SI3PSI

Lab 3: Objektno relaciono mapiranje

Cilj ove laboratorijske vežbe je da demonstrira upotreba alata za objektno-relaciono mapiranje (*eng. ORM*), na primeru Doctrine (doctrine.org), uz integraciju sa Codeigniter MVC framework-om.

Deo 1. Kreiranje klasa

Napraviti klasu `Person`, sa sledećim sadržajem:

```
<?php

namespace Entity;

/**
 * User Model
 *
 * @Entity
 * @Table(name="person")
 */
class Person {

    /**
     * @Id
     * @Column(type="integer", nullable=false)
     * @GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @Column(type="string", length=80, unique=true, nullable=false)
     */
    protected $name;

    /**
     * @Column(type="string", length=255, unique=true, nullable=false)
     */
    protected $email;

    public function setName($name) {
        $this->name = $name;
        return $this;
    }

    public function setEmail($email) {
        $this->email = $email;
        return $this;
    }

    public function getId() {
        return $this->id;
    }

    public function getName() {
        return $this->name;
    }

    public function getEmail() {
        return $this->email;
    }

}
```

Potom izvršiti sledeću naredbu:

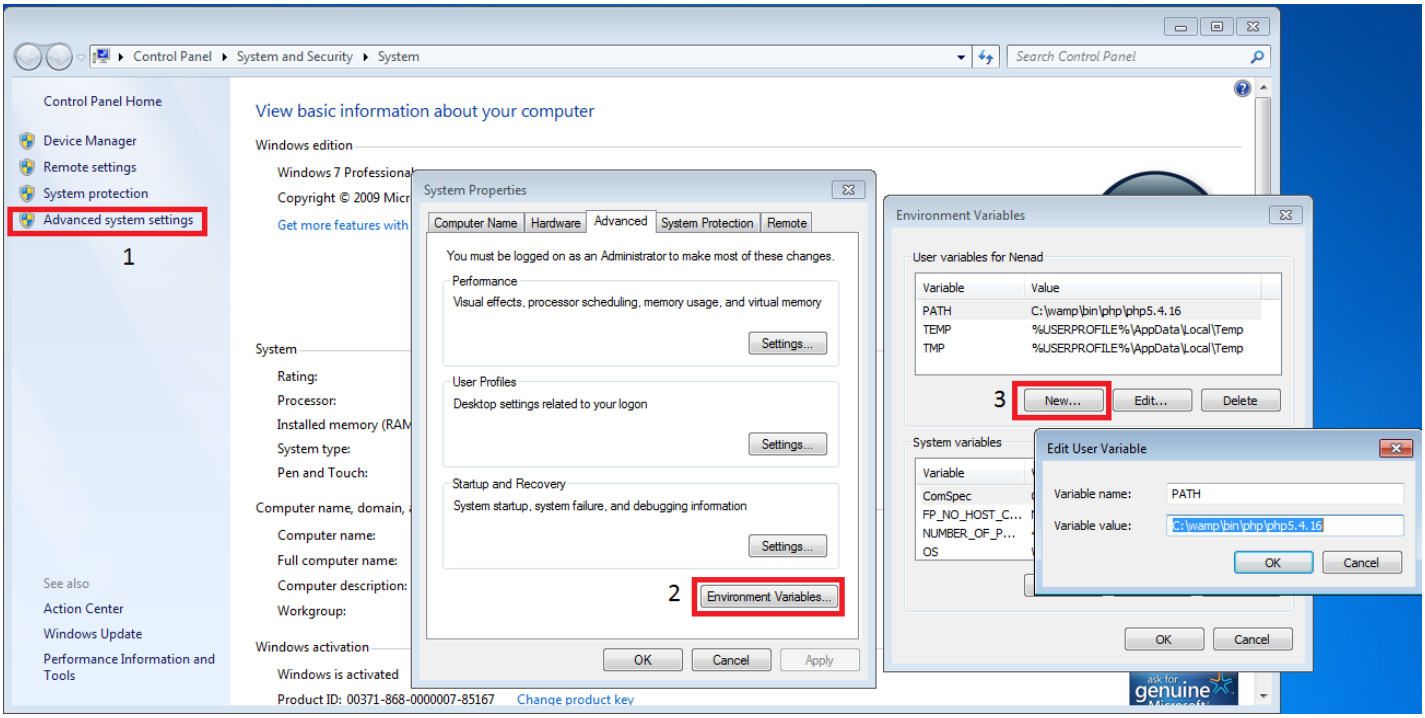
`orm:schema-tool:create --dump-sql`
Šta je rezultat izvršavanja? 'php' is not recognized as an internal or external command, operable program or batch file.

