Laborator 11 - Web Frameworks

Obiective: În urma parcurgerii acestui laborator studentul va:

- înţelege conceptele din spatele Symfony
- evalua beneficiile folosirii frameworkului
- trece prin procesul de instalare al Symfony
- crea un modul în cadrul unei aplicaţii

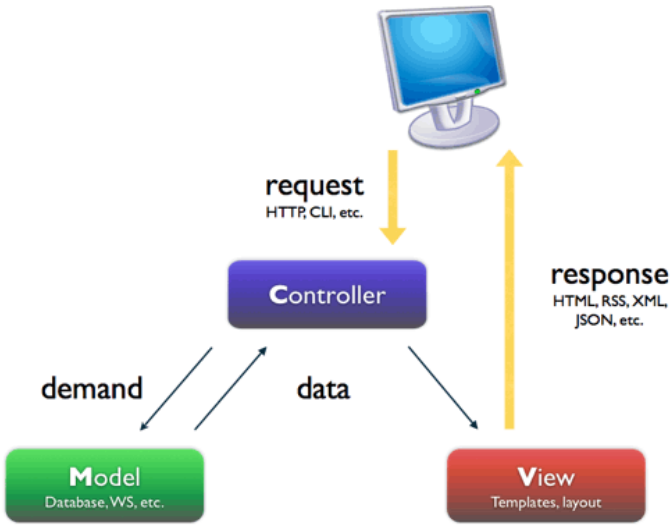
Symfony

Symfony este unul din cele mai populare frameworkuri PHP. Acest laborator este o introducere în Symfony şi urmăreşte să vă familiarizeze cu un proces simplificat de instalare şi cu operaţiile de bază care se pot efectua cu framework-ul.

Un framework este similar cu o bibliotecă şi/sau o aplicaţie, însă are în plus următoarele caracteristici determinante:

- **inversion of control**: fluxul datelor în program nu mai este dictat de apelant ci de framework
- **default behavior**: chiar şi atunci când nu este implementat nimic, framework-ul oferă o comportare implicită *funcţională*
- **extensibility**: framework-urile pot fi extinse prin suprascrierea metodelor din obiectele prezente
- **codul din framework nu trebuie modificat**. În schimb, trebuie folosită moştenirea şi adăugarea de componente noi pentru extinderea funcţionalităţii.

MVC în Symfony. Structura proiectului.



Symfony este bazat pe paradigma MVC şi obligă dezvoltatorul la păstrarea structurii MVC în toate proiectele în care este folosit. Structura proectului (punctele esenţiale) este următoarea:

```
apps/
  frontend/
    config/
    modules/
      <nume_modul>/
        actions/
          actions.class.php
        config/
        templates/
          indexSuccess.php
  templates/
config/
  doctrine/
    schema.yml
  databases.yml
data/
  fixtures/
    fixtures.yml
lib/
```


Search

- Repartizare teme
- Catalog PW

Laboratoare

- Laborator 01 - Introducere
- Laborator 02 - BD in PHP
- Laborator 03 - Arrays, Magic Methods
- Laborator 04 - (X)HTML, CSS
- Laborator 05 - Formulare HTML, Persistența Datelor
- Laborator 06 - Securitate I
- Laborator 07 - Securitate II
- Laborator 08 - JavaScript
- Laborator 09 - DOM scripting
- Laborator 10 - AJAX
- Laborator 11 - Web Frameworks

Teme

- Tema 01 - API pentru BD
- Tema 02 - Template System & Controller
- Tema 03 - Generator de formulare
- Tema 04 - Tree Web UI

Login

```
  filter/
  form/
  model/
web/
```

Proiectele în Symfony sunt împărţite în aplicaţii, care la rândul lor sunt împărţite în module. Modulele sunt legate de modele reale de date, prezente în baza de date. Folosing fişierele de configurare ale proiectului şi utilitarul din linie de comandă pus la dispoziţie de Symfony se defineşte legătura la baza de date, se generează tabele şi modulele aferente fiecărui model definit în proiect.

