# Symfonyの機能

Symfonyは、コントローラーで便利に使える多くの機能を提供しています。
ここでは「**HomeController.php**」ファイルに戻って確認してみましょう。

## ルートパラメーター

ルーターの便利な機能の1つは、GETパラメーターを使わず、URLから直接データを取得できることです。
もちろん、Symfonyでも可能です。

次のようなルートを作成してみましょう：

```
\#[Route("/bonjour/{nom}/{prenom}", name: "app_bonjour")]
public function bonjour($nom, $prenom, Request $request): Response
{
    dump($request);
    // dd($request);
    return $this->render("home/bonjour.html.twig", [
        "nom" => $nom,
        "prenom"=> $prenom
    ]);
}
```

このとき、使用するクラス（ここでは「Request」）には、対応する「use」宣言を忘れずに記述しましょう：

```
use Symfony\Component\HttpFoundation\Request;
```

ルートの中の中括弧（{}）は、メソッド引数として同じ名前で受け取れるパラメーターを定義します。

ルートパラメーターはさらに発展的な使い方もありますが、それは後ほど解説します。

対応するテンプレートも作成しましょう。最低限、次のような内容を含んでいればOKです：

```
{% extends 'base.html.twig' %}
{% block body %}
    <h2>こんにちは {{prenom ~ " " ~ nom}}</h2>
{% endblock %}
```

Twigにおけるチルダ（~）は文字列の連結を意味します。

## 重要な関数とクラス

さて、第3引数である `request` に注目してみましょう。

Symfonyでは、メソッドの引数にクラスを指定すると、自動的にそのクラスのインスタンスを生成し、引数に代入してくれます。
今回は次のように指定しました：

```
use Symfony\Component\HttpFoundation\Request;
```

### dumpとdd

Symfonyの *dump()* 関数を使うと、オブジェクトの中身を調べることができます。
ですがページを読み込んでも何も表示されないことがあります…どこに出力されたのでしょう？

実は、開発モード（dev）では、Symfonyは画面下部にツールバーを表示しています。
このツールバーには便利な開発情報が含まれており、左側にあるターゲットアイコンをホバーすると *dump()* の出力が見られます。インタラクティブでカラフルな表示です。

文字列、数値、配列など、さまざまな値を *dump()* してみてください。PHPの標準 `var_dump()` よりも遥かに便利です。

また、*dd()* という関数もあります。これは「dump and die（ダンプして終了）」を意味し、表示した後に即座にスクリプトの実行を停止します。この場合、出力は直接ブラウザ上に表示されます。

**Twig** にも *dump()* が使え、結果は直接ブラウザに表示されます。

### Requestクラス

前述の `Request` オブジェクトをダンプした結果を確認してみましょう：

- `attributes`: ルートに関する情報（パラメーターなど）を含む。
- `request`: `$_POST` データ。
- `query`: `$_GET` データ。
- `server`: `$_SERVER` データ。
- `files`: `$_FILES` データ。
- `cookies`: `$_COOKIE` データ。
- `session`: `$_SESSION` データ。
- その他：HTTPメソッド、URI、言語など。

つまり、RequestはPHPのスーパーグローバルに加えて、Symfony独自の情報もまとめて管理するオブジェクトです。

| PHPのスーパーグローバル | Symfonyでの書き方 | |---|---| | `$_GET` | `$request->query` | | `$_POST` | `$request->request` | | `$_SERVER` | `$request->server` | | `$_FILES` | `$request->files` | | `$_COOKIE` | `$request->cookies` | | `$_SESSION` | `$request->getSession()` |

## パラメーター付きルートの注意点

### ルートの定義順序

注意点：
もし今「`/bonjour/anglais/{username}`」のようなルートを `bonjour` の後に定義した場合、そのルートに

は到達できなくなります。

```
\#[Route("/bonjour/anglais/{username}", name: "app_hello")]
public function hello($username): RedirectResponse
{
    dd("Hello ".$username);
    return $this->redirectToRoute("app_bonjour");
}
```

*redirectToRoute()* メソッドは、指定したルート名にリダイレクトするために使います。

Symfonyは、ルートを定義された順に評価していきます。
つまり、`/bonjour/anglais/pierre` にアクセスすると、「bonjour」ルートにマッチし、`nom = anglais`、`prenom = pierre` になります。

より具体的なルートを前に書けば、どちらのルートも正しく機能します。ただし、「anglais」が実際に `nom` に使われていなければの話です。

このマッチングの詳細は、Symfonyツールバーの「Routing」セクション（画面左下）で確認できます。

## Twigでパラメーター付きルートを生成

`path()` 関数を使えば、ルート名からURLを生成できます。
パラメーター付きの場合は、連想配列で渡します：

```
<a href="{{ path("app_bonjour", {nom: "Fontaine", prenom: "Jean"}) }}">Jean
Fontaine</a>
<a href="{{ path("app_hello", {username: "John"}) }}">John Doe</a>
```