Podešavanje putanje do php.exe – potrebno je putanju do php.exe dodati u PATH:

Start> desni klik na „Computer“>Advanced system settings, Environment Variables > New

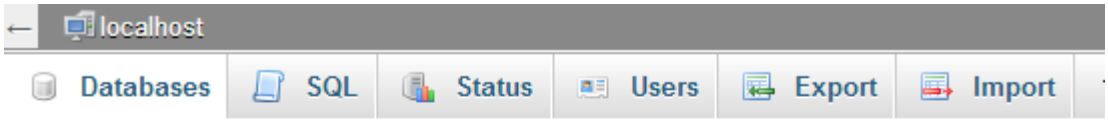
Dodati varijablu PATH sa vrednošću koja sadrži putanju do direktorijuma u kom je php.exe smešten (`C:\wamp\bin\php\php5.X.Y`).

Testirati u **novoj** konzoli (stara konzola neće „videti“ promene).



Deo 2. Kreiranje baze podataka i povezivanje sa DB

1. Otvoriti PHPMyAdmin (http://localhost/phpmyadmin)
2. Kreirati bazu PSIORM, uz odabir **utf_8_general_ci** za collation. Na šta to utiče?



3. Konfigurisati conf/databases.php kako je dato u sledćem listingu

Ponovo pokrenuti komandu `php application\doctrine orm:schema-tool:create`.

4. Pogledati sadržaj baze podataka (preko php-myadmin).

```
$active_group = 'default';
$active_record = FALSE;

$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'root';
$db['default']['password'] = ''; // prazno u lab, ubaciti Vaš password!!!
$db['default']['database'] = 'PSIORM'; //ime baze podataka (prethodno napravljena)
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = '';
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8_general_ci';
$db['default']['swap_pre'] = '';
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
```

Deo 3. Rad sa entitetima - perzistiranje

1. Napraviti kontroler `persons.php`, koji sadrži klasu `Persons`.
2. Napraviti akciju `index`
3. Sadržaj akcije indeks popuniti kao na sledećem listingu.

```
<?php

if (!defined('BASEPATH'))
    exit('No direct script access allowed');

class Persons extends CI_Controller {


    public function index() {
        $this->load->library('doctrine');
        $em = $this->doctrine->em;
        $data = array(
            array("Milutin Milankovic", "milutin@vasi1ona.com", true),
            array("Mihajlo Pupin", "mihajlo@pas1njak.com", true),
            array("Nikola Tesla", "nikola@nema1knjigu.com", false)
        );

        foreach ($data as $person) {
            $user = new Entity\Person();
            $user->setName($person[0]);
            $user->setEmail($person[1]);

            if ($person[2]) {
                $em->persist($user);
            }
        }
        $em->flush();
        $this->load->view('welcome_message');
    }

}

/* End of file welcome.php */
/* Location: ./application/controllers/welcome.php */
```

4. Otići na tu stranicu, tačno jedanput
5. Pogledati sadržaj tabele `person`.

Deo 4. Rad sa entitetima – validacija

1. Ponovo otići na prethodnu stranicu. Desila se greška zbog unique constraint-a.

2. Pogledati http://docs.doctrine-project.org/projects/doctrine1/en/latest/en/manual/data-validation.html#examples za spisak svih ugrađenih validatora. Pronaći gde se u modelu nalazi ova validacija.
3. Razrešenje: poziv metode `flush()` ubaciti u try-catch blok. Prilikom `flush` se promene "spuštaju" na BP, pa tada mogu da se pojave exception-i.
4. Ponovo otići na stranicu.
5. Da bi se izbegle ovakve situacije, neophodno je raditi proveru pre insert-a:

```
if ($person[2] && !$this->already_contained($user)) {
  $em->persist($user);
}
//…
private function already_contained($person){
        return $this->doctrine->em
                        ->getRepository('Entity\Person')
                        ->findBy(array('name' => $person->getName())) !=null;
    }
```

Deo 5. Pisanje upita – bazični upiti putem find/findAll

1. Dohvatiti kolekciju svih postojećih korisnika u akciji `index`, i proslediti view-u

```
$this->doctrine->em->getRepository('Entity\Person')->findAll()
```

2. Izmeniti view, tako da prikazuje podatke o svim osobama, i link koji vodi ka akciji `persons/profile`, koja prikazuje podatke o osobi

```
<?php
$this->load->helper('url_helper');
foreach ($persons as $person) {
    echo "<li>" . $person->getName();
    echo anchor('persons/profile/'.$person->getId());
    echo "</li>";
}
?>
```