Fişierele corespunzătoare claselor model se găsesc în folderul `lib/model`. Controller-ul este reprezentat de clasele din fişierele `/apps/<application_name>/modules/<module_name>/actions/actions.class.php`. View-ul este reprezentat de fişierele template ale fiecărui modul (`/apps/<application_name>/modules/<module_name>/templates/<action_name><action_status>.php`) şi de layoutul aferent fiecărei aplicaţii (`/apps/<application_name>/templates/<layout_name>.php`).

Avantajul acestui model arhitectural este decuplarea dintre model, view şi controller. Beneficiile sunt multiple: codul devine mult bine structurat, lizibil, uşor de întreţinut şi testat, uşor de modificat şi de adăugat feature-uri noi.

De asemenea, Symfony vine cu o *separare pe medii de execuţie* a proiectelor. Se pot defini mai multe medii de execuţie (development, testing, production etc), fiecare cu parametrii proprii. De exemplu, mediul `development` poate folosi o conexiune la o bază de date MySQL şi anumite setări pentru debugging / logging, iar mediu de production să aibă un backend Oracle şi să aibă toate flagurile de debug setate la 0.

Toate aceste caracteristici ar trebui să fie prezente într-un proiect web de calitate. Faptul că Symfony le oferă "out of the box" înseamnă mai puţin timp petrecut în repetarea anumitor taskuri de rutină (de exemplu cel cu switching environments) şi mai mult timp în dezvoltarea şi îmbunătăţirea aplicaţiei propriu-zise.

Există şi alte framework-uri ce au acelaşi scop final: crearea unei structuri şi punerea la dispoziţie a unor biblioteci de funcţii care să facă munca dezvoltatorului mai uşoară. Exemple de alte framework-uri sunt: CakePHP, CodeIgniter, Kohana, Zend Framework. Dacă veţi ajunge să scrieţi multe aplicaţii web fără framework-uri veţi realiza că v-aţi creat un pseudo-framework al vostru :). Standardizarea modului de dezvoltare al aplicaţiilor într-un limbaj atât de permisiv cum este PHP este foarte importantă pentru calitatea codului şi viteza de dezvoltare, mai ales în proiecte care implică mai mulţi dezvoltatori.

Instalare

Symfony oferă un pachet destinat experimentării cu frameworkul. El conţine un proiect "self-contained" care funcţionează fără alte dependinţe. Pentru a-l instala pe maşinile din laborator, urmaţi paşii de mai jos.

1. Descărcaţi această arhivă de pe site-ul proiectului. Dezarhivaţi-o în folderul www din rădăcina serverului web.
2. Redenumiţi folderul `sf_sandbox` în `<nume>`, unde <nume> este numele vostru.
3. Adăugaţi un virtual host pentru sandbox
   I. modificaţi fişierul \Windows\system32\drivers\etc\hosts, adăugând linia:
   ```
   127.0.0.1       <nume>
   ```
   II. modificaţi fişierul httpd.conf, decomentând liniile
   ```
   [...]
   #LoadModule rewrite_module modules/mod_rewrite.so
   [...]
   #Include conf/extra/httpd-vhosts.conf
   [...]
   ```
   III. modifcaţi fişierul `<Director_wamp>bin/apache/<Versiune_Apache>/conf/extra/httpd-vhosts.conf`; el va trebui să conţină doar:
   ```
   NameVirtualHost *:80
   <VirtualHost *:80>
       ServerName localhost
       DocumentRoot "C:\wamp\www"
   </VirtualHost>


   <VirtualHost *:80>
       ServerName <nume>
       DocumentRoot "C:\wamp\www\<nume>\web"
    <Directory "C:\wamp\www\<nume>\web">
     AllowOverride All
     Allow from All
    </Directory>

    Alias /sf "C:\wamp\www\<nume>\lib\vendor\symfony\data\web\sf"
    <Directory "C:\wamp\www\<nume>\lib\vendor\symfony\data\web\sf">
   ```

```
  AllowOverride All
  Allow from All
 </Directory>
</VirtualHost>
```

4. adăugaţi folderul php în PATH-ul Windows (Control Panel → System → Advanced → Environment Variables)
5. reporniţi serverul HTTP şi accesaţi url-ul:

```
http://<nume>/
```

Ar trebui să vedeţi prima pagină a unui proiect Symfony.

