en forme particulière de l'interface de l'application
- Créée par un controller pour répondre à une requête



les vues dans Slim

- 1 vue = 1 objet capable de répondre à la méthode `render()`
- créé par 1 service enregistré dans le container

```
$app->get('/hello/{name}', function ($rq, $rs, $ags) {  
    return $this->view->render($rs, 'profile.html',  
                                [ 'name' => $args['name'] ] );  
})->setName('profile');
```

intégration de Twig

- en utilisant le composant Twig-View pour créer le service de gestion des vues :

```
composer require "slim/twig-view"
```

```
$conf = [  
    'settings' => [ 'displayErrorDetails' => true,  
                    'tpl_dir' => './templates' ],  
    'view' => function( $c ) {  
        return new \Slim\Views\Twig(  
            $c['settings']['tpl_dir'],  
            [ 'debug' => true,  
              'cache' => $c['settings']['tpl_dir']  
            ]  
        );  
    }  
];  
  
$app = new \Slim\App(new \Slim\Container($conf));
```

Principe de base

- **NE JAMAIS DUPPLIQUER DU CODE HTML**
 - Complique le travail d'intégration
 - Rend risquée l'évolution de la présentation
 - Conduit à des incohérences de mise en forme
- **REUTILISER**
 - En utilisant le mécanisme de templates
 - En composant les objets générant la présentation