Template Caching

The result of processing any template can be cached. You can enable caching of the template in two ways:

1. Via the additional constructor parameters:

   ```
   $tpl = new Template('/path/to/template', 3600, 'TCID', new Cache\Memory());
   ```

   The second constructor parameter determines lifetime (in seconds) of the template cache. The third parameter is unique identifier of the template cache.

   > If cache identifier is not set, md5 of template will be used as the unique cache identifier. However, this approach does not provide a 100% guarantee there is no conflict. Therefore it is recommended to explicitly specify the cache ID.

   The fourth optional parameter of the constructor - the cache object. If this parameter is not specified, the internal framework cache will be used.

2. Via calling of the appropriate methods and properties of the class:

   ```
   // Creates an object of the template engine.
   $tpl = new Template('/path/to/template', 3600);
   // Sets new cache object for template caching.
   $tpl->setCache(new Cache\File());
   // Sets the cache identifier.
   $tpl->cacheID = 'TCID';
   // Sets the cache expiration time.
   $tpl->cacheExpire = 600;
   // Sets the cache group name.
   $tpl->cacheGroup = 'tpls';
   ```

You can check whether the template cache exists as follows:

```
if ($tpl->isExpired()) // cache is expired
else // cache is not expired
```

Caching of nested templates

Since the template variables can be objects of any data type, we can specify them as objects of the template engine, creating templates of any level of nesting. These nested templates can also be cached.

Suppose that we have two templates. The main template:

```
Common Template: <?=rand(0, 1000000);?>
<?=$subTemplate;?>
```

And the template that will be embedded in the main template:

```
Sub Template: <?=rand(0, 1000000);?>
```

Creates the cache object and adds the object of nested template to variable **$subTemplate**. The main template is cached for 10 seconds, and the nested template is cached for 5 seconds.

```
// Creates an object of the main template engine.
$tpl = new Template('/path/to/first/template', 10, 'tpl1');
// Adds the nested template.
$tpl->subTemplate = new Template('/path/to/second/template', 5, 'tpl2');
```

Now we should invoke method **render()** of the main template engine. Suppose we have the following result:

```
Common Template: 35678
```

```
Sub Template: 732213
```

Subsequent calls will return the same result in the next 5 seconds. After the 5 seconds subtemplate cache is cleared and the template will be processed for the second time. При этом At the same time, the main template cache is not expired yet. And the result of calling of method **`render`** of the main template engine after 5 seconds will be, for example, so:

```
Common Template: 35678
Sub Template: 18342
```

Thus, caching subtemplate nothing to do with caching of external template, and vice versa.