Integrating with CodeIgniter

This is recipe for using Doctrine 2 in your CodeIgniter (http://www.codeigniter.com) framework.

> This might not work for all CodeIgniter versions and may require slight adjustments.

Here is how to set it up:

Make a CodeIgniter library that is both a wrapper and a bootstrap for Doctrine 2.

Setting up the file structure

Here are the steps:

- Add a php file to your system/application/libraries folder called Doctrine.php. This is going to be your wrapper/bootstrap for the D2 entity manager.
- Put the Doctrine folder (the one that contains Common, DBAL, and ORM) inside that same libraries folder.
- Your system/application/libraries folder now looks like this:

  system/applications/libraries -Doctrine -Doctrine.php -index.html
- If you want, open your config/autoload.php file and autoload your Doctrine library.

  <?php $autoload[‘libraries’] = array(‘doctrine’);

Creating your Doctrine CodeIgniter library

Now, here is what your Doctrine.php file should look like. Customize it to your needs.

```php
<?php
use Doctrine\Common\ClassLoader,
    Doctrine\ORM\Configuration,
    Doctrine\ORM\EntityManager,
    Doctrine\Common\Cache\ArrayCache,
    Doctrine\DBAL\Logging\EchoSQLLogger;

class Doctrine {

  public $em = null;

  public function __construct()
  {
    // load database configuration from CodeIgniter
    require_once APPPATH.'config/database.php';

    // Set up class loading. You could use different autoloaders, provided by your favorite framework,
    // if you want to.
    require_once APPPATH.'libraries/Doctrine/Common/ClassLoader.php';

    $doctrineClassLoader = new ClassLoader('Doctrine',  APPPATH.'libraries');
    $doctrineClassLoader->register();
    $entitiesClassLoader = new ClassLoader('models', rtrim(APPPATH, "/" ));
    $entitiesClassLoader->register();
    $proxiesClassLoader = new ClassLoader('Proxies', APPPATH.'models/proxies');
    $proxiesClassLoader->register();

    // Set up caches
    $config = new Configuration;
    $cache = new ArrayCache;
    $config->setMetadataCacheImpl($cache);
    $driverImpl = $config->newDefaultAnnotationDriver(array(APPPATH.'models/Entities'));
    $config->setMetadataDriverImpl($driverImpl);
    $config->setQueryCacheImpl($cache);

    $config->setQueryCacheImpl($cache);

    // Proxy configuration
    $config->setProxyDir(APPPATH.'/models/proxies');
    $config->setProxyNamespace('Proxies');

    // Set up logger
    $logger = new EchoSQLLogger;
    $config->setSQLLogger($logger);

    $config->setAutoGenerateProxyClasses( TRUE );

    // Database connection information
    $connectionOptions = array(
        'driver' => 'pdo_mysql',
        'user' =>     $db['default']['username'],
        'password' => $db['default']['password'],
        'host' =>     $db['default']['hostname'],
        'dbname' =>   $db['default']['database']
    );

    // Create EntityManager
    $this->em = EntityManager::create($connectionOptions, $config);
  }
}
```

Please note that this is a development configuration; for a production system you'll want to use a real caching system like APC, get rid of EchoSqlLogger, and turn off autoGenerateProxyClasses.

For more details, consult the Doctrine 2 Configuration documentation (http://www.doctrine-project.org/documentation/manual/2_0/en/configuration#configuration-options).

Now to use it

Whenever you need a reference to the entity manager inside one of your controllers, views, or models you can do this:

```php
<?php
$em = $this->doctrine->em;
```

That's all there is to it. Once you get the reference to your EntityManager do your Doctrine 2.0 voodoo as normal.

Note: If you do not choose to autoload the Doctrine library, you will need to put this line before you get a reference to it:

```
<?php
$this->load->library('doctrine');
```

Good luck!