Cache

Overview

A HubzeroCacheManager object is available for managing cache data storage and retrieval.

The Cache service comes with a wrapper class to easily work with multiple cache storage driver instances from a single object.

```
$manager = new HubzeroCacheManager($app);
```

Throughout this documentation the Cache facade will be used as it provides a convenient, terse access to the underlying implementations of the cache manager.

Drivers

A number of cache drivers are available.

Custom Drivers

The extend method on the HubzeroCacheManager can be used to extend the cache facility. It is used to bind a custom driver resolver to the manager. The following example demonstrates how to register a new cache driver named "example":

```
Cache::extend('example', function($config)
{
    return new ExampleStore;
});
```

The first argument passed to the extend method is the name of the driver, which will correspond to the driver option in the config/cache.php configuration file. The second argument is a Closure that should return an HubzeroCacheStorageStorageInterface instance. The closure is passed an array of configuration values.

Retrieving Items

The get method on the Cache facade is used to retrieve items from the cache. If the item does

not exist in the cache, null will be returned.

```
$value = Cache::get('key');
```

A default value can be passed as a second argument to the get method. This value will be returned if the cache store fails to find an item associated with the specified key or the data had expired. The default may be also be a closure:

```
$value = Cache::get('key', 'default');

$value = Cache::get('key', function() { return 'default'; });
```

The all method can be used to retrieve **all** items in the cache store.

```
$data = Cache::all();
```

Checking For Item Existence

The has method may be used to determine if an item exists in the cache:

```
if (Cache::has('key'))
{
    //
}
```

Storing Items

The put method on the Cache object is used to store items in the cache. When placing an item in the cache, the number of minutes for which the value should be cached will also need to specified:

```
Cache::put('key', 'value', $minutes);
```

Alternatively, the add method will only add the item to the cache if it does not already exist in the cache store:

```
Cache::add('key', 'value', $minutes);
```

The forever method may be used to store an item in the cache permanently. These values must be manually removed from the cache using the forget method:

```
Cache::forever('key', 'value');
```

Removing Items

You may remove items from the cache using the forget method on the Cache object:

```
Cache::forget('key');
```

Everything may be removed from the cache store by calling the clean method.

```
Cache::clean();
```

To limit the clean method to specific group of cached data, such as just cached data for the Tags component, a cache group name may be passed. In the example below, this will only remove cached data for the "tags" cache group.

```
Cache::clean('tags');
```

Finally, there is a gc (for "Garbage Collection") method for removing expired data.

```
Cache::gc();
```