

# cfcache

## Description

Speeds up page rendering when dynamic content does not have to be retrieved each time a user accesses the page. To accomplish this, cfcache creates temporary files that contain the static HTML returned from a ColdFusion page. You can use cfcache for simple URLs and URLs that contain URL parameters.

## Category

Other

## Implemented

## Usage Syntax

```
[action="String"]
[key="String"]
[username="String"]
[password="String"]
[protocol="String"]
[timespan="timespan"]
[idletime="timespan"]
[directory="String"]
[cachedirectory="String"]
[timeout="Any"]
[expireURL="String"]
[port="number"]
[id="String"]
[name="String"]
[cachenname="String"]
/>
```

[] = Optional attribute

## Attributes

Name	Type	Required	Default	Description
action	String	No		put   cache   clientcache   servercache   flush   optimal   content - check the examples below for details
key	String	No		
username	String	No		When required for basic authentication, a valid username.
password	String	No		When required for basic authentication, a valid password.
protocol	String	No		Specifies the protocol used to create pages from cache. Either http:// or https://. The default is http://.
timespan	timespan	No		The amount of time an element will be cached. After this time span the element will not be used anymore and the cache will be updated.
idletime	idletime	No		If a cached element will not be used within this time it will be invalidated.
directory	String	No		
cachedirectory	String	No		
timeout	Any	No		
expireURL	String	No		
port	number	No		
id	String	No		
name	String	No		This attribute contains the name of the variable in the current variables scope, that will be filled with the result of a call.
cachenname	String	No		This attribute contains the name of a cache defined in the Railo administrator.

The actions get and put allow you to save and read elements that are stored in the cache specified in the administrator under "template cache". This is not the CFM template cache but the cache used for generating HTML snippets or templates with the help of this tag. In addition to the functions cache\*() you can use the tag cfcache for the same purpose.

## Contents

- [cfcache](#)
- [Description](#)
- [Category](#)
- [Implemented](#)
- [Usage Syntax](#)
- [Attributes](#)
- [Example Usage](#)

## Example Usage

### put

The following example shows you how to put any object into the default Railo template cache.

```
action="put" id="idforthekey" value="#value2store#" timespan="#createTimeSpan(0,0,10,0)#" idletime="#createTimeSpan(0,0,1,0)#">
```

If you want to use a different cache provider defined in the admin you need to provide the *name* attribute as well which specifies the name of one of in the admin defined caches. So the example should look like this:

```
action="put" key="idforthekey" value="#value2store#" timespan="#createTimeSpan(0,0,10,0)#" idletime="#createTimeSpan(0,0,1,0)#" name="someCacheName">
```

### get

This example returns the key stored above from the default cache:

```
action="get" id="idforthekey" name="returnVar" cachename="someCacheName">
```

### cache

#### clientcache

#### servercache

#### flush

#### optimal

#### content

The action **content** allows you to use the end tag as well. This gives you much more flexibility about what things to be cached. You can cache a normal HTML snippet defined by the attribute key. The attribute key can be as individual as needed. As soon as the tag is called within the defined timespan with the same key, it will deliver the content generated exactly for this key.

```
action="content" key="#someKeyThatDistinguishesBetweenCachedValues#" timespan="#createTimeSpan(0,0,15,0)#">
Something time intense that generates HTML like a header for example
```

Here is an example for the same behavior as above but with the usage of a different cache defined in the Railo admin.

```
action="content" key="#someKeyThatDistinguishesBetweenCachedValues#" timespan="#createTimeSpan(0,0,15,0)#" name="cacheName">
Something time intense that generates HTML like a header for example
```

This is how you would assign the cache value to a variable:

```
variable="sCache">
  action="content" key="#irgendwaseindeutiges#" timespan="#createTimeSpan(0,0,15,0)#">
  Irgendetwas was einen output generiert
```