<https://symfony.com/doc/master/setup.html>

Instalarea și configurarea cadrului Symfony

Cerințe tehnice [¶](https://symfony.com/doc/master/setup.html#technical-requirements)

Înainte de a crea prima aplicație Symfony, trebuie să:

* Instalați PHP 7.2.9 sau o versiune ulterioară și aceste extensii PHP (care sunt instalate și activate în mod implicit în majoritatea instalațiilor PHP 7): [Ctype](https://php.net/book.ctype) , [iconv](https://php.net/book.iconv) , [JSON](https://php.net/book.json) , [PCRE](https://php.net/book.pcre) , [Session](https://php.net/book.session) , [SimpleXML](https://php.net/book.simplexml) și [Tokenizer](https://php.net/book.tokenizer) ;
* [Install Composer](https://getcomposer.org/download/) , care este utilizat pentru instalarea pachetelor PHP;
* [Instalați Symfony](https://symfony.com/download) , care creează în computer un apel numit binar symfony care oferă toate instrumentele de care aveți nevoie pentru a vă dezvolta aplicația local.

Crearea aplicațiilor Symfony

Deschideți terminalul consolei și rulați oricare dintre aceste comenzi pentru a crea o nouă aplicație Symfony:

|  |
| --- |
| symfony new my\_project\_name --full  symfony new my\_project\_name |

Singura diferență între aceste două comenzi este numărul de pachete instalate implicit. --fullOpțiunea instalează toate pachetele de care aveți nevoie , de obicei , pentru a construi aplicații web, astfel încât dimensiunea de instalare va fi mai mare.

Dacă nu puteți sau nu doriți să [instalați Symfony](https://symfony.com/download) din orice motiv, rulați aceste comenzi pentru a crea noua aplicație Symfony folosind Composer:

|  |
| --- |
| composer create-project symfony/website-skeleton my\_project\_name  composer create-project symfony/skeleton my\_project\_name |

Indiferent de comanda pe care o executați pentru a crea aplicația Symfony. Toate acestea vor crea un nou my\_project\_name/director, vor descărca unele dependențe în el și chiar vor genera directoarele de bază și fișierele de care va trebui să începeți. Cu alte cuvinte, noua dvs. aplicație este gata!

**NOTĂ**

Directorul cache și jurnalele proiectului (implicit <project>/var/cache/ și <project>/var/log/) trebuie să fie redactate de serverul web. Dacă aveți probleme, citiți cum să *[configurați permisiunile pentru aplicațiile Symfony](https://symfony.com/doc/master/setup/file_permissions.html)* .

Executarea aplicațiilor Symfony [¶](https://symfony.com/doc/master/setup.html#running-symfony-applications)

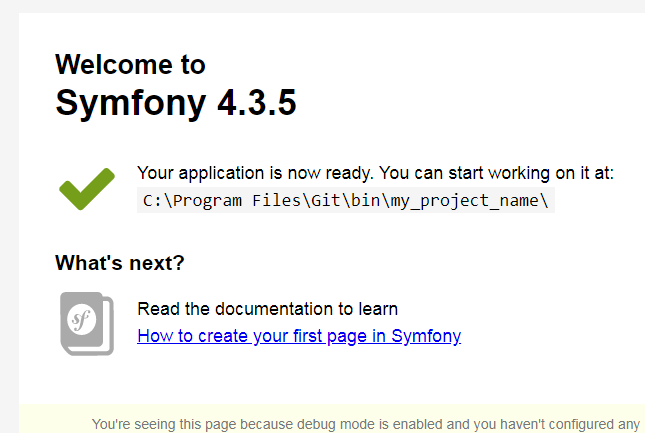
La producție, ar trebui să utilizați un server web precum Nginx sau Apache (consultați *[configurarea unui server web pentru a rula Symfony](https://symfony.com/doc/master/setup/web_server_configuration.html)* ). Dar pentru dezvoltare, este mai convenabil să folosiți *[serverul web local](https://symfony.com/doc/master/setup/symfony_server.html)* furnizat de Symfony.

Acest server local oferă asistență pentru HTTP / 2, TLS / SSL, generarea automată de certificate de securitate și multe alte funcții. Funcționează cu orice aplicație PHP, nu numai proiectele Symfony, deci este un instrument de dezvoltare foarte util.

