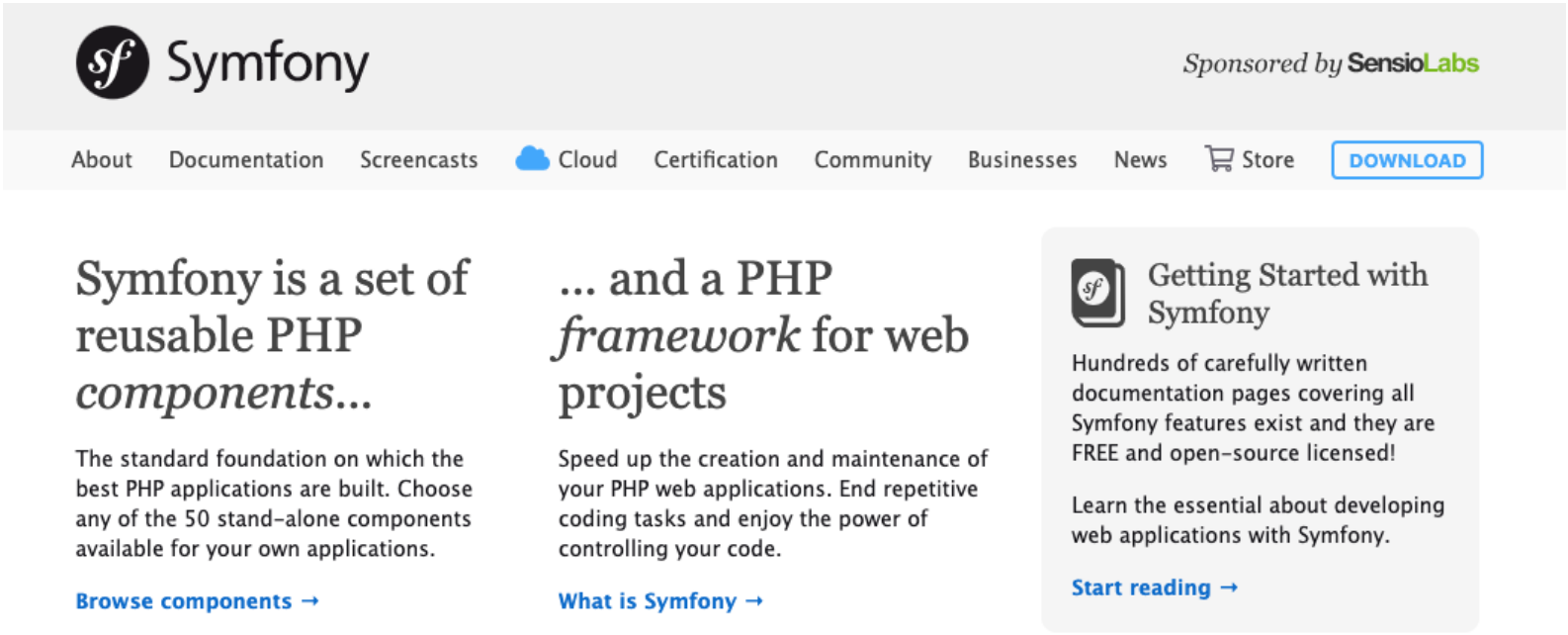


https://symfony.com/

Install Symfony

- Login to the DAW2 server (IP 10.10.23.184) on the DEEI network using PuTTY and change to the public_html folder

```
a12345@daw2:~/public_html/$
composer create-project symfony/website-skeleton:"^4.4" LAB9_10
```

This command will create a directory named `LAB9_10` containing a fresh Symfony installation with many Symfony components already installed

Configure Symfony

- **Setup the database in** hidden file `.env`.

```
a12345@daw2:~/public_html/LAB9_10$ nano .env
```

```
DATABASE_URL=mysql://a999999:password@127.0.0.1:3306/db_a999999?serverVersion=15.1
```