3. Napraviti akciju profile, i view profile, kao u priloženom kodu:

```
    // akcija persons/profile
    public function profile($id) {
        $this->load->library('doctrine');
        $em = $this->doctrine->em;
        $person = $em->find("Entity\Person", $id);
        $this->load->view('profile', array('person' => $person));
    }
```

```
<?php
  echo $person->getName() . ", " . $person->getEmail() . ", ".$person->getId();
// views/profile.php
?>
```

Deo 6. Izmena modela

1. Dodati u model podatak `age`, uz odgovarajuće getter-e i setter-e
2. Dodati mapiranje za kolonu:

```
    /**
     * @Column(type="integer", nullable=false)
     */
```

3. Izvršiti komandu

```
php doctrine orm:validate-schema
```

4. Ukoliko je sve u redu, može se generisati SQL. Izvršiti komandu

```
php doctrine orm:schema-tool:update
```

5. Izvršiti istu komandu, iz opciju `--force`.
6. Pogledati sadržaj baze podataka.
7. Izmeniti niz sa naučnicima u welcome/index akciji, tako da se mejlovi razlikuju. Otići na tu akciju. Šta se dogodilo? Srediti problem, tako da se unesu novi podaci u bazu, tako što se za svakog naučnike obezbedi da imaju 56, 67, 75 godina, redom. Takođe, obezbedite da svi naučnici iz liste budu sačuvani (i Nikola Tesla).

Deo 7. Pisanje složenijih upita

Napisati upit koji dohvata sve naučnike koji imaju između 50 i 70 godina.

1. Putem Query Builder-a:

```
        $em = $this->doctrine->em;
        $qb = $em->createQueryBuilder();
        $qb->select('p')
                ->from('Entity\Person', 'p')
                ->where('p.age >= :ageMin and p.age <= :ageMax')
                ->orderBy('p.name', 'ASC')
                ->setParameter('ageMin', 50)
                ->setParameter('ageMax', 70);
        $q = $qb->getQuery();
        $result = $q->getResult();
```

2. Putem Doctrine Query Language

```
    public function ageTestDQL() {
        $this->load->library('doctrine');
        $em = $this->doctrine->em;
        $q = $em->createQuery(
             "select p from Entity\Person p where p.age >= :ageMin and p.age <= :ageMax");
// korišćenje imenovanih parametara
        $q->setParameter('ageMin', 50)
          ->setParameter('ageMax', 70); // postavljanje vrednosti imenovanim parametrima
        $result = $q->getResult();

        foreach ($result as $value) {
            echo $value->getName();
        }
    }
```

3. Putem Criteria. Criteria pruža način da se specificira koji uslovi moraju biti zadovoljeni. Primenjuje se na neku od kolekcija: bilo na kolkekciju u memoriji ili u BP (u prvom slučaju, radi sa objektima u memoriji, u drugom, generišu se odgovarajući SQL upiti)

```
    public function ageTestCriteria() {
        $this->load->library('doctrine');
        $em = $this->doctrine->em;
        $criteria = Criteria::create()
                ->where(Criteria::expr()->gt("age", 50))
                ->andWhere(Criteria::expr()->lt("age", 70))
                ->orderBy(array("name" => Criteria::ASC))
                ->setFirstResult(0)
                ->setMaxResults(20)
        ;
        $result = $em->getRepository("Entity\Person")->matching($criteria);
        foreach ($result as $value) {
            echo $value->getName();
        }
    }
```

Deo 8. Asocijacije

1. Dodati entitet `Book`, sa sledećim sadržajem:

```
<?php

namespace Entity;

/**
 * User Model
 *
 * @Entity
 * @Table(name="book")
 */
class Book {

    /**
     * @Id
     * @Column(type="integer", nullable=false)
     * @GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @Column(type="string", nullable=false)
     */
    protected $username;

    /**
     * @ManyToOne(targetEntity = "Person", inversedBy = "booksAuthored")
     */
    protected $primaryAuthor;

    /**
     *
     * @ManyToMany(targetEntity="Person",
     *       inversedBy="booksCoAuthored")
     */
    protected $coauthors;

    // Napisati jednostavne getter i setter funkcije!!!

    public function setPrimaryAuthor($primaryAuthor) {
        $this->primaryAuthor = $primaryAuthor;
        $primaryAuthor->authoredBook($this);
    }

    public function setPrimaryAuthor($primaryAuthor) {
        $this->primaryAuthor = $primaryAuthor;
        $primaryAuthor->authoredBook($this); //nije obavezno
    }

    public function addToCoauthor($coauthor) {
        $this->coauthors []= $coauthor;
        // obavezno, inače se asocijacija ne vidi
        // dok se flush() ne pozove
        $coauthor->coauthoredBook($this);
    }
}
```