Deschideți terminalul consolei, treceți în noul dvs. director de proiect și porniți serverul web local astfel:

|  |
| --- |
| cd my-project/  symfony server:start |

Deschideți browserul și navigați la http://localhost:8000/. Dacă totul funcționează, veți vedea o pagină de bun venit. Mai târziu, când ați terminat de lucrat, opriți serverul apăsând Ctrl+Cpe terminal.



Configurarea unui proiect Symfony existent [¶](https://symfony.com/doc/master/setup.html#setting-up-an-existing-symfony-project)

Pe lângă crearea de noi proiecte Symfony, veți lucra și la proiecte create deja de alți dezvoltatori. În acest caz, trebuie doar să obțineți codul proiectului și să instalați dependențele cu Composer.

<https://symfony.com/doc/master/page_creation.html>

Creați-vă prima pagină în Symfony

* + [Crearea unei pagini: Rută și controler](https://symfony.com/doc/master/page_creation.html#creating-a-page-route-and-controller)
  + [Rute de adnotare](https://symfony.com/doc/master/page_creation.html#annotation-routes)

Crearea unei noi pagini - indiferent dacă este o pagină HTML sau un punct final JSON - este un proces în doi pași:

1. **Creați un traseu** : o rută este adresa URL (de exemplu /about) către pagina dvs. și indică un controler;
2. **Creați un controler** : Un controler este funcția PHP pe care o scrieți care creează pagina. Luați informațiile de solicitare primite și le utilizați pentru a crea un Responseobiect Symfony , care poate reține conținut HTML, un șir JSON sau chiar un fișier binar precum o imagine sau un PDF.

Crearea unei pagini: Rută și controler [¶](https://symfony.com/doc/master/page_creation.html#creating-a-page-route-and-controller)

**BACSIS**

Înainte de a continua, asigurați-vă că ați citit articolul de *[configurare](https://symfony.com/doc/master/setup.html)* și puteți accesa noua aplicație Symfony în browser.

Să presupunem că doriți să creați o pagină - /lucky/number- care generează un număr norocos (bine, la întâmplare) și o tipărește. Pentru aceasta, creați o clasă „Controller” și o metodă „controller” în interiorul acesteia:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?php  *// src/Controller/LuckyController.php*  namespace App\Controller;  use Symfony\Component\HttpFoundation\Response;  class LuckyController  {  public function number()  {  $number = random\_int(0, 100);  return new Response(  '<html><body>Lucky number: '.$number.'</body></html>'  );  }  } |

Acum trebuie să asociați această funcție de controler cu o adresă URL publică (de exemplu /lucky/number), astfel încât number()metoda să fie executată atunci când un utilizator răsfoiește la ea. Această asociere este definită prin crearea unei **rute** în config/routes.yaml fișier:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | *# config/routes.yaml*  *# the "app\_lucky\_number" route name is not important yet*  app\_lucky\_number:  path: /lucky/number  controller: App\Controller\LuckyController::number |

*Компонент Yaml загружает и сбрасывает файлы YAML.*

Компонент Symfony Yaml анализирует YAML строки, чтобы преобразовать их в PHP массивы. Он также может преобразовывать PHP массивы в YAML строки.

[YAML](http://yaml.org/), YAML не является языком разметки - это дружелюбный стандарт сериализации данных для всех языков программирования. YAML является отличным форматом для ваших файлов конфигурации. Файлы YAML так же экспрессивны, как файлы XML, и так же читаемы, как файлы INI.

Asta e! Dacă utilizați serverul web Symfony, încercați-l accesând:

[*http: // localhost: 8000 / lucky/number*](http://localhost:8000/lucky/number)

Dacă vedeți că un număr norocos vă este imprimat înapoi, felicitări! Dar înainte să fugi să joci la loterie, verifică cum funcționează asta. Vă amintiți cei doi pași pentru crearea unei pagini?

1. *Creați un traseu* : În config/routes.yaml, ruta definește adresa URL a dvs.

pagina ( path) și ce controllersă apeleze. Veți afla mai multe despre [rutarea](https://symfony.com/doc/master/routing.html) în propria secțiune, inclusiv despre cum să faceți adrese URL *variabile* ;

1. *Creați un controler* : Aceasta este o funcție în *care* construiți pagina și în cele din urmă returnați un Responseobiect. Veți afla mai multe despre [controlerele](https://symfony.com/doc/master/controller.html) din propria lor secțiune, inclusiv despre cum să returnați răspunsurile JSON.