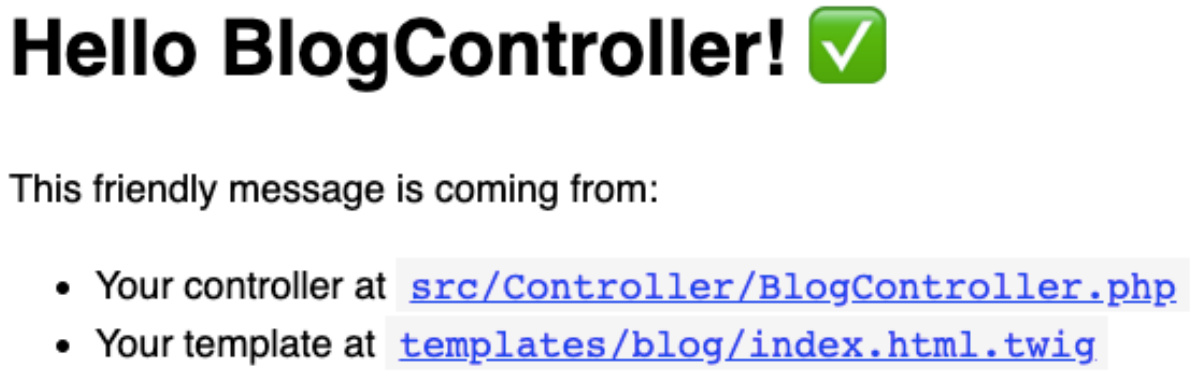
Make your first controller

```
a12345@daw2:~/public_html$ cd LAB9_10

a12345@daw2:~/public_html/LAB9_10$ php bin/console make:controller

Choose a name for your controller class (e.g. GrumpyChefController):
 > BlogController
```

Welcome to Symfony !

