BTS SIO :: SLAM :: PARTIE 1

FRAMEWORK

Version 1.1

FRAMEWORK

- Utiliser un framework permet de se concentrer sur la valeur ajoutée réelle
- Les fondations d'une application web sont laissées à une communauté compétente
- Un framework donne un cadre de travail (Traduction)
- C'est une méthode et des choix architecturaux
- Cela permet de s'affranchir de pas mal de tâches

LARAVEL

- Créé en 2011 par Taylor Otwell
- Visibilité mondiale en 2013 avec la version 4
- Certains composants sont issus de Symfony et d'autres conçus pour Laravel
- Il utilise PHP
- Le nom Laravel vient de l'univers de Narnia (Cair Paravel)

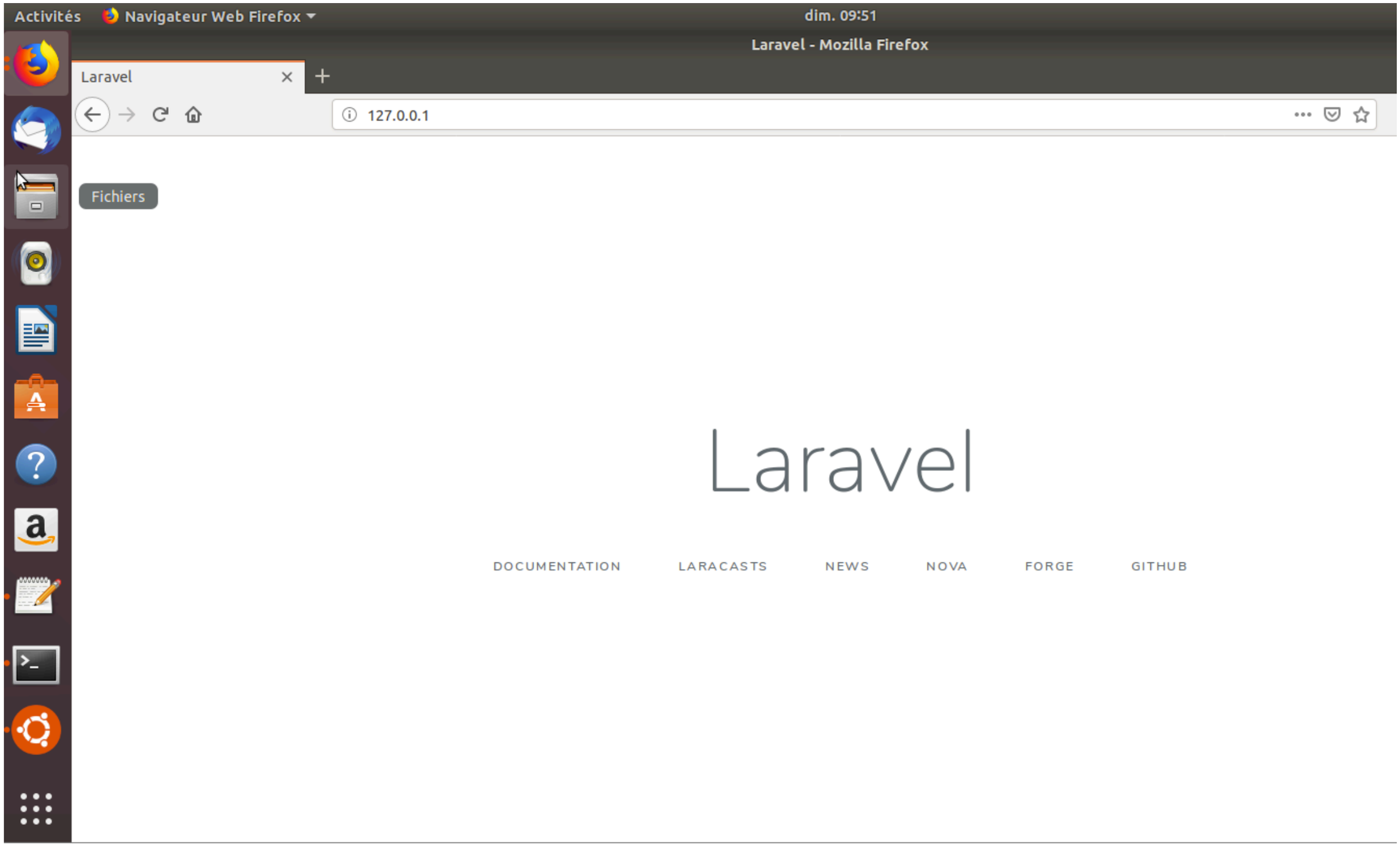
BTS SIO : SLAM : LARAVEL

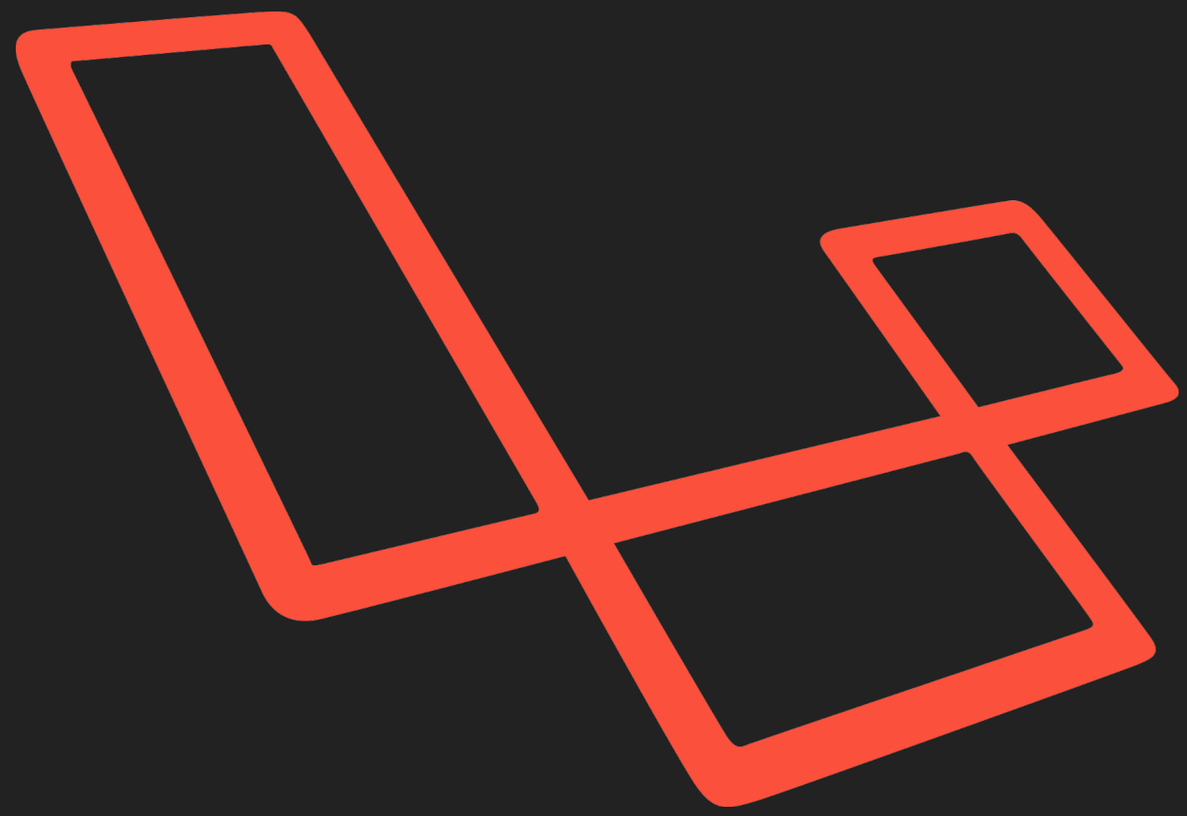
INSTALLATION

INSTALLATION

- Verifier que VirtualBox est bien installé sur votre machine (sinon le faire…)
- Installer Ubuntu 18.04
- Suivre les étapes indiqué sur ce site : https://www.howtoforge.com/tutorial/install-laravel-on-ubuntu-for-apache/

INSTALLATION : OK !





CHAP 1

PREMIER PAS...

STRUCTURE DE L'APPLICATION

```
(base) macbook-de-seb:nouvelle-star sebastien$ ls -1
app
artisan
bootstrap
composer.json
composer.lock
config
database
package.json
phpunit.xml
public
readme.md
resources
routes
server.php
storage
tests
vendor
webpack.mix.js
(base) macbook-de-seb:nouvelle
```

- app : les fichiers PHP (classes)
- bootstrap : fichiers lancés par Laravel à chaque page (initialisation)
- config : configuration du site en ligne
- database : pour les migrations avec Eloquent
- public : point d'entrée de l'application pour le serveur.
- resources : les vues, gabarits, javascript (tout ce qui est pas PHP)
- routes : liste des routes de l'application (voir chapitre suivant)
- storage : logs, cache, versions compilées des fichiers du framework
- tests : pour les tests unitaire
- vendor : les dépendances de l'application : bibliothèque.