Rute de adnotare [¶](https://symfony.com/doc/master/page_creation.html#annotation-routes)

În loc să definiți ruta dvs. în YAML, Symfony vă permite, de asemenea, să utilizați rutele de *adnotare* . Pentru a face acest lucru, instalați pachetul de adnotări:

|  |
| --- |
| composer require annotations |

Acum puteți adăuga ruta direct *deasupra* controlerului:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | // src/Controller/LuckyController.php  // ...  + use Symfony\Component\Routing\Annotation\Route;  class LuckyController  {  + /\*\*  + \* @Route("/lucky/number")  + \*/  public function number()  {  // this looks exactly the same  }  } |

Asta e! Pagina - http://localhost:8000/lucky/numberva funcționa exact ca înainte! Adnotările sunt modalitatea recomandată de configurare a rutelor.

Instalarea automată a rețetelor cu Symfony Flex [¶](https://symfony.com/doc/master/page_creation.html#auto-installing-recipes-with-symfony-flex)

Poate că nu ați observat, dar când ați alergat composer require annotations, s-au întâmplat două lucruri speciale, ambele datorită unui puternic plugin Composer numit [Flex](https://symfony.com/doc/master/setup.html#symfony-flex) .

În primul rând, annotationsnu este un nume adevărat de pachet: este un *alias* (adică scurtătură) la care se rezolvă Flex sensio/framework-extra-bundle.

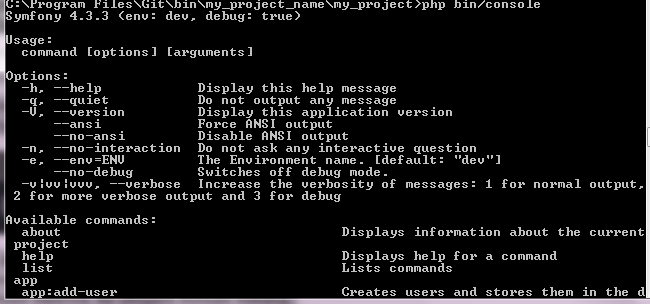
În al doilea rând, după ce acest pachet a fost descărcat, Flex a executat o *rețetă* , care este un set de instrucțiuni automatizate care spun Symfony cum să integreze un pachet extern. [Rețetele Flex](https://flex.symfony.com/) există pentru multe pachete și au capacitatea de a face multe, cum ar fi adăugarea fișierelor de configurare, crearea de directoare, actualizarea .gitignore și adăugarea de configurații noi în .envfișierul dvs. Flex *automatizează* instalarea pachetelor, astfel încât să puteți reveni la codare.

comandă bin / console  [¶](https://symfony.com/doc/master/page_creation.html#the-bin-console-command)

Proiectul dvs. are deja un instrument puternic de depanare în interior: bin/consolecomanda. Încercați să-l executați:

|  |
| --- |
| php bin/console |

Ar trebui să vedeți o listă de comenzi care vă pot oferi informații de depanare, ajutați la generarea de coduri, la generarea migrațiilor de bază de date și multe altele. Pe măsură ce instalați mai multe pachete, veți vedea mai multe comenzi.



Pentru a obține o listă cu *toate* rutele din sistemul dvs., utilizați debug:routercomanda:

|  |
| --- |
| php bin/console debug:router |

Ar trebui să vedeți app\_lucky\_numbertraseul dvs. chiar de sus:

| **Nume** | **Metodă** | **Sistem** | **Gazdă** | **cale** |
| --- | --- | --- | --- | --- |
| app\_lucky\_number | ORICE | ORICE | ORICE | /numar norocos |

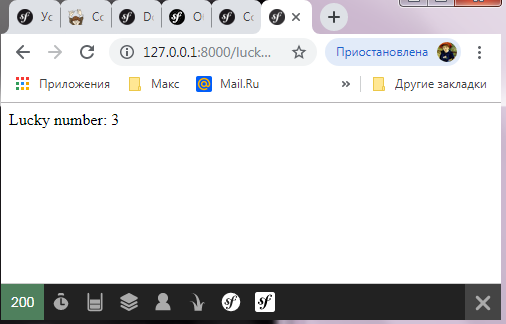
De asemenea, veți vedea mai jos rutele de depanare app\_lucky\_number- mai multe pe rutele de depanare din secțiunea următoare.