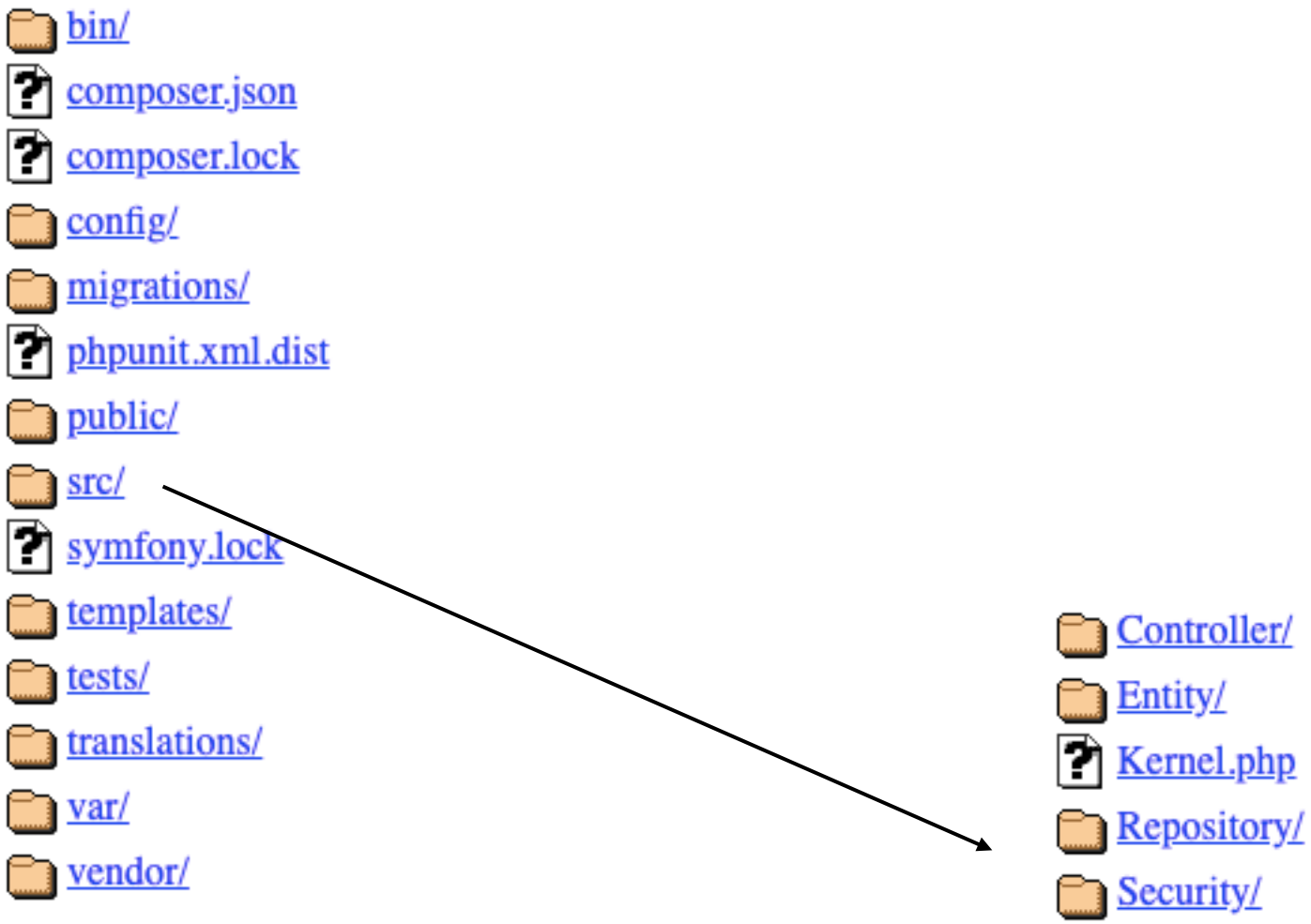
- Point your browser to

  http://daw.deei.fct.ualg.pt/~a12345/LAB9_10/public/index.php/blog

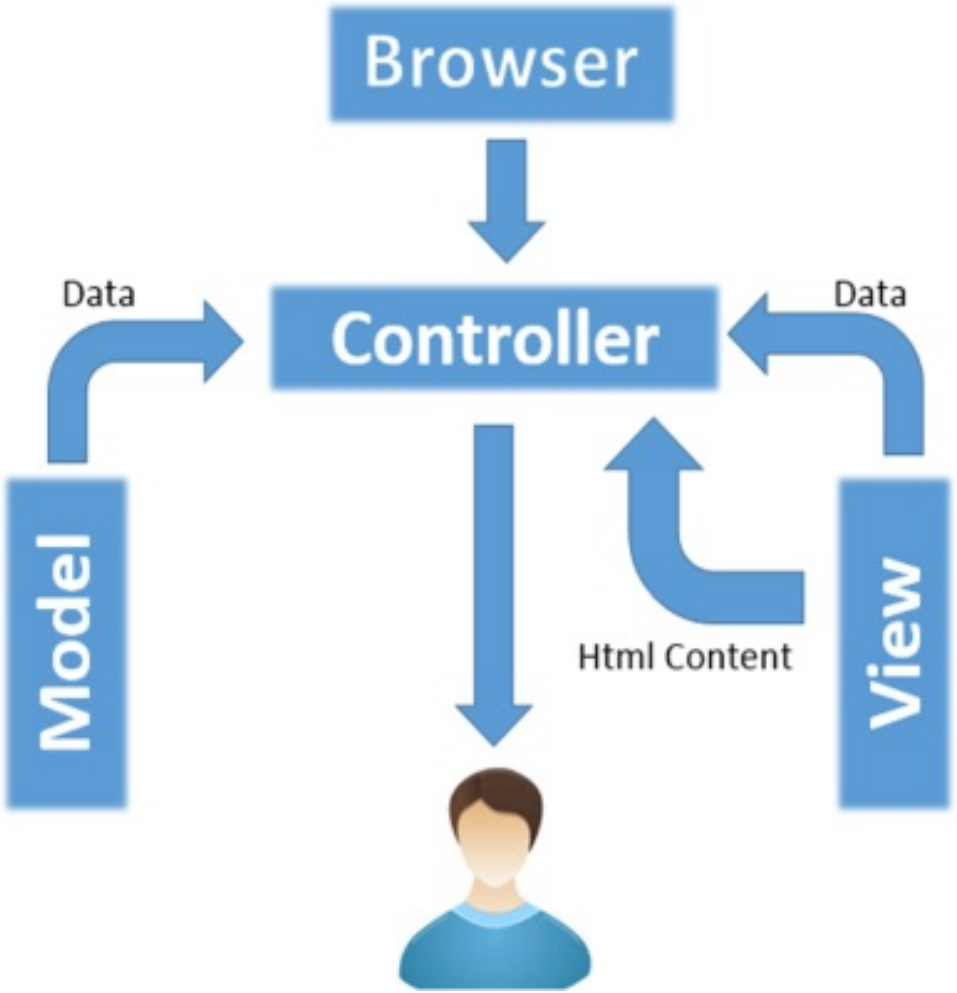


If you see the web page above you have successfully installed Symfony!

Directory Structure



MVC Paradigm



MVC Paradigm …

- Controllers – This folder holds the controllers of the application.
- Entity – The database model is placed in this folder.
- templates – Application's template files are placed in this folder.

Extra:

- public - place static resources (images, css, js) in this folder

The controller class

```
<?php

namespace App\Controller;

use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use App\Controller\Stud_model;


class Stud extends AbstractController
{
    private $stud_model;
    public function __construct(Stud_model $stud_model)
     {
         $this->stud_model = $stud_model;
     }


    /**
      * @Route("/stud", name="stud.view")
      */
    public function index()
    {
        $data['records'] = $this->stud_model->get_all();
        return $this->render('stud_view.html.twig', $data);
     }
```

The controller class …

- The general syntax for calling the controller is

  http://www. domain.com/index.php/class/method/arg

- If rewriting base is active in .htaccess

  http://www. domain.com/class/method/arg

URL Rewriting