CONFIGURATION

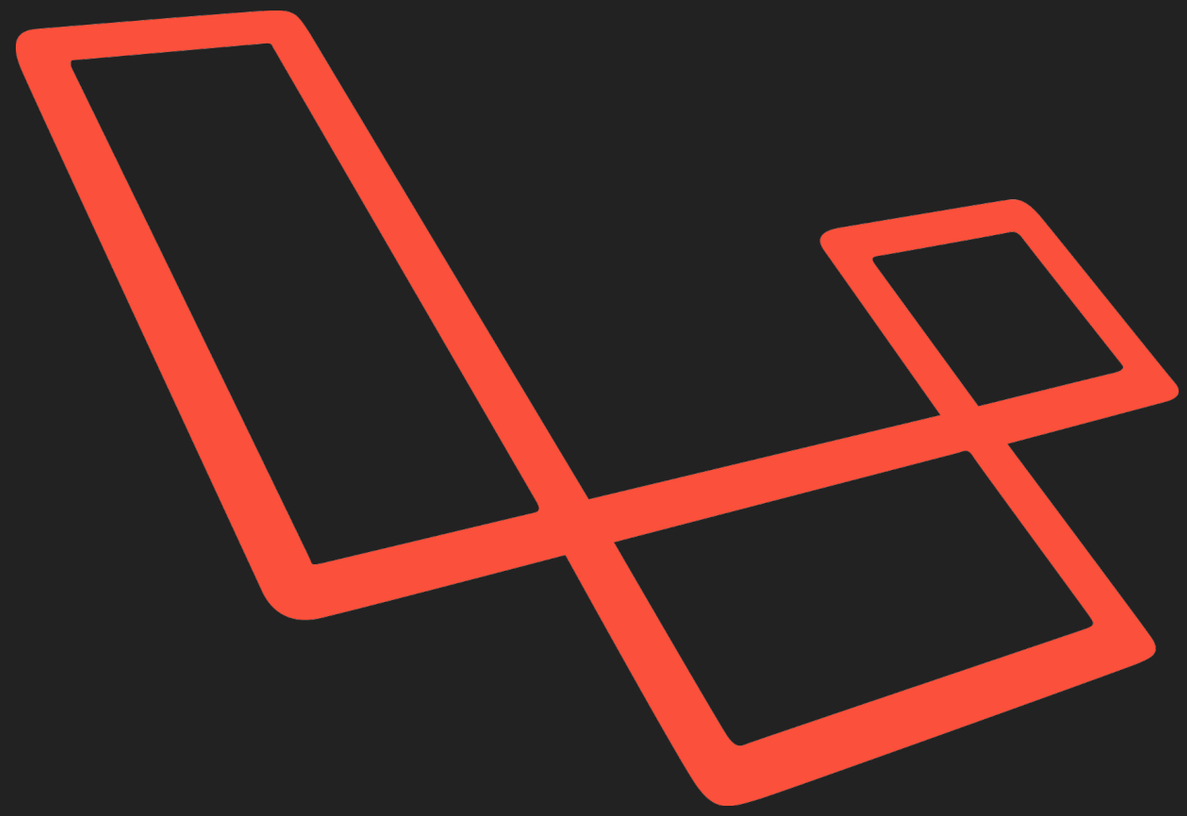
- Il existe un fichier .env qui contient la configuration locale de Laravel (Git ne le prend donc pas en charge !!! -> .gitignore)
- Rappel : un fichier qui commence par un . est invisible par défaut. Il faut faire ls -a pour le voir.
- Il est important de bien comprendre que .env ne gère que la configuration local pour la mise en service ça sera le répertoire **config**

MODIFICATION

- On va modifier le .env
- On modifie que APP_NAME et APP_URL
- On s'occupera de l'installation de MySQL et de son paramétrage un peu plus tard.

```
sebastien@sebastien-VirtualBox:/var/www/html/NouvelleStar$ ls -a
.           composer.json  .env              phpunit.xml  server.php
..          composer.lock  .env.example      public       storage
app         config         .gitattributes    readme.md    tests
artisan     database       .gitignore        resources    vendor
bootstrap   .editorconfig  package.json      routes       webpack.mix.js
sebastien@sebastien-VirtualBox:/var/www/html/NouvelleStar$ sudo gedit .env
```

```
APP_NAME=NouvelleStar
APP_ENV=local
APP_KEY=base64:gJXBDp/hoeI9uCb6Jdjoh/3ussiokoRpHTvTh9nESFY=
APP_DEBUG=true
APP_URL=http://nouvelle-star.dev
```



CHAP 2

LES ROUTES...

