Class Loading

ClickBlocks uses autoloading mechanism of undefined classes, interfaces and traits. You do not need to follow any naming convention of classes or files. You can specify any file name with the class and place it in a folder with any name and it will still be found and connected to your script automatically.

For the same reason, it is possible to easily use the classes of third-party developers. You just place the class files in any directory of the project. No additional code changes are required.

The work of the class loader is based on searching classes within files of the project. When some unknown class, interface, or trait is found in a script, the loader searches for all classes in the allowed directories and files, and then places the information about the location of found classes in the file **classmap.php** (located in the root directory) as an associative array. The keys of this array are class names and the values are full paths to class files. The loader is active during initialization of the framework (method **init()** of class **CB**).

Configuration of the class loader

There are several methods of the class **CB** allowing flexible control loading of classes. For example, you can get an array of all the paths to the classes or set your class map:

```
// Initializes the framework and gets an instance of class CB.
$cb = CB::init();
// Gets the class map.
print_r($cb->getClassMap());
// Sets the own class map.
$cb->setClassMap(['foo\myclass1' => '/path/to/class1',
                  'foo\myclass2' => '/path/to/class2',
                  'foo\myclass3' => '/path/to/class3', ...], '/path/to/the/classmap/file.php');
```

> Note that the class names must be in lower case and should not have symbol "\" at the beginning.

By default, the loader looks for classes in the root directory and all its subdirectories. But what if you need to connect classes from other directories, for example, from a directory that is outside of the root directory of a site. In such cases, you should explicitly specify the directory in which to search for classes:

```
// Initializes the framework and gets an instance of class CB.
$cb = CB::init();
// Sets directories in which the class search occurs.
$cb['autoload']['directories'] = ['/path/to/directory1' => false,
                                  '/path/to/directory2' => false,
                                  '/path/to/directory3' => true,
                                  ...]);
// Gets directory list to search.
print_r($cb['autoload']['directories']);
```

For each folder, you can specify whether to search for classes in its subdirectories or not. Values of the directory array are boolean parameters that responsible for searching in subdirectories. Value TRUE corresponds the search in subdirectories.

You can also determine what specific files to look for classes. This can be done by setting the regular expression for file names. Only files whose names will match the regular expression will be included in the search for classes.

```
// Initializes the framework and gets an instance of class CB.
$cb = CB::init();
// Sets the mask for class file name.
$cb['autoload']['mask'] = '.+\.(php|html|inc)$';
// Gets the current mask.
echo $cb['autoload']['mask'];
```

The default file name mask is defined as "/.+.php$/i". That is, classes are searched only in files with php.

If you need to remove from the search a particular file or files (also as a directory), you can easily do this:

```
// Initializes the framework and gets an instance of class CB.
$cb = CB::init();
// Sets exclusions for files and directories.
$cb['autoload']['exclusions'] = ['/my/file1', '/my/file2', '/my/directory', ...];
// Gets exclusions.
print_r($cb['autoload']['exclusions']);
```

Besides of the files or directories that you define yourself, the framework does not include in search the following directories: .hg, .git and .svn

Manual loading of classes

If necessary, you can load any class manually:

```
// Initializes the framework and gets an instance of class CB.
$cb = CB::init();
// Loads class My\Namespace\Class.
$cb->loadClass('My\Namespace\Class');
```

If such class exists, the method **loadClass()** returns TRUE. Otherwise, the class loader tries to find the required classes. If it does not succeed the method returns FALSE.

You can also find all classes and create class map whenever you want.

```
// Initializes the framework and gets an instance of class CB.
$cb = CB::init();
// Searches all classes.
echo 'Found classes: ' . $cb->createClassMap();
```

Method **createClassMap()** returns the number of all found classes.

In some cases, you may need to change the logic of the class loader. To do this the framework provides the ability to set own function of class autoloading. Under the autoload function is understood some of the delegates that will be called whenever the php interpreter will meet unspecified class, interface, or trait. The delegate will take the name of an including class. The delegate will have to include file of the required class or produce an error that the class was not found.

The following example shows the definition of a custom startup method classes. In this method, a class **Foo** is replaced with a dummy object (this approach can be used, for example, when creating a mocking objects in unit-testing).

```
// Initializes the framework and gets an instance of class CB.
$cb = CB::init();
// Sets own autoload function.
$cb['autoload']['callback'] = 'loader';

...

// Defines the function of class autoloading.
function loader($class)
{
  // Replaces class Foo to mock class.
  if ($class == 'Foo') eval('class Foo {}');
  // Other classes are loaded by standard way.
  return CB::getInstance()->loadClass($class);
}
```

> Determining of your own autoload method does not change the mechanism of class autoloading (based on their search). It only allows you to define your own way to add classes to the script or the reaction to the loading of non-existent classes.

Class loading configuration settings

Besides the direct adjustment of the class loader you can set the appropriate configuration settings that allow the get the same result. All possible settings of the class loader are shown below.

For INI-configuration file:

```
...
[autoload]
search       = 1
unique       = 1
classmap     = "classmap.php"
mask         = "/.+\.php\z/i"
timeout      = 300
callback     = "MyClass::method"
directories  = "... some json here ..."
exclusions   = "... some json here ..."
```

For PHP-configuration file:

```
...
'autoload' => ['search'      => true,
               'unique'      => true,
               'classmap'    => 'classmap.php',
               'timeout'     => 300,
               'mask'        => '/^*.\.php$/i',
               'callback'    => 'MyClass::method',
               'directories' => ['/root' => true, '/application' => false],
               'exclusions'  => ['/tmp', '/utils']],
...
```

These configuration parameters are defined as follows:

- **search** - boolean parameter that determines whether undefined classes should be loaded automatically or not. At that, if this parameter is FALSE the exception will be thrown when the interpreter encounters an undefined class.
- **unique** - boolean parameter. If it is TRUE the exception will be thrown when one or more classes with the same names are found.
- **classmap** - path to the classmap file.
- **timeout** - the integer positive value greater, or equal to, zero that determines the number of seconds of the waiting unlocking of the classmap file. The classmap file can be locked when there are multiple requests for class search. In this case the only one process performs the file search, and the other processes sleep during **timeout** seconds. If after the expiration of the given time the classmap is still locked the script will try to search classes by itself.
- **mask** - the regular expression (PCRE compatible) that should match the name of files containing classes.
- **callback** - the delegate that specifies the custom method of class loading.
- **directories** - array of directories in which the class search should be performed.
- **exclusions** - array of files or directories that should be excluded from the class search.