Veți afla despre multe alte comenzi în timp ce continuați!

Bara de instrumente pentru depanare Web: Debugging Dream [¶](https://symfony.com/doc/master/page_creation.html#the-web-debug-toolbar-debugging-dream)

Una dintre caracteristicile *ucigașe* ale Symfony este bara de instrumente de depanare Web: o bară care afișează o cantitate *uriașă* de informații de depanare de-a lungul părții de jos a paginii dvs. în timpul dezvoltării. Toate acestea sunt incluse în cutie folosind un [pachet](https://symfony.com/doc/master/setup.html" \l "symfony-packs) numit [Symfony](https://symfony.com/doc/master/setup.html" \l "symfony-packs)symfony/profiler-pack .

Veți vedea o bară neagră în partea de jos a paginii. Veți afla mai multe despre toate informațiile pe care le deține pe parcurs, dar nu ezitați să experimentați: treceți peste și faceți clic pe pictogramele diferite pentru a obține informații despre rutare, performanță, înregistrare și multe altele.



Redarea unui șablon [¶](https://symfony.com/doc/master/page_creation.html#rendering-a-template)

Dacă returnați HTML de pe controler, probabil veți dori să redau un șablon. Din fericire, Symfony vine cu [Twig](https://twig.symfony.com/) : un limbaj de șablon ușor, puternic și de fapt destul de distractiv.

**Почему еще один шаблонизатор?**

Когда речь заходит о механизмах шаблонов в PHP, многие люди скажут вам, что сам PHP является механизмом шаблонов. Но даже если PHP начал свою жизнь как язык шаблонов, он не развивался так, как в последние годы. На самом деле, **он не поддерживает многие функции**, которые должны иметь современные движки шаблонов:

* **Кратко** : язык PHP многословен и становится смешно многословным, когда дело доходит до экранирования:
* <? php echo $ var ?>
* <? php echo htmlspecialchars ( $ var , ENT\_QUOTES , 'UTF-8' ) ?>

Для сравнения, Twig имеет очень лаконичный синтаксис, который делает шаблоны более читабельными:

{{ var }}

{{ var|escape }}

{{ var|e }} *{# shortcut to escape a variable #}*

* **Очистка сообщений об ошибках** : всякий раз, когда у вас возникает проблема с синтаксисом в шаблоне, Twig выводит полезное сообщение с именем файла и номером строки, в которой возникла проблема. Это значительно облегчает отладку.
* **Быстро** : одна из целей Twig - быть как можно быстрее. Для достижения максимальной скорости Twig компилирует шаблоны в простой оптимизированный PHP-код. Издержки по сравнению с обычным кодом PHP были сведены к минимуму.

<https://twig.symfony.com/>

Asigurați-vă că LuckyControllerextinde [AbstractController](https://github.com/symfony/symfony/blob/master/src/Symfony/Bundle/FrameworkBundle/Controller/AbstractController.php" \o "Symfony\\Bundle\\FrameworkBundle\\Controller\\AbstractController)clasa de bază a Symfony :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | // src/Controller/LuckyController.php  // ...  + use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  - class LuckyController  + class LuckyController extends AbstractController  {  // ...  } |

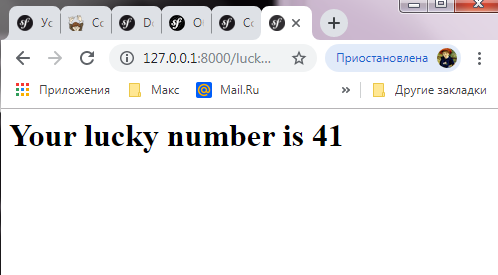
Acum, utilizați render()funcția la îndemână pentru a reda un șablon. Treceți-o pe o number variabilă, astfel încât să o puteți utiliza în Twig:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | *// src/Controller/LuckyController.php*  namespace App\Controller;  *// ...*  class LuckyController extends AbstractController  {  */\*\**  *\* @Route("/lucky/number")*  *\*/*  public function number()  {  $number = random\_int(0, 100);  return $this->render('lucky/number.html.twig', [  'number' => $number,  ]);  }  } |

Fișierele de șabloane se află în templates/director, care a fost creat automat pentru dvs. când ați instalat Twig. Creați un nou templates/luckydirector cu un nou number.html.twigfișier în interior:

|  |  |
| --- | --- |
| 1  2 | *{# templates/lucky/number.html.twig #}*  <h1>Your lucky number is {{ number }}</h1> |

{{ number }}Sintaxa este utilizată pentru a *imprima* variabile în crenguță. Reîmprospătați browserul pentru a obține *noul* dvs. număr norocos!