LES ROUTES

- Associer une URI à un code à exécuter
- La liste des routes se trouvent dans routes/web.php
- Voici le fichier web.php

```
sebastien@sebastien-VirtualBox:/var/www/html/NouvelleStar/routes$ cat web.php
<?php

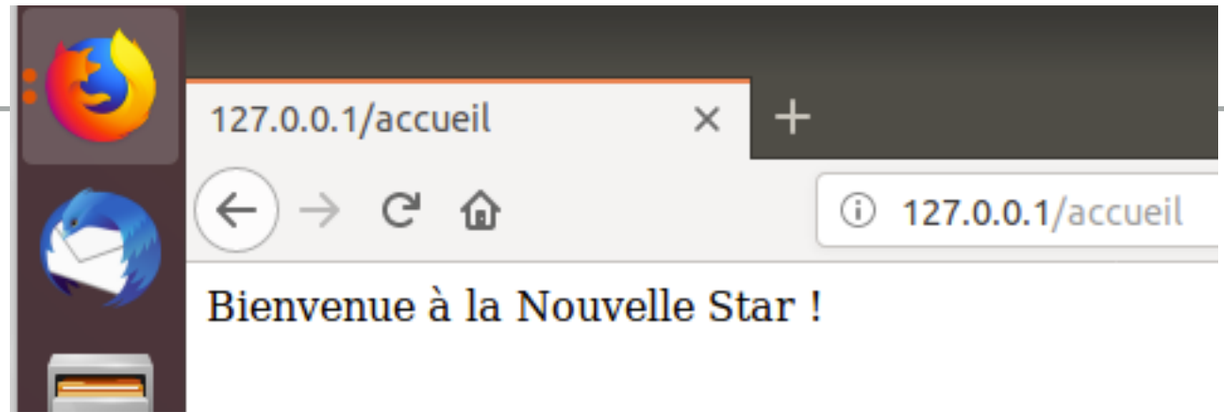
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});
```

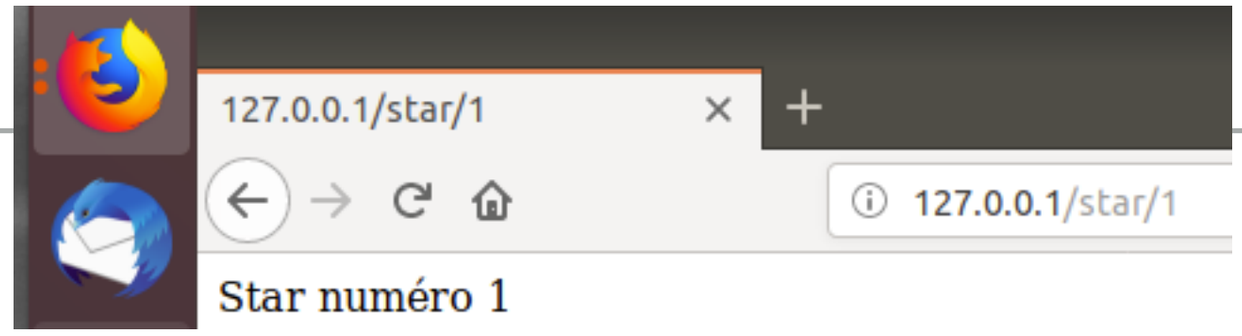
LES ROUTES

- Il existe plusieurs méthodes pour la classe Route
  - get, post, put, delete (méthodes HTML)
  - Ce qui se traduit par Route::post($uri, function() {…}
    Route::put($uri, function() {…} , etc.
- La fonction peut avoir du code logique PHP
- Cette fonction est dite anonyme (elle a pas de nom ^^)
- Cette fonction peut retourner une vue ou simplement du texte : return « Bonjour le monde !»