2. Dodati sledeća dva polja i metodu u entitet `Person`; napisati getter-e i setter-e.

```
/**
 * @OneToMany(targetEntity = "Book", mappedBy = "primaryAuthor")
 */
protected $booksAuthored;

/**
 * @ManyToMany(targetEntity="Book", mappedBy="coauthors")
 * @var Book[]
 */
protected $booksCoAuthored = null;

public function authoredBook($book){
   $this->booksAuthored[] = $book;
}

 public function coauthoredBook($book) {
    $this->booksCoAuthored[] = $book;
}
```

3. Ponoviti korak 5 iz dela 6.
4. Pogledati sadržaj baze podataka.

Deo 9. Kreiranje asocijacije

1. U kontroleru `persons`, napraviti akciju addBook, kao u priloženom kodu:

```
public function addBook($primaryAuthorId, $bookName, $secondAuthorId){
    $this->load->library('doctrine');
    $em = $this->doctrine->em;
    $primAuthor = $em->getRepository("Entity\Person")->find($primaryAuthorId);
    $secAuthor = $em->getRepository("Entity\Person")->find($secondAuthorId);
    echo $primAuthor->getName();
    $book = new Entity\Book();
    $book->setName($bookName);
    $book->setPrimaryAuthor($primAuthor);
    $book->addToCoauthor($secAuthor);
    $em->persist($book);
    echo $book->getName();
    $this->load->view('profile', array('person'=>$primAuthor));
    $em->flush();
}
```

2. Otići na akciju preko linka: `index.php/persons/addBook/1/Vasiona/2`
3. Izmeniti profil.php, tako da za svakog korisnika postoji listanje knjiga kojima je autor, i da se za svakog koautora knjige koja se upravo dodaje izlistaju knjige na kojima je takođe bio koautor.

```
<div id="body">

    <?php
    echo $person->getName() . ", " . $person->getEmail() . ", " . $person->getId() .
                              ", " . $person->getAge();
    echo "<ul>";
    // listanje knjiga primarnog autora
    foreach ($person->getBooksAuthored() as $book) {
        echo "<li>" . $book->getName() . "</li>";
    }
    echo "</ul>";

    echo "<h2> coauthors</h2>";
    $coauthors = $book->getCoauthors();
    // listanje knjiga svakog od koautora
    foreach ($coauthors as $coauth) {
        echo $coauth->getName();
        echo "<ul>";
        // listanje knjiga kojima je coauth takođe koautor.
        foreach ($coauth->getBooksCoauthored() as $bk) {
            echo "<li>" . $bk->getName() . "</li>";
        }
        echo "</ul>";
    }
    ?>

    <br/>
</div>
```

4. Da li se upravo dodata knjiga pojavila?
5. Staviti pod komentar poziv funkcije `addToCoauthor` unutar metode `Person::addToCoauthor`. Da li se sada pojavila u spisku knjiga koautora? Ponovo posetiti stranicu. Da li se sada pojavila? Zašto?

Deo 10. Pohlepno učitavanje

Dodati sledeće dve akcije u kontroler `Persons`.

```
public function listBooks($primaryAuthorId) {
    $this->load->library('doctrine');
    $em = $this->doctrine->em;
    $q = $em->createQuery("select p from Entity\Person p where p.id=:id");
    $q->setParameter('id', $primaryAuthorId);
    $p = $q->getSingleResult();
    $this->load->view('profile', array('person' => $p));
}

public function listBooksEager($primaryAuthorId) {
    $this->load->library('doctrine');
    $em = $this->doctrine->em;
    $q = $em->createQuery("select p from Entity\Person p where p.id=:id");
    $q->setParameter('id', $primaryAuthorId);
    $q->setFetchMode("Entity\Person", "booksAuthored", "EAGER");
    $p = $q->getSingleResult();
    $this->load->view('profile', array('person' => $p));
}
```

5. Analizirati vremena izvršavanja (u donjem desnom uglu stranice stoji vreme izvršavanja).

Deo 11. Join

Dohvatiti samo one osobe koja imaju makar jednu knjigu na kojoj su primarni autori.

Dodati i posetiti sledeću akciju

```
public function listAuthorsWithBooksWritten(){
    $this->load->library('doctrine');
    $em = $this->doctrine->em;
    $q = $em->createQuery('SELECT p FROM Entity\Person p JOIN p.booksAuthored');
    $result = $q->getResult();
    foreach ($result as $author){
        echo $author->getName();
    }
}
```