[*http: // localhost: 8000 / norocos / numărul*](http://localhost:8000/lucky/number)

Acum s-ar putea să vă întrebați unde s-a dus Bara de instrumente de depanare Web: asta pentru că nu există nicio </body>etichetă în șablonul curent. Puteți adăuga singur elementul de corp sau extinde base.html.twig, care conține toate elementele HTML implicite.

În articolul cu *[șabloane](https://symfony.com/doc/master/templates.html)* , veți afla totul despre Twig: cum să faceți buclă, să redați alte șabloane și să folosiți sistemul său puternic de moștenire a machetei.

Verificarea structurii proiectului [¶](https://symfony.com/doc/master/page_creation.html#checking-out-the-project-structure)

Vesti bune! Ați lucrat deja în cele mai importante directoare din proiectul dvs.:

config/

Conține ... configurare !. Veți configura rute, *[servicii](https://symfony.com/doc/master/service_container.html)* și pachete.

src/

Tot codul dvs. PHP locuiește aici.

templates/

Toate șabloanele tale de Twig trăiesc aici.

De cele mai multe ori, veți lucra în src/, templates/sau config/. Pe măsură ce continuați să citiți, veți afla ce se poate face în fiecare dintre acestea.

Ce zici de celelalte directoare din proiect?

bin/

Celebrul bin/consolefișier trăiește aici (și alte fișiere executabile mai puțin importante).

var/

Aici sunt stocate fișierele create automat, precum fișierele cache ( var/cache/) și jurnalele ( var/log/).

vendor/

Biblioteci terțe (adică „furnizor”) locuiesc aici! Acestea sunt descărcate prin intermediul managerului de pachete [Composer](https://getcomposer.org/) .

public/

Aceasta este rădăcina documentului pentru proiectul dvs.: puneți aici toate fișierele accesibile publicului.

Și când instalați pachete noi, directoarele noi vor fi create automat atunci când este nevoie.

Ce urmeaza? [¶](https://symfony.com/doc/master/page_creation.html#what-s-next)

Felicitări! Începi deja să stăpânești Symfony și să înveți un mod cu totul nou de a construi aplicații frumoase, funcționale, rapide și de întreținut.

Ok, este timpul să terminați să stăpâniți elementele fundamentale citind aceste articole:

* [Routing](https://symfony.com/doc/master/routing.html)
* [Controlor](https://symfony.com/doc/master/controller.html)
* [Crearea și utilizarea șabloanelor](https://symfony.com/doc/master/templates.html)
* [Configurarea Symfony](https://symfony.com/doc/master/configuration.html)

Apoi, aflați despre alte subiecte importante cum ar [*fi containerul de servicii*](https://symfony.com/doc/master/service_container.html) , sistemul de *[formulare](https://symfony.com/doc/master/forms.html)* , folosind [*Doctrine*](https://symfony.com/doc/master/doctrine.html) (dacă trebuie să interogați pe o bază de date) și multe altele!

## <https://symfony.com.ua/doc/current/quick_tour/the_big_picture.html>

# Imaginea [mare](https://symfony.com.ua/doc/current/quick_tour/the_big_picture.html#id1)

Începeți cu Symfony în 10 minute! Adevărat! Acesta este tot timpul necesar pentru a înțelege cele mai importante concepte și a începe să dezvolți un proiect real!

Dacă ați folosit deja cadrul web, atunci ar trebui să vă simțiți ca acasă folosind Symfony. Dacă nu, atunci bineveniți la noua modalitate de dezvoltare a aplicațiilor web. Symfony acoperă cele mai bune practici, are compatibilitate înapoi (Da! Actualizarea este întotdeauna ușoară și sigură!) Și oferă suport pe termen lung.