LES ROUTES :: TAF



- En étudiant la syntaxe de web.php ajouter une route qui affiche « Bienvenue à la Nouvelle Star ! » si l'on a l'URI **/accueil**.
- Ajouter une route **/a-propos** qui affichera juste « C'est la page à propos »
- Ajouter une route **/stars/sophie** qui affiche « Page de la nouvelle star sophie ! »
- Important : retourner du texte sans passer par une vue (on verra les vues au chapitre suivant)
- Un *return* de texte suffit donc pour cette fois



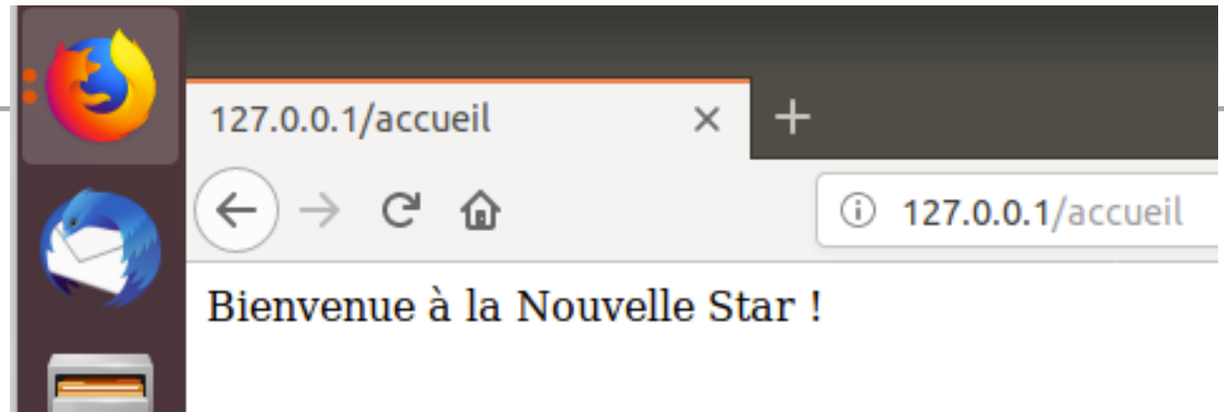
LES ROUTES :: PARAMETRES

- On peut passer des paramètres aux routes. Imaginons que /star/1 affiche la star Xavier
  /star/2 affiche la star Mathieu
  etc.

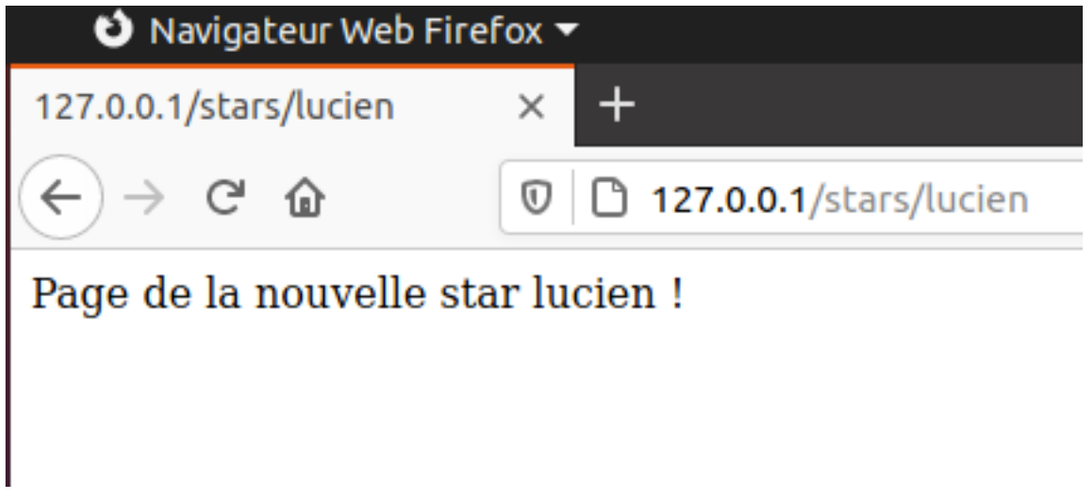
- On ne va pas créer autant de route qu'il y a d'occurence de star. La solution ? Passer le paramètre (1 ou 2 dans l'exemple)

