Design Patterns: The Registry Pattern

Ben Ramsey

Atlanta PHP • 5 Feb 2009

What Is a Design Pattern?

“A design pattern is a general reusable solution to a commonly occurring problem in software design. ...”

What Is a Design Pattern?

"... A design pattern is not a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations."

-- Wikipedia

Patterns We've Discussed

- Singleton
- Factory

Registry Pattern

- Application data store
- Acts as a “dictionary” of name/value pairs of data
- Can contain scalars, arrays, objects, etc.
- Pass it around to have a more-or-less “global” data store; sensitive to scope

```php
<?php

class Registry {

    protected $_store = array();

    public function register($label, $object)
    {
        if (!isset($this->_store[$label])) {
            $this->_store[$label] = $object;
        }
    }

    public function get($label)
    {
        if (isset($this->_store[$label])) {
            return $this->_store[$label];
        }
        return false;
    }

    /* ... */
<?php

    /* ... */

    public function unregister($label)
    {
        if (isset($this->_store[$label])) {
            unset($this->_store[$label]);
        }
    }

    public function has($label)
    {
        return isset($this->_store[$label]);
    }
}
```

Example Usage

```php
<?php

$registry = new Registry();
$db = new PDO('mysql:dbname=testdb;host=localhost');

// Store the object to the registry
$registry->register('db', $db);

/* ... */

// Meanwhile, in another part of your application
$db = $registry->get('db');

$results = $db->query('...');
```

The downside (or upside) is you have to pass the Registry object around.

Singleton Registry

- The store itself is a singleton
- Truly “global” data store
- No need to pass registry around, exists in all scopes

```php
<?php

class SingletonRegistry {

    protected static $_store = array();

    public function register($label, $object)
    {
        if (!isset(self::$_store[$label])) {
            self::$_store[$label] = $object;
        }
    }

    public function get($label)
    {
        if (isset(self::$_store[$label])) {
            return self::$_store[$label];
        }
        return false;
    }

    /* ... */
<?php

    /* ... */

    public function unregister($label)
    {
        if (isset(self::$_store[$label])) {
            unset(self::$_store[$label]);
        }
    }

    public function has($label)
    {
        return isset(self::$_store[$label]);
    }
}
```

Example Usage

```php
<?php

$db = new PDO('mysql:dbname=testdb;host=localhost;');

$registry = new SingletonRegistry();
$registry->register('db', $db);

class User
{
    public function __construct($id = null)
    {
        if (!is_null($id)) {
            $registry = new SingletonRegistry();
            $db = $registry->get('db');

            // Use the DB object to query the database for
            // the user record and populate the User object
        }
    }
}
```

Criticism

- The Registry is a kind of global variable; global variables create code smell
- Martin Fowler advocates the use of static methods for the Registry; this creates mixed feelings in developer communities