RESPONSE

RESPONSE HEADER

- API dédiée
  - $response->setCharset('ISO-8859-1');
  - $response→setStatusCode(Response::HTTP_NOT_FOUND);
  - ...
- Header brut
  - $response->headers->set('Content-Type', 'text/plain');

RESPONSE BODY

- Plusieurs contenus de réponses possibles
  - HTML (le plus habituel)
  - JSON (lorsque propose une api par exemple)
  - Binaire (download de fichier, image ...)
  - Stream (jouer de la musique ...)

RESPONSE (HTML/CSS/...)

- Directement
  - Response encapsule simplement le HTML

```php
use Symfony\Component\HttpFoundation\Response;
$rep = new Response();
$rep->setContent("<html>...");
echo $rep->getContent();
```

- Composé
  - C'est la meilleur façon d'organiser et de faire le rendu

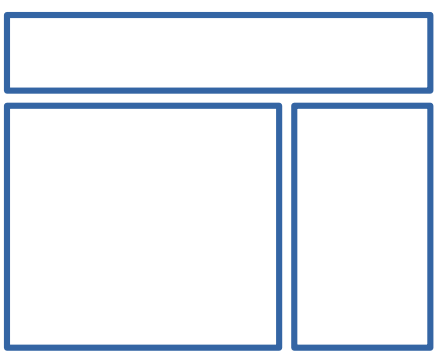
RENDU & TEMPLATE

Tableau de valeurs
DATA

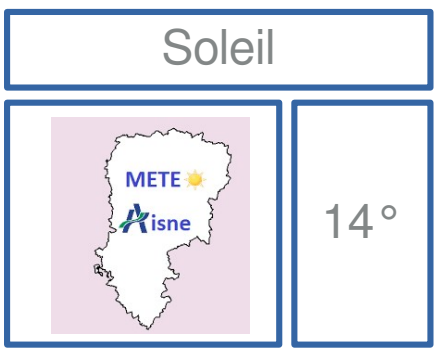
Titre	Soleil
Carte	lundi.jpg
Temp	14

+

Affichage
HTML+CSS



=



EXEMPLE TWIG

```php
$load = new Twig\Loader\FilesystemLoader( __DIR__.'/views/');
$env = new Twig\Environment($load);

echo $env→render(
    'firstTwigTemplate.html.twig',
    array('msg' => 'Mise en place d\'un mécanisme de template')
);
```

Voir commit

- Sépare bien le front du back
  - Par utilisation d'une syntaxe différente
- Gère des problématiques front
  - Les caractères espaces
  - L'échappement des caractères HTML
  - La possibilité de faire des fonctions/filtres qui n'affectent que certain templates

RÉFÉRENCES TWIG

https://twig.symfony.com/doc

Twig Reference

Browse the online reference to learn more about built-in features.

Tags

apply
autoescape
block
deprecated

Filters

abs
batch
capitalize
column
format_number
format_time
inky
inline_css
reverse
round
slice
sort

Functions

attribute
block
constant
cycle

Tests

constant
defined
divisibleby
empty

Twig Reference from Symfony

Symfony provides many more features via the `symfony/twig-bridge` Composer package.

Tags

form_theme
stopwatch
trans

Filters

abbr_class
abbr_method
file_excerpt

Functions

absolute_url
asset
asset_version
render
render_esi
url

Tests

rootform
selectedchoice

TWIG SYNTAXE

- {{ ... }}
  - Pour afficher le contenu de variable
- {% ... %}
  - Pour des instructions (test, boucle ...)
  - https://twig.symfony.com/doc/2.x/tags/index.html
- {# ... #}
  - Commentaire

TWIG SYNTAXE

- Où mettre ses templates
  - **twig.default_path** dans config/packages/twig.yaml
- Nommer ses templates
  - **snake case** (ex :blog_posts.twig, admin/default_theme/blog/index.twig)
  - Mettre deux extension (html.twig , xml.twig ...)
- Accès aux variables exemple : `client.name`

```
$client['name'] puis $client->name puis $client->name()
puis $client->getName() puis  $client->isName() puis $client->hasName()
                                    null
```

ACCÉDER À TWIG DANS VOTRE CONTROLLER

- Par héritage AbstractController

```php
class ProductController extends AbstractController
{
    public function index()
    {
        return $this->render('product/index.html.twig',
```

- Par injection de dépendance
  - (cf. service container)

Twig Ref symfony utile

- URL

```
@Route("/", name="client_signup")
<a href="{{ path('client_signup') }}">Homepage</a>
```

- Avec Paramètre Query

```
@Route("/client/{id}", name="client_info")
<a href="{{ path('client_info', {id: client.id}) }}">{{ client.name }}</a>
```

- IMG, CSS, JS => asset

```
<img src="{{ asset('images/logo.png') }}"/>
<img src="{{ absolute_url(asset('images/logo.png')) }}"/>
```