The segments in a URL normally follow this pattern

```
www.domain.com/class/method/arg1/arg2/…
```

- The **first segment** represents the controller class
- The **second segment** represents the class method (function)
- The **third segment**, and any additional segments, represent any variables that will be passed to the controller.

Symfony Routes

```
/**
  * @Route("/stud", name="stud.view")
  */
 public function index()


/**
  * @Route("/stud/delete/{roll_no?}", name="delete")
  */
 public function delete($roll_no = NULL)


 /**
   * @Route("/stud/add", name="stud.add")
   */
 public function add()
```

URL Rewriting …

- **URL** `/~a12345/LAB9_10/public/index.php/stud`

gets method index() in file Stud.php

`/users/a12345/public_html/LAB9_10/src/Controller/Stud.php`

- **URL** `/~a12345/LAB9_10/public/index.php/stud/delete/89067`

gets method "delete($roll_no = NULL)" in file Stud.php

… and passes value "89067" to the delete() method

The model class

```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Doctrine\DBAL\Driver\Connection;


class Stud_model extends AbstractController
{
   private $connection;
   public function __construct(Connection $connection)
     {
          $this->connection = $connection;
     }

    public function get_all()
    {
      $sql =  "SELECT * FROM stud";
      $students = $this->connection->fetchAll($sql);
      return $students;
    }
```