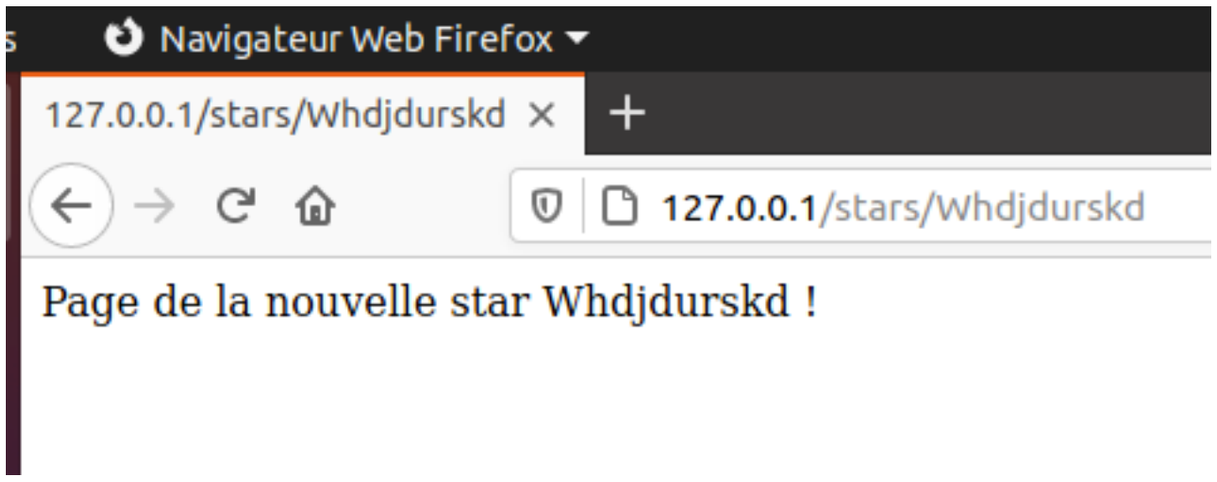
- ```
  Route::get('/star/{id}', function ($id) {
          return "Star numéro ". $id;
  });
  ```
  ```

LES ROUTES :: TAF



- Créer une route pour que quelque soit le prénom cela fonctionne. C'est à dire que nous aurons Page de la nouvelle star … avec le prénom.
- Exemple : /stars/lucien /stars/laure etc.



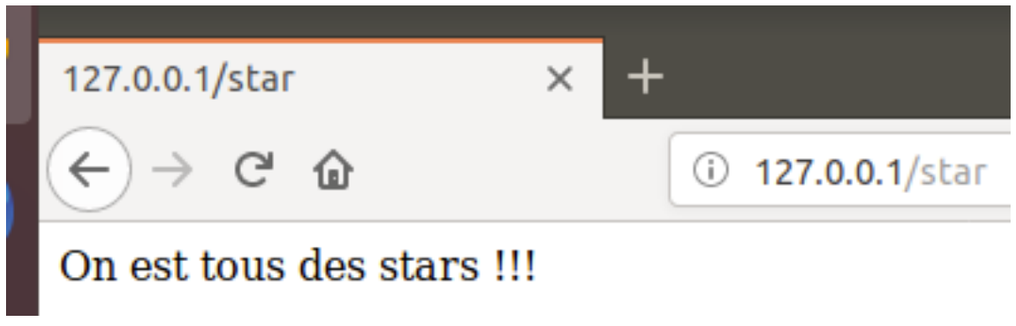


LES ROUTES :: PARAMETRES

- Si un paramètre est facultatif il suffit de mettre ? derrière

- Exemple :