Modelarea datelor

Symfony propune un model de dezvoltare "database agnostic". Frameworkul vine împreună cu două biblioteci de ORM: Propel şi Doctrine. Oricare dintre ele poate fi folosit într-un proiect Symfony. Avantajul adus de aceste biblioteci este maparea automată a datelor în obiecte definite de programator. În acest laborator, vom folosi Doctrine, deoarece sandbox-ul vine cu Doctrine preconfigurat.

În mod normal, conexiunea la baza de date este configurată prin fişierul `config/databases.yml`. Sandbox-ul vine preconfigurat cu o bază de date SQLite şi de aceea nu este necesară modificarea acestui fişier în afara căii către fişierul .db care va reprezenta baza voastră de date:

```
dsn: 'sqlite://C:\wamp\www\<nume>\config\<nume_fisier_db>.db'
```

schema.yml

Structura datelor din proiectul vostru trebuie definită în fişierul `schema.yml` din folderul `config/doctrine/`. YAML este formatul fişierelor de configurare symfony. Exemplul de mai jos este un exemplu de `schema.yml`:

schema.yml

```
# config/doctrine/schema.yml
JobeetCategory:
  actAs: { Timestampable: ~ }
  columns:
    name: { type: string(255), notnull: true, unique: true }

JobeetJob:
  actAs: { Timestampable: ~ }
  columns:
    category_id:  { type: integer, notnull: true }
    type:         { type: string(255) }
    company:      { type: string(255), notnull: true }
    logo:         { type: string(255) }
    url:          { type: string(255) }
    position:     { type: string(255), notnull: true }
    location:     { type: string(255), notnull: true }
    description:  { type: string(4000), notnull: true }
    how_to_apply: { type: string(4000), notnull: true }
    token:        { type: string(255), notnull: true, unique: true }
    is_public:    { type: boolean, notnull: true, default: 1 }
    is_activated: { type: boolean, notnull: true, default: 0 }
    email:        { type: string(255), notnull: true }
    expires_at:   { type: timestamp, notnull: true }
  relations:
    JobeetCategory: { onDelete: CASCADE, local: category_id, foreign: id, foreignAlias: Jobee

JobeetAffiliate:
  actAs: { Timestampable: ~ }
  columns:
    url:       { type: string(255), notnull: true }
    email:     { type: string(255), notnull: true, unique: true }
    token:     { type: string(255), notnull: true }
    is_active: { type: boolean, notnull: true, default: 0 }
  relations:
    JobeetCategories:
      class: JobeetCategory
      refClass: JobeetCategoryAffiliate
      local: affiliate_id
      foreign: category_id
      foreignAlias: JobeetAffiliates

JobeetCategoryAffiliate:
  columns:
    category_id:  { type: integer, primary: true }
    affiliate_id: { type: integer, primary: true }
  relations:
```

```
    JobeetCategory:  { onDelete: CASCADE, local: category_id, foreign: id }
    JobeetAffiliate: { onDelete: CASCADE, local: affiliate_id, foreign: id }
```

După ce aţi creat fişierul, îl puteţi folosi pentru a vă **genera baza de date**. Acest lucru se face prin comenzile:

```
symfony doctrine:build --model
symfony doctrine:build --forms
symfony doctrine:build --sql
symfony doctrine:insert-sql
```