The Twig template

```
<html lang = "en">

   <head>
     <meta charset = "utf-8">
     <title>Students Example</title>
   </head>

   <body>
   <img src="{{ asset('images/student.png') }}" width="208"
height="124"/>
   <br />

     <a href = "{{ path('stud.add') }}">Add</a>
         <table border = "1">
           <tr>
             <td>Sr#</td>
             <td>Roll No.</td>
             <td>Name</td>
             <td>Edit</td>
             <td>Delete</td>
           <tr>
```

```
            {% for r in records %}
                <tr>
                <td> {{loop.index}} </td>
                <td> {{r.roll_no}} </td>
                <td> {{r.Name}} </td>
                <td><a href = "{{ path('stud.update', { 'roll_no':
r.roll_no }) }}">Edit</a></td>
                                    <td><a href = "{{ path('stud.delete', {
'roll_no': r.roll_no }) }}">Delete</a></td>
                <tr>
            {% endfor %}
    </table>
  </body>
</html>
```

Twig URL Functions

```
/**
   * @Route("/stud/update/{roll_no?}", name="stud.update")
   */
   public function update($roll_no = NULL)
```

path

```
path('stud.update', { 'roll_no': 23 })
```

```
http://daw.deei.fct.ualg.pt/~a12345/LAB9_10/public/index.php/stud/update/23
```

asset

```
asset('images/student.png');
```

```
http://daw.deei.fct.ualg.pt/~a12345/LAB9_10/public/images/student.png
```

Form validation (Twig template)

```
<html lang = "en">
   <head>
     <meta charset = "utf-8">
     <title>Students Example</title>
   </head>
   <body>
         {% if errors > 0 %}
            <h2>Validation failed</h2>
            <ul>
            {% for errorMessage in errorMessages %}
               <li> {{errorMessage}} </li>
            {% endfor %}
            </ul>
         {% endif %}

     <form method="post" action="{{path('stud.add_action')}}">
         <input type="hidden" name="token" value="{{ csrf_token('myform') }}" />
         <label>Roll No.</label>
         <input type="text" name="roll_no" value=" {{ oldroll_no }} "/><br/>
         <label>Name</label>
         <input type="text" name="name" value=" {{ oldname }} "/><br/>
         <input type="submit" value="Add" />
     </form>
   </body>
</html>
```

Form validation ..

```
public function add_action(Request $request, ValidatorInterface $validator)
     {
          $roll_no=$request->request->get('roll_no');
          $name=$request->get('name');

          // test if student is already in database
          $student = $this->stud_model->get($roll_no);
          $value = ( $student == false ? '' : $student['roll_no'] );

          $input = ['roll_no' => $roll_no, 'name' => $name];
          $constraints = new Assert\Collection([
             'roll_no' => [
                new Assert\NotBlank,
                new Assert\Type('numeric'),
                new Assert\NotEqualTo(['value' => $value, 'message' => "This value
is already in the database"])],

             'name' => [new Assert\notBlank],
          ]);
```

Form validation ...

```
$violations = $validator->validate($input, $constraints);

if (count($violations) > 0) {

$accessor = PropertyAccess::createPropertyAccessor();

$errorMessages = [];

foreach ($violations as $violation) {

   $accessor->setValue($errorMessages,
      $violation->getPropertyPath(),
      $violation->getMessage());
}

$data['errors'] = count($violations);
$data['errorMessages'] = $errorMessages;
$data['oldroll_no'] = $roll_no;
$data['oldname'] = $name;
return $this->render('stud_add.html.twig', $data);
}
```

Cross Site Request Forgery Protection

- **Cross-site request forgery CSRF** is a type of malicious exploit of a website where unauthorized commands are submitted from a user that the web application trusts. In a CSRF attack, an innocent end user is tricked by an attacker into submitting a web request that they did not intend.

```
public function add_action(Request $request, ValidatorInterface $validator)
     {


$token = $request->request->get("token");

          if (!$this->isCsrfTokenValid('myform', $token)) {
              return new Response("Operation not allowed", Response::HTTP_OK,
                  ['content-type' => 'text/plain']);
          }
```