## Descărcați Symfony [¶](https://symfony.com.ua/doc/current/quick_tour/the_big_picture.html#symfony)

Pentru a începe, asigurați-vă că ați instalat [Composer](https://getcomposer.org/) și că aveți versiunea PHP 7.1.3 sau o versiune ulterioară.

Ești gata? În terminal, faceți:

|  |
| --- |
| composer create-project symfony/skeleton quick\_tour |

Se creează un nou director quick\_tour/cu o aplicație Symfony nouă, dar puternică:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | quick\_tour /  ├─ .env  ├─ .env.dist  ├─ bin / console  ├─ compozitor.json  ├─ compozitor.lock  ├─ configura /  ├─ public / index.php  ├─ src /  ├─ symfony.lock  ├─ var /  └─ vânzător / |

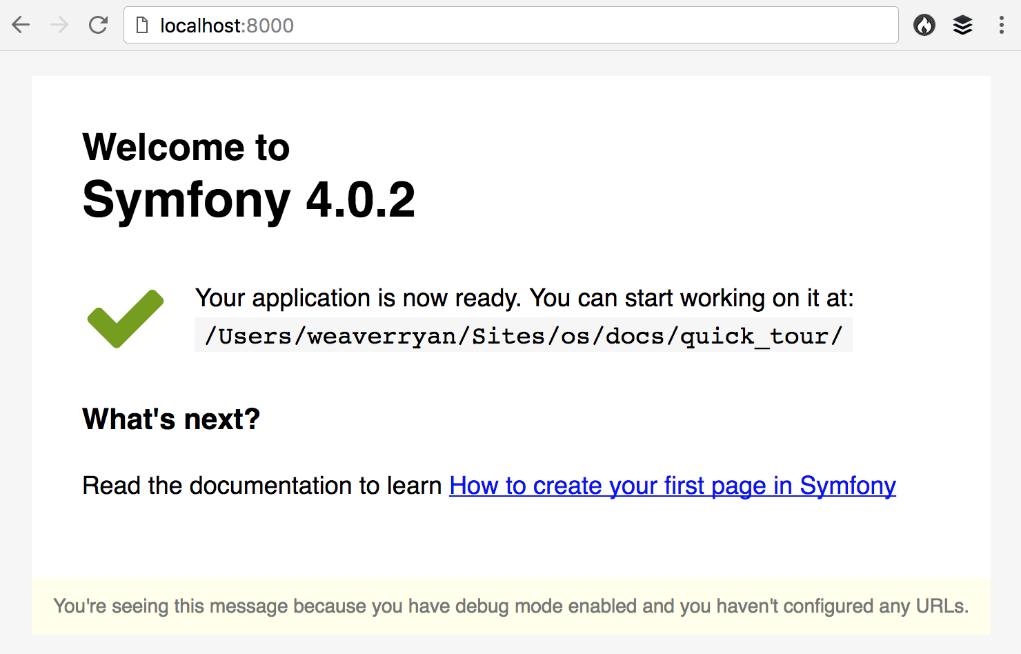
Putem deja încărca proiectul în browser? Desigur! Puteți instala [Nginx sau Apache](https://symfony.com.ua/doc/current/setup/web_server_configuration.html) și le puteți configura cu directorul documentului rădăcină public/. Dar pentru dezvoltare, Symfony are propriul său server. Instalați și rulați-l cu:

|  |
| --- |
| composer necesită server --dev  php server bin / console: start |

Încercați noua dvs. aplicație făcând clic pe linkul http://localhost:8000din browserul dvs.!

**Перейти в каталог quick\_tour и**

**Symfony server:start**



## Noțiuni de bază: Traseu, Controler, Răspuns [¶](https://symfony.com.ua/doc/current/quick_tour/the_big_picture.html#id2)

Există doar aproximativ 15 fișiere în proiectul nostru, dar a devenit deja o API elegantă, o aplicație web de lux sau microservice. Symfony începe de mici, dar crește cu tine.

Dar înainte de a merge prea departe, să săpăm în elementele de bază construind prima noastră pagină.

