NAME

LWP::ConnCache − Connection cache manager

NOTE

This module is experimental. Details of its interface is likely to change in the future.

SYNOPSIS

```
use LWP::ConnCache;
my $cache = LWP::ConnCache->new;
$cache->deposit($type, $key, $sock);
$sock = $cache->withdraw($type, $key);
```

DESCRIPTION

The `LWP::ConnCache` class is the standard connection cache manager for LWP::UserAgent.

METHODS

The following basic methods are provided:

new

```
my $cache = LWP::ConnCache->new( %options )
```

This method constructs a new LWP::ConnCache object. The only option currently accepted is `total_capacity`. If specified it initializes the ‘‘total_capacity’’ in LWP::ConnCache option. It defaults to `1`.

total_capacity

```
my $cap = $cache->total_capacity;
$cache->total_capacity(0); # drop all immediately
$cache->total_capacity(undef); # no limit
$cache->total_capacity($number);
```

Get/sets the number of connection that will be cached. Connections will start to be dropped when this limit is reached. If set to `0`, then all connections are immediately dropped. If set to `undef`, then there is no limit.

capacity

```
my $http_capacity = $cache->capacity('http');
$cache->capacity('http', 2 );
```

Get/set a limit for the number of connections of the specified type that can be cached. The first parameter is a short string like `"http"` or `"ftp"`.

drop

```
$cache->drop(); # Drop ALL connections
# which is just a synonym for:
$cache->drop(sub{1}); # Drop ALL connections
# drop all connections older than 22 seconds and add a reason for it!
$cache->drop(22, "Older than 22 secs dropped");
# which is just a synonym for:
$cache->drop(sub {
    my ($conn, $type, $key, $deposit_time) = @_;
    if ($deposit_time < 22) {
        # true values drop the connection
        return 1;
    }
    # false values don't drop the connection
    return 0;
}, "Older than 22 secs dropped" );
```

Drop connections by some criteria. The `$checker` argument is a subroutine that is called for each connection. If the routine returns a TRUE value then the connection is dropped. The routine is called with `($conn, $type, $key, $deposit_time)`as ar guments.

Shortcuts: If the `$checker` argument is absent (or `undef`) all cached connections are dropped. If the `$checker` is a number then all connections untouched that the given number of seconds or more are dropped. If`$checker` is a string then all connections of the gi ven type are dropped.

The `reason` is passed on to the ‘‘dropped’’ in LWP::ConnCache method.

prune

```
$cache->prune();
```

Calling this method will drop all connections that are dead. This is tested by calling the ‘‘ping’’ in LWP::ConnCache method on the connections. If the ‘‘ping’’ in LWP::ConnCache method exists and returns a false value, then the connection is dropped.

get_types

```
my @types = $cache->get_types();
```

This returns all the `type` fields used for the currently cached connections.

get_connections

```
my @conns = $cache->get_connections(); # all connections
my @conns = $cache->get_connections('http'); # connections for http
```

This returns all connection objects of the specified type. If no type is specified then all connections are returned. In scalar context the number of cached connections of the specified type is returned.

PROTOCOL METHODS

The following methods are called by low-level protocol modules to try to save away connections and to get them back.

deposit

```
$cache->deposit($type, $key, $conn);
```

This method adds a new connection to the cache. As a result, other already cached connections might be dropped. Multiple connections with the same type/key might be added.

withdraw

```
my $conn = $cache->withdraw($type, $key);
```

This method tries to fetch back a connection that was previously deposited. If no cached connection with the specified `$type`/$key is found, then `undef` is returned. There is not guarantee that a deposited connection can be withdrawn, as the cache manger is free to drop connections at any time.

INTERNAL METHODS

The following methods are called internally. Subclasses might want to override them.

enforce_limits

```
$conn->enforce_limits([$type])
```

This method is called with after a new connection is added (deposited) in the cache or capacity limits are adjusted. The default implementation drops connections until the specified capacity limits are not exceeded.

dropping

```
$conn->dropping($conn_record, $reason)
```

This method is called when a connection is dropped. The record belonging to the dropped connection is passed as the first argument and a string describing the reason for the drop is passed as the second argument. The default implementation makes some noise if the `$LWP::ConnCache::DEBUG` variable is set and nothing more.

SUBCLASSING

For specialized cache policy it makes sense to subclass `LWP::ConnCache` and perhaps override the ‘‘deposit’’ in LWP::ConnCache, ‘‘enforce_limits’’ in LWP::ConnCache, and ‘‘dropping’’ in LWP::ConnCache methods.

The object itself is a hash. Keys prefixed with `cc_` are reserved for the base class.

SEE ALSO

LWP::UserAgent

COPYRIGHT