Reţineţi că se poate şi invers, să se genereze schema.yml din structura unei baze de date existente. Aceast lucru se face prin comanda:

```
symfony doctrine:build-schema
```

fixtures.yml

Symfony permite popularea iniţială (pentru development) a bazei de date din fişiere de configurare:

categories.yml

```
# data/fixtures/categories.yml
JobeetCategory:
  design:
    name: Design
  programming:
    name: Programming
  manager:
    name: Manager
  administrator:
    name: Administrator
```

jobs.yml

```
# data/fixtures/jobs.yml
JobeetJob:
  job_sensio_labs:
    JobeetCategory: programming
    type:         full-time
    company:      Sensio Labs
    logo:         sensio-labs.gif
    url:          http://www.sensiolabs.com/
    position:     Web Developer
    location:     Paris, France
    description:  |
      You've already developed websites with symfony and you want to work
      with Open-Source technologies. You have a minimum of 3 years
      experience in web development with PHP or Java and you wish to
      participate to development of Web 2.0 sites using the best
      frameworks available.
    how_to_apply: |
      Send your resume to fabien.potencier [at] sensio.com
    is_public:    true
    is_activated: true
    token:        job_sensio_labs
    email:        job@example.com
    expires_at:   '2010-10-10'

  job_extreme_sensio:
    JobeetCategory:  design
    type:         part-time
    company:      Extreme Sensio
    logo:         extreme-sensio.gif
    url:          http://www.extreme-sensio.com/
    position:     Web Designer
    location:     Paris, France
    description:  |
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
      eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
      enim ad minim veniam, quis nostrud exercitation ullamco laboris
      nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
      in reprehenderit in.

      Voluptate velit esse cillum dolore eu fugiat nulla pariatur.
      Excepteur sint occaecat cupidatat non proident, sunt in culpa
      qui officia deserunt mollit anim id est laborum.
```

```
    how_to_apply: |
      Send your resume to fabien.potencier [at] sensio.com
    is_public:    true
    is_activated: true
    token:        job_extreme_sensio
    email:        job@example.com
    expires_at:   '2010-10-10'
```

Datele din aceste fişiere pot fi inserate în baza de date folosind comanda:

```
symfony doctrine:data-load
```

Generarea modulelor

Symfony poate autogenera module pe baza modelelor descrise în `schema.yml`. Aceste module conţin operaţiunile de bază ce pot fi efectuate asupra datelor (index, show, new, create, edit, update, delete). Folosiţi comanda de mai jos pentru a vă crea un modul:

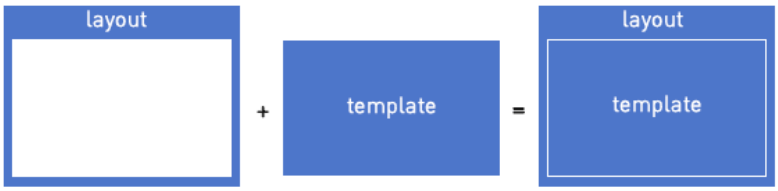
```
symfony doctrine:generate-module --with-show --non-verbose-templates frontend job JobeetJob
```

Fişierele modulului sunt create în calea `<proiect>/apps/frontend/modules/<nume_modul>`. Puteţi verifica modulul în browser la adresa:

```
http://<nume>/frontend_dev.php/job
```

Layout şi template

Pentru includerea elementelor generale în fiecare pagină, Symfony foloseşte design-pattern-ul `decorator`. Asta înseamnă că template-ul este "decorat" numai după ce este inclus într-un **layout** mai mare, deja randat. Avantajul este că elementele sunt "complete", într-o anumită măsură (atât layoutul cât şi template-ul sunt independente una faţă de cealaltă).