コントローラーでのリダイレクトも同様です：

```
// dd("Hello ".$username);
return $this->redirectToRoute("app_bonjour", ["nom"=>"smith",
"prenom"=>$username]);
```

## ルートのプレフィックス

Symfonyでは、コントローラー単位でルートのプレフィックスを設定できます：

```
\#[Route('/user')]
class UserController
{
  \#[Route('/profil', name:"profil")]
  public function profil(): response{}
}
```

この例では、`/user/profil` が最終的なルートパスとなります。
このコントローラー内の他のすべてのルートも「/user」で始まります。
これは後ほどユーザーのCRUDを作る際にも活用します。

### デフォルト値付きのルート

ルートパラメーターには、関数の引数と同様にデフォルト値を設定できます：

```
\#[Route("/bonjour/anglais/{username}", name: "app_hello", defaults:
["username"=>"John"])]
```

または、短縮記法で書くことも可能です：

```
\#[Route("/bonjour/{nom}/{prenom?Jean}", name: "app_bonjour")]
```

注意：複数のパラメーターがある場合、デフォルト値を設定できるのは右側の引数のみです。

### 正規表現によるルート制限

現在は、ルートパラメーターにどんな文字列でも入ってしまいます（例：「55 22」）。
これを正規表現で制限できます：

```
\#[Route("/bonjour/anglais/{username}",
    name: "app_hello",
    defaults: ["username"=>"John"],
    requirements: ["username"=>"^[a-zA-Z]+$"])]
```

これで数字などを含めたアクセスは別ルートに飛ばされます。

短縮記法でも書けます：

```
\#[Route("/bonjour/{nom<^[a-zA-Z]+$>}/{prenom<^[a-zA-Z]+$>?Jean}", name:
"app_bonjour")]
```

このようにして、URLに許可する値を厳密に制御できます（例：英字のみ、スラッグ、IDなど）。

## Symfonyの開発ツールバー

ダンプやルーティング以外にも、次の情報が表示されます：

- リクエストのステータス、マッチしたルート
- ページ読み込み時間
- メモリ使用量

- ログインユーザー（後述）
- Twigテンプレートのレンダリング情報
- dumpされた内容

右側には：

- サーバー情報
- Symfonyのバージョン
- ツールバーの表示／非表示切り替えボタン

このツールバーは本番環境では表示されません。
本番環境にデプロイする際には環境を忘れずに切り替えましょう。

## セッションの管理

Symfonyには独自のセッション管理があります。
「bonjour」ページに訪問カウンターを追加してみましょう：

```
$sess = $request->getSession();

if($sess->has('nbVisite')) $nb = $sess->get("nbVisite")+1;
else $nb = 1;

$sess->set("nbVisite", $nb);
```

- `has()`：キーの存在確認
- `get()`：値の取得（デフォルト値を渡すことも可能）
- `set()`：値の設定
- `remove()`：キーの削除

Twigでカウンターを表示：

```
<h3>すでに {{ app.session.get("nbVisite") }} 回訪問しています！</h3>
```

### フラッシュメッセージ

Symfonyでは、簡単に一時的なメッセージ（フラッシュメッセージ）を表示できます。

`bonjour` コントローラーで追加してみましょう：

```
$this->addFlash("bonjour", "こんにちは $prenom $nom さん！");
```

`hello` コントローラーでも：

```

```

```
$this->addFlash("redirect", "リダイレクトされました！");
$this->addFlash("bonjour", "こんにちは $username さん！");
```

テンプレートでは次のように表示します：

```
{% for message in app.flashes('bonjour') %}
    <div>{{ message }}</div>
{% endfor %}
```

フラッシュメッセージは一度表示されると自動的に消えます。

複数のカテゴリをまとめて取得することも可能です：

```
{% for categorie, messages in app.flashes(['bonjour', "redirect"]) %}
  {% for message in messages %}
    <div>
      <strong>{{ categorie }} :</strong>
      <br>
      {{ message }}
    </div>
  {% endfor %}
{% endfor %}
```

全カテゴリを取得するには：

```
{% for categorie, messages in app.flashes %}
```

最後に、このフラッシュメッセージのブロックを `base.html.twig` に移して再利用できるようにしましょう：

```
{% for label, messages in app.flashes %}
  {% for message in messages %}
    <div class="alert alert-{{ label }}">
      {{ message }}
    </div>
  {% endfor %}
{% endfor %}
```

このデザインは、後でBootstrapで整えましょう。