```php
Route::get('/star/{id?}', function ($id = null) {
        if ($id == null) {
                return  "On est tous des stars !!!";
        } else {
                return "Star numéro ". $id;
        }
});
```



LES ROUTES :: TYPES DE REPONSES

- Dans les exemples précédents la fonction de rappel (*callback)* renvoyait une chaine de caractères.

- On peut aussi retourner du HTML grâce au système de vue (que nous verrons juste après)

- Du JSON

- Un fichier

LES ROUTES :: REPONSES TEXTUELLES

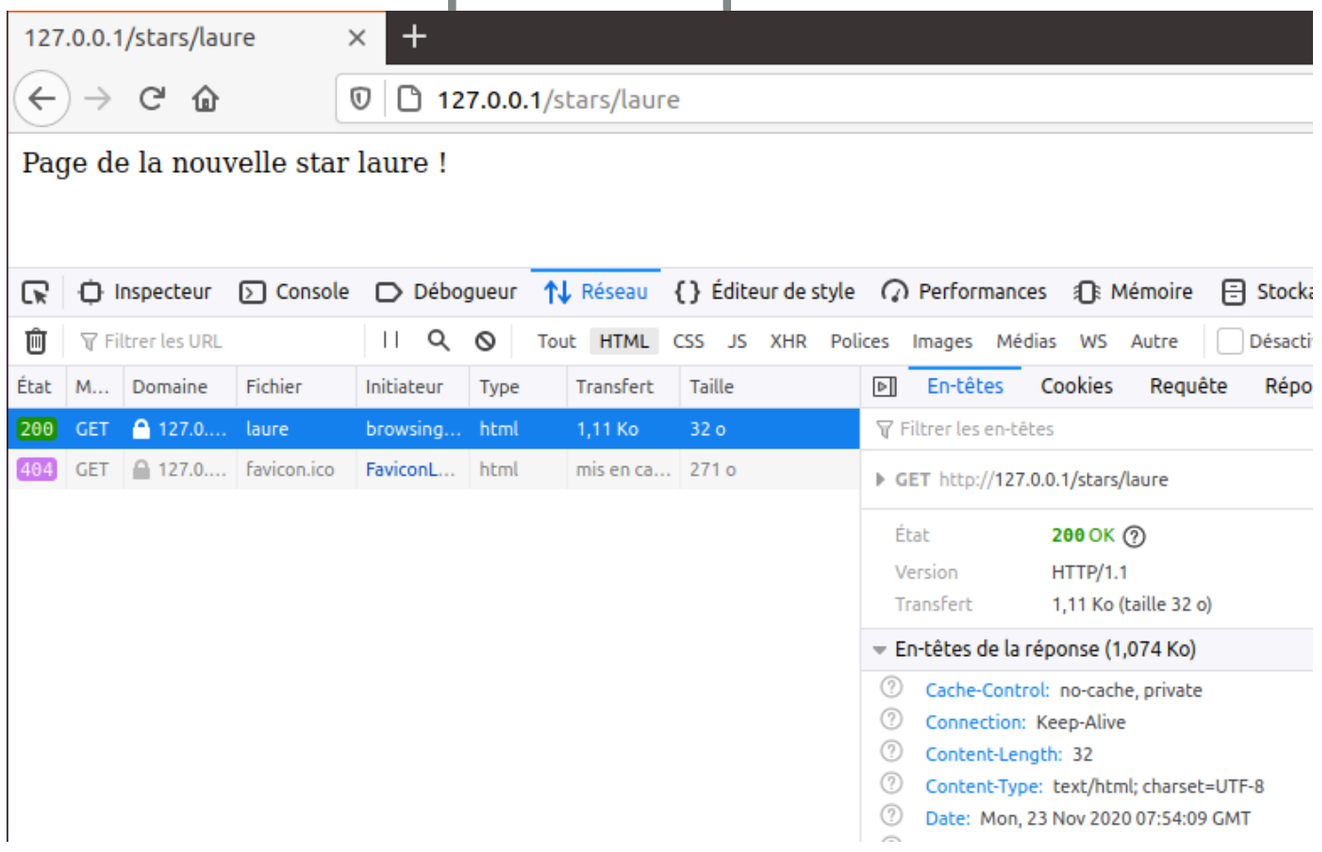
- En fait même quand vous retournez que du texte, le framework transforme cette chaîne en une réponse HTTP.
  - Il utilise la fonction response qui génère un objet
  - Il ajoute le code 200 (statut qui indique que la requête c'est bien déroulée)
  - Et un **Content-Type text/HTML** (même si c'est du texte c'est le choix par défaut)

LES ROUTES :: REPONSES TEXTUELLES

- Si l'on devait bien faire les choses on pourrait faire ceci :