Creaţi un layout pentru proiectul vostru, care să conţină următorul cod:

layout.php

```
<!-- apps/frontend/templates/layout.php -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Jobeet - Your best job board</title>
    <link rel="shortcut icon" href="/favicon.ico" />
    <?php include_javascripts() ?>
    <?php include_stylesheets() ?>
  </head>
  <body>
    <div id="container">
      <div id="header">
        <div class="content">
          <h1><a href="<?php echo url_for('job/index') ?>">
            <img src="/images/logo.jpg" alt="Jobeet Job Board" />
          </a></h1>

          <div id="sub_header">
            <div class="post">
              <h2>Ask for people</h2>
              <div>
                <a href="<?php echo url_for('job/index') ?>">Post a Job</a>
              </div>
            </div>

            <div class="search">
              <h2>Ask for a job</h2>
              <form action="" method="get">
                <input type="text" name="keywords"
                  id="search_keywords" />
                <input type="submit" value="search" />
                <div class="help">
                  Enter some keywords (city, country, position, ...)
```

```
              </div>
            </form>
          </div>
        </div>
      </div>

      <div id="content">
        <?php if ($sf_user->hasFlash('notice')): ?>
          <div class="flash_notice">
            <?php echo $sf_user->getFlash('notice') ?>
          </div>
        <?php endif ?>

        <?php if ($sf_user->hasFlash('error')): ?>
          <div class="flash_error">
            <?php echo $sf_user->getFlash('error') ?>
          </div>
        <?php endif ?>

        <div class="content">
          <?php echo $sf_content ?>
        </div>
      </div>

      <div id="footer">
        <div class="content">
          <span class="symfony">
            <img src="/images/jobeet-mini.png" />
            powered by <a href="http://www.symfony-project.org/">
            <img src="/images/symfony.gif" alt="symfony framework" />
            </a>
          </span>
          <ul>
            <li><a href="">About Jobeet</a></li>
            <li class="feed"><a href="">Full feed</a></li>
            <li><a href="">Jobeet API</a></li>
            <li class="last"><a href="">Affiliates</a></li>
          </ul>
        </div>
      </div>
    </div>
  </body>
</html>
```

```
<!-- apps/frontend/modules/job/templates/indexSuccess.php -->

<div id="jobs">
  <table class="jobs">
    <?php foreach ($jobeet_jobs as $i => $job): ?>
      <tr class="<?php echo fmod($i, 2) ? 'even' : 'odd' ?>">
        <td class="location"><?php echo $job->getLocation() ?></td>
        <td class="position">
          <a href="<?php echo url_for('job/show?id='.$job->getId()) ?>">
            <?php echo $job->getPosition() ?>
          </a>
        </td>
        <td class="company"><?php echo $job->getCompany() ?></td>
      </tr>
    <?php endforeach ?>
  </table>
</div>
```

Încărcarea fişierelor CSS şi Javascript

Descărcaţi arhiva cu imagini şi arhiva cu stiluri şi dezarhivaţi-le în folderele `web/images`, respectiv `web/css`.

Pentru a adăuga fişiere javascript şi css în view, se modifică fişierul de configurare `<proiect>/apps/frontend/config/view.yml`. Adăugaţi css-urile descărcate în view prin linia:

```
stylesheets:    [main.css, jobs.css, job.css, print: { media: print }]
```

În acest caz, nu toate fişierele sunt necesare în fiecare pagină a fiecărui modul. Symfony mai pune la dispoziţie şi un mod alternativ de includere a css-urilor din fişiere de configurare:

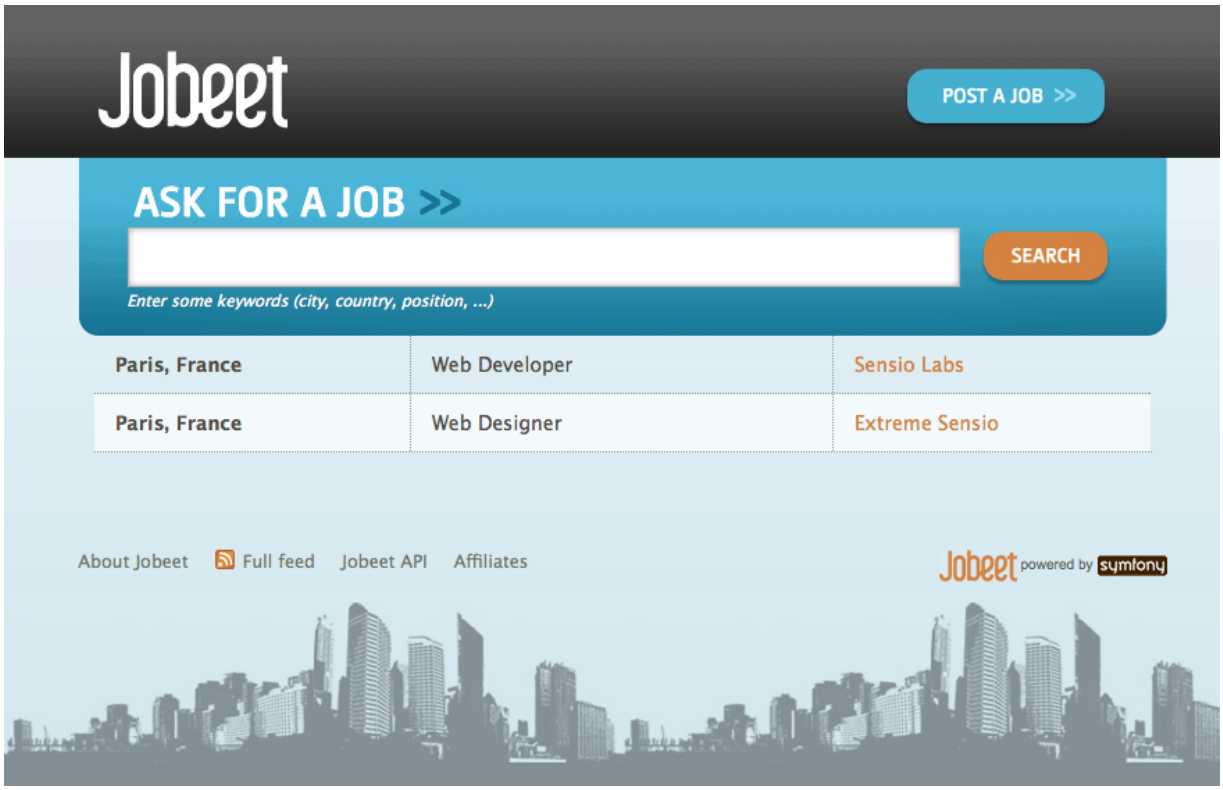
```
# apps/frontend/modules/job/config/view.yml
indexSuccess:
```

```
  stylesheets: [jobs.css]

showSuccess:
  stylesheets: [job.css]
```

Acum, fişierul `jobs.css` va fi încărcat doar în cazul randării template-ului pentru acţiunea `index`, iar `job.css` doar pentru `show`. Se procedează analog pentru fişierele javascript.

Acum, proiectul vostru ar trebui să arate ca în imaginea de mai jos:



Taskuri

1. Urmaţi paşii din acest laborator pentru a avea un proiect funcţional instalat pe sistemul vostru. **(5p)**
2. Adăugaţi un modul aplicaţiei.
   I. Modificaţi `schema.yml`, adăugând structura datelor aferente modulului vostru. **(1p)**
   II. Modificaţi `fixtures` pentru a adăuga nişte date de test în baza de date. **(1p)**
   III. Folosiţi `symfony` în linie de comandă pentru a vă genera tot ce trebuie pentru ca modulul să fie funcţional. **(2p)**
   IV. Modificaţi modelul pentru a adăuga o metodă nouă pe care să o folosiţi în controller. **(1p)**
   V. Adăugaţi o acţiune nouă în controller, în afara celor default. (ce face ea este la latitudinea voastră) **(1p)**
   VI. Creaţi un view pentru acţiunea creată la punctul anterior. **(1p)**

Resurse

Laboratorul a fost construit pe baza unei părţi a tutorialului "Practical Symfony".

(cc) BY-NC-SA W3C CSS DOKUWIKI W3C XHTML 1.0 RSS XML FEED