Sessions

```
use Symfony\Component\HttpFoundation\Session\SessionInterface;

  private $session;
  public function __construct(SessionInterface $session)
{
          $this->session = $session;
}
```

- Retrieving session data

```
$user_id = $this->session->get('userid');
```

- Assigning session data

```
$this->session->set('userid', $user['id']);
```

- Unset session data

```
$this->session->set('userid', '');
$this->session->remove('userid');
```

- Destroy session

```
$this->session->clear();
```

Cookies 1

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

// Assigning cookie data

```
$cookie = Cookie::create('REMEMBERME')
    ->withValue('7635cc0b149328a901')
     ->withExpires(time() + (60 * 24 * 2)) // 2 days
    ->withDomain('daw.deei.fct.ualg.pt');

$response = new Response();
$response->setStatusCode(Response::HTTP_MOVED_PERMANENTLY);
$response->headers->set('Location', $this->generateUrl('message'));
$response->headers->setCookie($cookie);

$response->send();
```

Cookies 2

// Assigning cookie data (more simple version)

```
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\HttpFoundation\Cookie;

$response = new RedirectResponse($this->generateUrl('message'));
$cookie = Cookie::create(' REMEMBERME ', '7635cc0b149328a901', new
\DateTime('+2 days'))
$response->headers->setCookie($cookie);
$response->send();
```

// Assigning cookie data (even more simple version)

```
$response = new Response();
$cookie = Cookie::create(' REMEMBERME ', '7635cc0b149328a901', new
\DateTime('+2 days'))
$response->headers->setCookie($cookie);
$response->send();
```

Cookies 3

```
// Retrieving cookie data

    $remember_digest = $request->cookies->get('REMEMBERME');
```

Email

```
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Mailer\MailerInterface;
use Symfony\Component\Mime\Email;

class MailerController extends AbstractController
{
    /**
     * @Route("/email")
     */
    public function sendEmail(MailerInterface $mailer)
    {
        $email = (new Email())
            ->from('hello@example.com')
            ->to('you@example.com')
            //->cc('cc@example.com')
            //->bcc('bcc@example.com')
            //->replyTo('fabien@example.com')
            //->priority(Email::PRIORITY_HIGH)
            ->subject('Time for Symfony Mailer!')
            ->text('Sending emails is fun again!')
            //->html('<p>See Twig integration for better HTML integration!</p>');
            //->text(fopen('/path/to/emails/user_signup.txt', 'r'))
            //->html(fopen('/path/to/emails/user_signup.html', 'r'))

        $mailer->send($email);
```

Email with Twig template 1

CONTROLADOR

```
use Symfony\Bridge\Twig\Mime\TemplatedEmail;
use Symfony\Component\Mime\Address;

$email = (new TemplatedEmail())
    ->from('webmaster@deei.fct.ualg.pt')
    ->to(new Address('a12345@ualg.pt'))
    ->subject('Nova password')

    // path of the Twig template to render
    ->htmlTemplate('email_template.html.twig')

    // pass variables (name => value) to the template
    ->context([
        'expiration_date' => new \DateTime('+1 days'),
        'username' => 'Pedro Silva',
        'reset_digest' => '2e26a470fbe31f3b0e3da2b21eeb6d31',
    ])
;
```

Email with Twig template 2

TEMPLATE **'email_template.html.twig**

```
<html>
<head>
<body>
<pre>
Caro {{ username }}
Para obter uma nova password clique no link

http://daw.deei.fct.ualg.pt/~a12345/LAB8_10/blog/new_password/{{reset_digest}}
Este link é valido até {{ expiration_date | date('d-m-Y') }}

                    Se NÃO pediu uma nova password IGNORE este email.


                    Cumprimentos,

                    webmaster!

                    Página Web:     http://intranet.deei.fct.ualg.pt/DAW/
                    E-mail:         webmaster@deei.fct.ualg.pt

                    NOTA: Não responda a este email, não vai obter resposta!
</pre>
</body>
</html>
```