Începeți de la config/routes.yaml: aici putem specifica adresa URL a noii noastre pagini. Eliminați comentariile din exemplul care există deja în fișier:

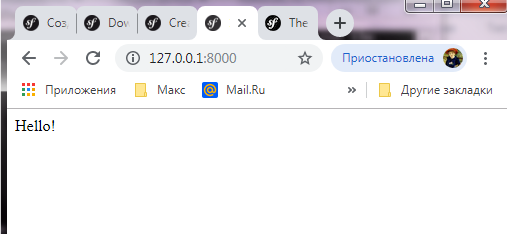
|  |  |
| --- | --- |
| 1  2  3 | index :  calea : /  controller : 'App \ Controller \ DefaultController :: index' |

Aceasta se numește un traseu : definește adresa URL a paginii dvs. ( /) și „controlerului”: o funcție care va fi apelată de fiecare dată când cineva merge la această adresă URL. Această funcție nu există încă, așa că haideți să o creăm!

În src/Controller, creați o clasă nouă DefaultControllerși în interiorul acesteia o metodă index:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | App pentru spații de nume \ Controller ;  folosiți Symfony \ Component \ HttpFoundation \ Răspuns ;  class DefaultController  {  index public function () { return new Response ( 'Hello!' ); } } |

Asta este totul! Încercați să mergeți la pagina de start: http://localhost:8000/. Symfony vede că adresa URL se potrivește cu traseul nostru și execută o nouă metodă index().



Un controler este o funcție normală cu o singură regulă: ar trebui să returneze un obiect Symfony Response. Dar acest răspuns poate conține orice: text simplu, JSON sau o pagină HTML completă.

Dar sistemul de rutare este mult mai puternic. Deci, să facem traseul mai interesant:

|  |  |
| --- | --- |
| 1  2  3  4  5 | # index  / routes.yaml  index: - calea: /  + calea: / hello / {name}  controller: 'App \ Controller \ DefaultController :: index' |

URL - ul pentru aceasta pagina sa schimbat: Acum este /hello/\*: {name}actioneaza ca un wildcard, de potrivire ceva. Totul se îmbunătățește! Actualizați și controlerul:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // src / Controller / DefaultController.php  // ...  - indexul funcției publice ()  + indexul funcției publice (nume $)  {  - returnați un nou răspuns („Bună ziua!”);  + returnați nou Răspuns („Bună ziua, nume $!”);  } |

Încercați această pagină accesând http://localhost:8000/hello/Symfony. Ar trebui să vezi: Bună ziua, Symfony! Valoarea {name}din URL este disponibilă ca argument $nameîn controler.

Dar ar putea fi și mai ușor! Așadar, să instalăm suportul pentru adnotare:

|  |
| --- |
| composer require annotations |

Acum, comentează ruta YAML adăugând un caracter #:

|  |  |
| --- | --- |
| 1  2  3  4 | # config / routes.yaml  # index:  # path: / Hi / {name}  # controller: 'App \ Controller \ DefaultController :: index' |

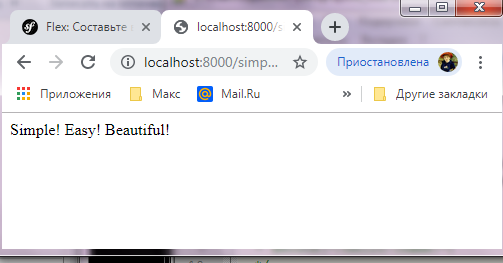
În schimb, adăugați o rută direct deasupra metodei controlerului:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // src / Controller / DefaultController.php  // ...  + utilizați Symfony \ Component \ Routing \ Annotation \ Route;  + / \*\*  + \* @Route ("/ Hello / {name}")  + \* /  indicele funcției publice (nume $) |

Funcționează la fel ca înainte! Dar folosind adnotări, ruta și controlorul trăiesc chiar unul lângă altul. Ai nevoie de altă pagină? Adaugă doar o altă rută și o metodă la DefaultController:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | // src / Controller / DefaultController.php  // ...  / \*\*  \* @Route ("/ simplitate")  \* /  public function simple ()  {  return new Response ( 'Simple! Easy! Beautiful!' );  } |

Traseul poate face și mai mult, dar îl vom lăsa data viitoare. Acum, aplicația noastră are nevoie de mai multe funcții! Ca un motor de șabloane, logare, instrumente de depanare și multe altele.



Continuați să citiți [Flex: compuneți-vă aplicația](https://symfony.com.ua/doc/current/quick_tour/flex_recipes.html) .