Example code , Tuulia Antonius

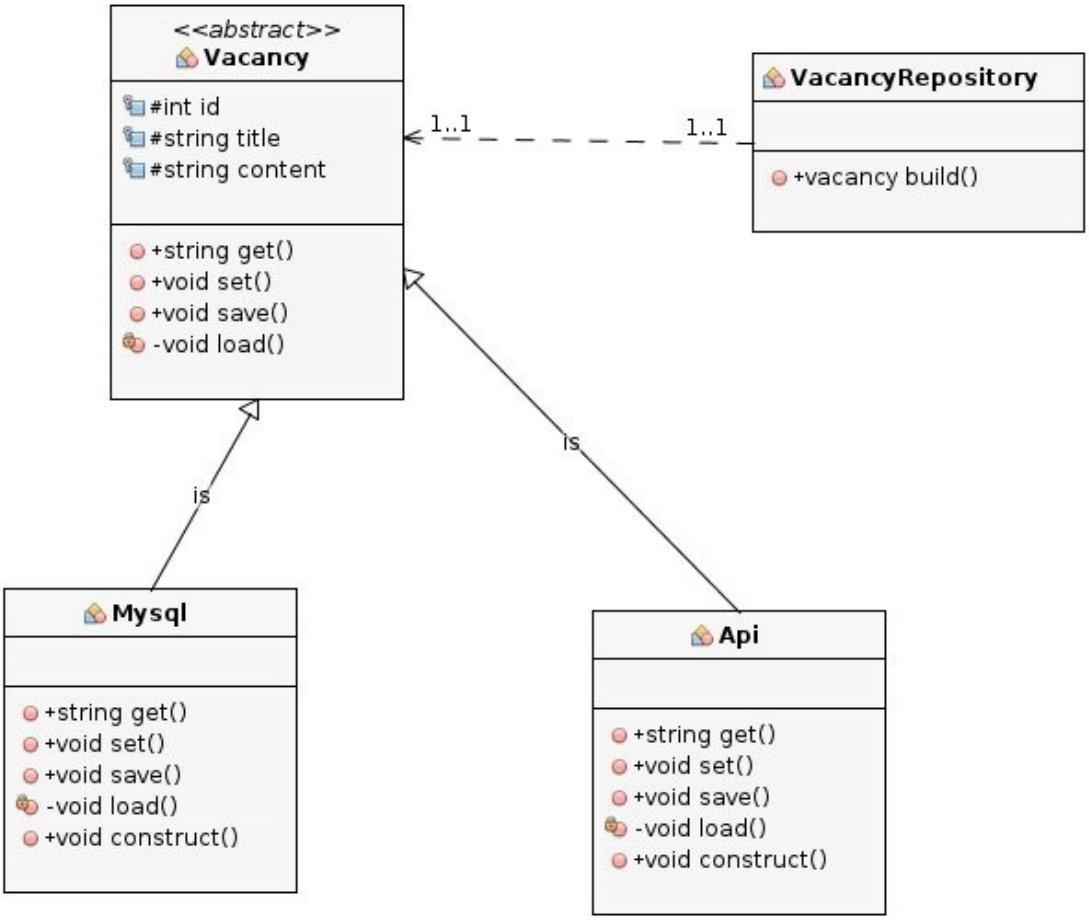
Table of Contents

Introduction

In this example I present my solution for a system in which I want to get vacancy data from two different data sources. I can get the data from the database or from an external source through an API. It is possible to change the data source dynamically.
I present here the architecture of the most important classes, code and the unit tests.

Architecture

The class 'VacancyFactory' presents the Factory design pattern. Here I can build an object which connects to Mysql database or to API.



I have four main methods

Get($property)
Get the value of of the given property.

Set
Set **t**he value of of the given property.

Load
Load the data for the vacancy from the data source

Save
Load the data of the vacancy to the data source

Code

Unit tests

```php
<?php
/**
 * A test class to test different data sources
 *
 */
require_once '../global.php';
use Lib\Factor\Vacancy;
class VacancyTest extends PHPUnit_Framework_TestCase
{
    /**
     * Set up
     */
    protected function setUp() {
        Lib\Models\Vacancy::resetDataSource();
    }
    /**
     * test Mysql datasource
     */
    public function testMysql() {
        // Test get
        $testVacancyId = 1;
        $vacancy = Vacancy::build('Mysql', $testVacancyId);
        $this->assertEquals('Vacancy 1', $vacancy->get('title'));
        // Test save
        $newTitle = 'Cool';
        $vacancy->set('title' , $newTitle);
        $vacancy->save();
        // load again
        $vacancySaved = Vacancy::build('Mysql', $testVacancyId);
        $this->assertEquals($newTitle, $vacancySaved->get('title'));
        // Test insert new vacancy
        $newTitle = 'New Cool';
        $vacancyNew = Vacancy::build('Mysql');
        $vacancyNew->set('title' , $newTitle);
        $newVacancyId =$vacancyNew->save();
        // load again
        $vacancyNewSaved = Vacancy::build('Mysql', $newVacancyId);
        $this->assertEquals($newTitle, $vacancyNewSaved->get('title'));
    }
    /**
     * test Api datasource
     */
    public function testApi() {
        // Test get
        $testVacancyId = 2;
        $vacancy = Vacancy::build('Api', $testVacancyId);
        $this->assertEquals('API Vacancy 2', $vacancy->get('title'));
        // Test save
        $newTitle = 'Cool Api';
        $vacancy->set('title' , $newTitle);
        $vacancy->save();
        // load again
        $vacancySaved = Vacancy::build('Api', $testVacancyId);
        $this->assertEquals($newTitle, $vacancySaved->get('title'));
        // Test insert new vacancy
```

```
        $newTitle = 'New Cool Api';
        $vacancyNew = Vacancy::build('Api');
        $vacancyNew->set('title' , $newTitle);
        $newVacancyId =$vacancyNew->save();
        // load again
        $vacancyNewSaved = Vacancy::build('Api', $newVacancyId);
        $this->assertEquals($newTitle, $vacancyNewSaved->get('title'));
    }
}
```

