Prefab & Registry

Prefab is a factory wrapper for singleton classes. It uses the Registry class to store objects.

Namespace: `\`
File location: `lib/base.php`

Prefab

If you want one of your classes to be singleton, just make it extend the Prefab class:

```
class MyClass extends \Prefab {

  private $year;

  function getYear() {
    return $this->year;
  }

  function __construct() {
    $this->year=date('Y');
  }
}
```

This way, the class will be instantiated no more than once. To retrieve the single object, use the static `instance()` method:

```
// somewhere in the code
$obj=MyClass::instance(); // First call: a new object is created
echo $obj->getYear();

// somewhere else in the code
$obj=MyClass::instance(); // Second call: the object already exists and is simply ret
urned
echo $obj->getYear();
```

NB: Most F3 classes (Base, Cache, View, Template, Web, etc.) are derived from Prefab

Registry

Under the hood, each single object is stored in the Registry. The Registry accepts a few commands:

Get

Retrieve an object from the registry

```
$obj=\Registry::get('MyClass');
```

Set

Store an object into the registry

```
$obj=new MyClass();
\Registry::set('MyClass',$obj);
```

Clear

Remove an object from the registry

```
\Registry::clear('MyClass');
```

Exists

Check if an object is stored in the registry

```
if (\Registry::exists('MyClass'))
  echo 'Singleton instanciated';
else
  echo 'Singleton not instanciated';
```

Passing arguments

If you need to pass arguments during class instantiation, you can do it in two ways:

Using Prefab

```
// somewhere in the code
$obj = MyClass::instance($a,$b); // First call: a new object is created

// somewhere else in the code
$obj = MyClass::instance(); // Second call: the existing object is returned
```

Beware that the arguments passed to the *instance()* function after the first call will be ignored.

Using Registry

```
$obj1 = new MyClass($arg1);
\Registry::set('MyClass1',$obj1);

$obj2 = new MyClass($arg2);
\Registry::set('MyClass2',$obj2);
```