```php
Route::get('/bonjour', function () {
    return response('Bonjour !',200)
    ->header('Content-Type', 'text/plain');
});
```

- Voir l'entête en utilisant les outils Inspecter puis Réseau



LES ROUTES :: REPONSES HTML

- On n'a pas grand chose à faire car Laravel retourne du code HTML par défaut.

- Ce qui veut dire qu'au lieu de mettre simplement du texte vous pouvez mettre du code HTML

- **TAF** : Modifier l'accueil en mettant une balise <html><body><h1>

- Il est fastidieux de faire cela directement. On utilisera pour cela les vues. (chapitre suivant)

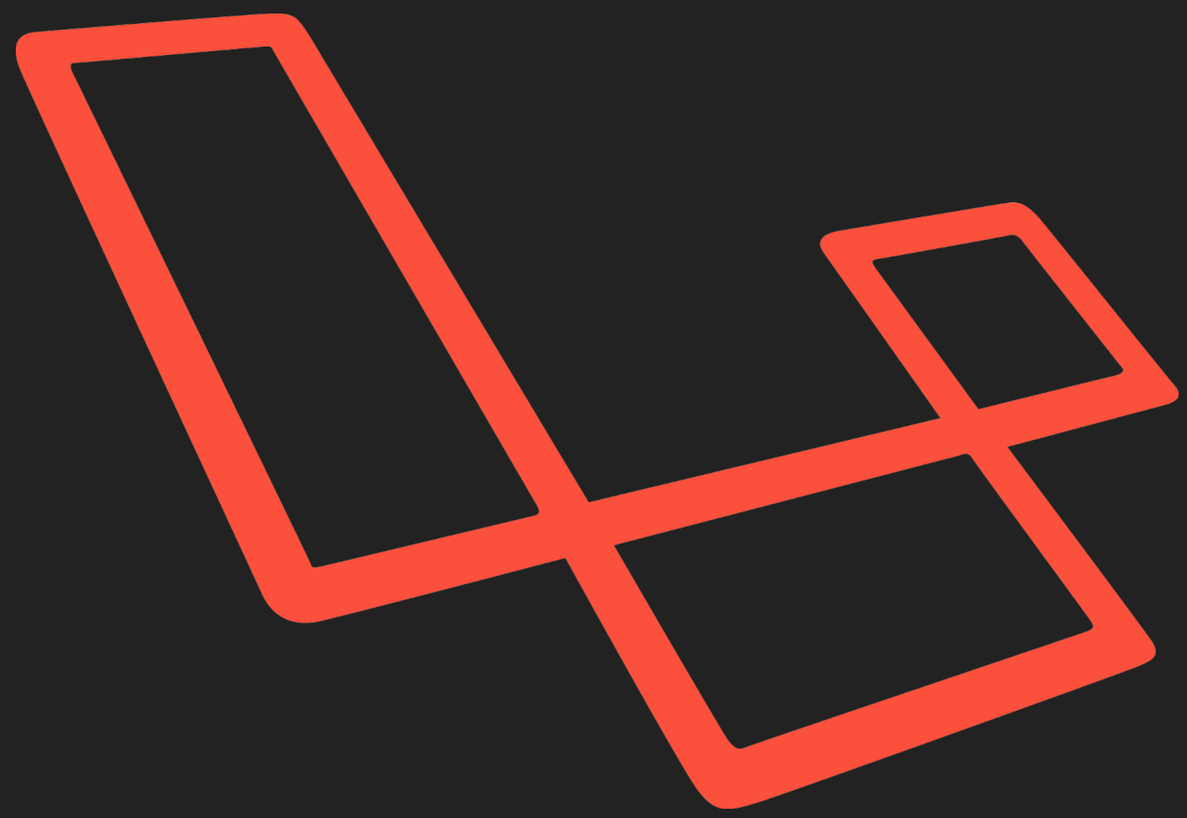
LES ROUTES :: REDIRECTION

- Vous avez la possibilité sous Laravel de faire également des redirections.
- On utilise le mot clef **redirect()**
- **TAF :** Créer un chemin /ancien qui redirige vers /nouveau celui-ci affichera simplement « Nouvelle page ! »
- **TAF** : faire une chemin /recherche qui redirige vers http://www.google.fr

LES ROUTES :: LES GROUPES DE ROUTES

- On peut vite avoir des dizaines de routes et certaines partagent des propriétés communes.
- Il peut être intéressant de les regrouper afin d'éviter des répétition et surtout pour plus de flexibilité.
- Exemple :

```php
Route::prefix('admin')->group(function () {
    Route::get('/users', function () {
    return "Groupe admin page user !";
});
```



DEBOGUAGE

OUTILS POUR DEBOGUER

- Je vais vous montrer quelques outils pour débogguer.
- Nous allons le faire dans le fichier contenant les routes (web.php) puisque c'est celui que nous avons vu, mais il est évident que ces outils s'appliquent partout dans Laravel.
- Commande **dd**
  A ne pas utiliser en production !
- dd stop le programme

```php
Route::get('/secret', function () {
    $date = "Accès à la page";
    dd ($date);
    return "Page secrète !";
});
```

OUTILS POUR DEBOGUER

- **TAF** : tester le code précedent avec dd.
- **TAF** : il existe aussi l'instruction dump. Testez là ! Quelle(s) différence(s) y a t'il ?
- Si on ne veut pas afficher à l'écran on utilisera les Log.
- Ces informations sont stockées dans storage/logs
- Il y a différents niveaux de sévérité : info, erreur, début, etc
- **TAF** : Modifier pour utiliser Log avec le niveau notice (Log::notice) allez voir si la ligne est bien dans le fichier logs