VacancyFactor

```php
<?php
/**
 * A factory class to create a data source object
 *
 */
namespace Lib\Factor;
class Vacancy {
    /**
     * build
     *
     * Build a datasource object for given datasource type
     * If id is given, read the data
     *
     * @param string $dataSource
     * @param type $id
     * @return \Lib\Factor\dataSource
     * @throws Exception
     */
    static public function build($dataSource, $id = false) {
        $dataSource = 'Lib\Models\Vacancy\\' . $dataSource;
        if (class_exists($dataSource)) {
            return new $dataSource($id);
        } else {
            throw new Exception("Invalid product type given.");
        }
    }
}
```

Vacancy

```php
<?php
/**
 * An abstract class to be used by a specified data source class
 *
 */
namespace Lib\Models;
abstract class Vacancy
{
    /**
     * The id of the vacancy
     *
     * @var integer
     */
    protected $id;
    /**
     * The vacancy title
     *
     * @var string
     */
    protected $title;
    /**
     * The vacancy content/description
     *
     * @var string
     */
    protected $content;
    /**
     * The vacancy description
     *
     * @var string
     */
    protected $description;
    /**
     * __construct
     *
     * @param integer $vacancyId
     */
    abstract public function __construct($id = false) ;
    /**
     * Save the current data to the current data source
     */
    abstract public function save() ;
    /**
     *  Load the data for given vacancy
     *
     * @param integer $vacancyId
     */
    abstract protected function load($vacancyId);
    /**
     * Public function to get a property
     *
     * @param type $property
     * @return string
     */
```

```
    public function get($property) {
        if ($this->isMember($property)) {
            return $this->$property;
        }
    }
    /**
     *
     * @param type $property
     * @param type $value
     */
    public function set($property, $value) {
        if ($this->isMember($property)) {
            $this->$property = $value;
        }
    }
    /**
     * Check if the propery is a member of this class and can be read or edited
     *
     * @param type $property
     * @return boolean
     * @throws Exception
     */
    private function isMember($property) {
        // Define members which can be get and set
        $members = array_keys(get_object_vars($this));
        unset($members['testData']);
        unset($members['id']);
        unset($members['childClass']);
        // Check
        if (in_array($property, $members)) {
            return true;
        } else {
            throw new Exception('property ' . $property . ' not found.');
        }
    }
}
```

Mysql

```php
<?php
/**
 * A datasource class to handle transactions to a MySql database.
 *
 */
namespace Lib\Models\Vacancy;
use Lib\Models\Vacancy;                          // Generic Vacacy Model
use Lib\Models\Mysql\Vacancy as sqlVacancy;      // Sql  Abstraction layer
class Mysql extends Vacancy {
    /**
     * __construct
     *
     * @param integer $vacancyId
     */
    public function __construct($vacancyId = false) {
        if ($vacancyId) {
            $this->load($vacancyId);
        }
    }
    /**
     * Here we would save the data to the database with SQL
     *
     * @return integer
     */
    public function save() {
        if (!$this->id) {
            // In case this is a new item define id and insert and get new id
            $vacancy = sqlVacancy::create();
            $vacancyId = $this->id;
        } else {
            // Load exisisting data
            $vacancy = sqlVacancy::load($vacancyId);
            $vacancyId = $this->id;
        }
        // Update
        $vacancy->title    = $this->title;
        $vacancy->content  = $this->content;
        $vacancy->save();
        return $vacancyId;
    }
    /**
     *  Load the data for given vacancy from database
     *
     * @param integer $vacancyId
     */
    protected  function load($vacancyId) {
        $vacancy = sqlVacancy::load($vacancyId);
        $this->id       = $vacancyId;
        $this->title    = $vacancy->title;
        $this->content  = $vacancy->content;
    }
}
```

API

```php
<?php
/**
 * A datasource class to handle transactions to a remote data source
 * by an API.
 *
 */
namespace Lib\Models\Vacancy;
use Lib\Models\Vacancy;
use Lib\Models\Vacancy as sqlVacancy;     // Vacancy API
class Api extends Vacancy {
    /**
     * __construct
     *
     * @param integer $vacancyId
     */
    public function __construct($vacancyId = false) {
        parent::initialize();
        if ($vacancyId) {
            $this->load($vacancyId);
        }
    }
    /**
     * Here we would save the data to  remote data source
     *
     * @return integer
     */
    public function save() {
        if (!$this->id) {
            // In case this is a new item define id and insert and get new id
            $vacancy = sqlVacancy::create();
            $vacancyId = $this->id;
        } else {
            // Load exisisting data
            $vacancy = sqlVacancy::load($vacancyId);
            $vacancyId = $this->id;
        }
        // Update
        $vacancy->title    = $this->title;
        $vacancy->content  = $this->content;
        $vacancy->save();
        return $vacancyId;
    }
    /**
     *  Load the data for given vacancy
     *
     *
     * @param integer $vacancyId
     */
    protected  function load($vacancyId) {
        $vacancy = sqlVacancy::load($vacancyId);
        $this->id       = $vacancyId;
        $this->title    = $vacancy->title;
        $this->content  = $vacancy->content;
    }
}
```