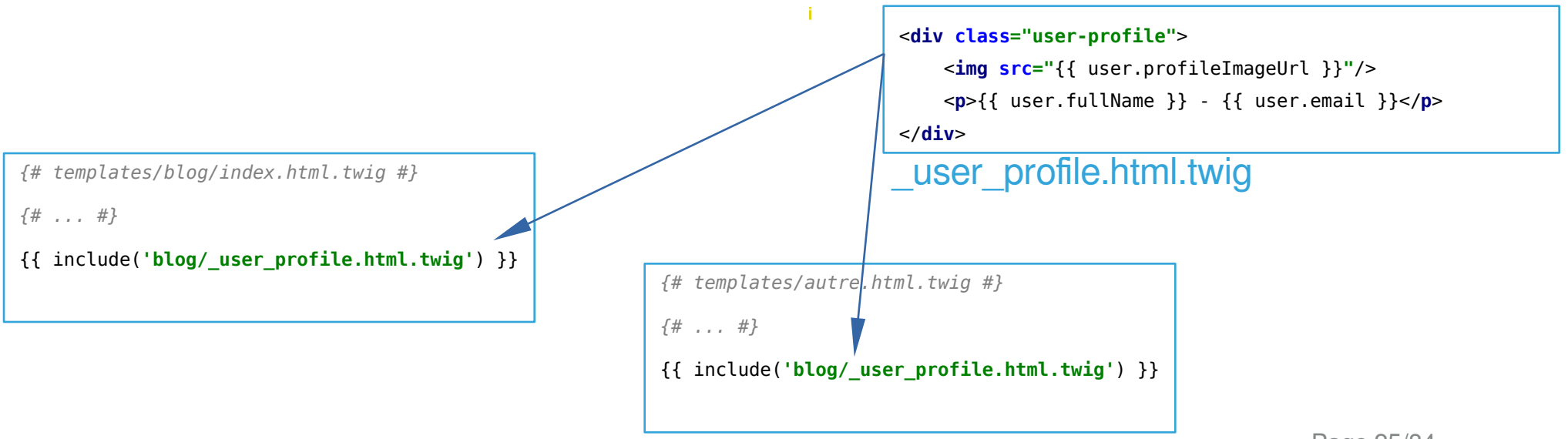
Twig Globale Variable APP

- app.user
  - Si il y a une authentification c'est l'objet user avec les infos de l'utilisateur connecté
- app.request
- app.session
  - Session en cour, null sinon
- app.flashes
  - Array avec les messages flashs enregistrés en session
- app.environment
  - Configuration environment (dev, prod, etc).
- app.debug
  - True si en mode debug, false sinon
- app.token
  - Un objet représentant la sécurité par token. (Il implémente l'interface TokenInterface)

Accessible partout
dans vos templates

INCLUDE VS. EMBEDDING VS EXTEND

- Include
  - Réutilisation du même code à plusieurs endroit



INCLUDE VS. EMBEDDING VS EXTEND

- Embedding (associé au cache)
  - Intégrer le résultat du rendu plutôt que de refaire tout
  - Pertinent lorsque l'exécution est coûteuse → accès DB

```
<div id="sidebar">
    {{ render(path('latest_articles', {max: 3})) }}
    {{ render(url('latest_articles', {max: 3})) }}
</div>
```

Url visible
Doit avoir
une route

```
{{ render(controller(
    'App\\Controller\\BlogController::recentArticles', {max: 3}
)) }}
```

Appel
direct du
controller

COMPARAISON EMBEDDING , INCLUDE

- Identique
  - Les deux ont été ajoutés à twig pour rendre des extraits de code HTML réutilisables.
  - Les deux sont utilisés pour la séparation fonctionnelle des modèles (pied de page, en-tête, ...)
  - Les deux sont dynamiques.
- Différent
  - Inclure est utilisé pour encapsuler du code qui a une structure HTML non flexible.
  - Intégrer permet la flexibilité.
- Conclusion
  - Inclure ne doit être utilisé que lorsque vous devez diviser un modèle en plusieurs sous-modèles fonctionnels et réutiliser le code encapsulé ailleurs.
  - Embed est utilisé lorsque vous devez personnaliser les modèles réutilisables.

INCLUDE VS. EMBEDDING VS EXTEND

- Extends
  - Permet de mettre en place la ligne graphique → Layout
  - Layout déclare des éléments appelés {%bloc pouvant être définit plus tard

```twig
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}My Application{% endblock %}</title>
</head>
<body>
<div id="content">
    {% block body %}{% endblock %}
</div>
</body>
</html>
```

```twig
{% extends '/layout.html.twig' %}

{% block title %}List Clients{% endblock %}

{% block body %}
    <ul>
    {% for client in clients %}
        <li><b>{{ client.nom }}</b> {{ client.prenom }}</li>
    {% endfor %}
    </ul>
{% endblock %}
```

Autres types de responses JSON, FILE ...

- Grace à des helpers hérités de AbstractController
  - JSON : `return $this->json(['username' => 'jane.doe']);`
  - File : `return $this->file('/path/to/some_file.pdf');`
- D'autres méthodes utiles (helpers)
  - URL `$url = $this->generateUrl('app_lucky_number', ['max' => 10]);`
  - Config Value : `$contentsDir = $this->getParameter('kernel.project_dir').'/contents';`
  - Et d'autres : forward, redirect, addFlash, isGranted, form, doctrine ...