Cache Control

Setting your headers correctly is the foundation for proper HTTP caching.

You can easily set the Cache-Control header like this:

```
get '/' do
  cache_control :public
  "cache it!"
end
```

Pro tip: Set up caching in a before filter:

```
before do
  cache_control :public, :must_revalidate, :max_age => 60
end
```

If you are using the expires helper to set the corresponding header, Cache-Control will be set automatically for you:

```
before do
  expires 500, :public, :must_revalidate
end
```

To properly use caches, you should consider using etag or last_modified. It is recommended to call those helpers *before* doing any heavy lifting, as they will immediately flush a response if the client already has the current version in its cache:

```
get "/article/:id" do
  @article = Article.find params['id']
  last_modified @article.updated_at
  etag @article.sha1
  erb :article
end
```

It is also possible to use a weak ETag:

```
etag @article.sha1, :weak
```

These helpers will not do any caching for you, but rather feed the necessary information to your cache. If you are looking for a quick reverse-proxy caching solution, try rack-cache:

```
require "rack/cache"
require "sinatra"

```

```
use Rack::Cache

get '/' do
  cache_control :public, :max_age => 36000
  sleep 5
  "hello"
end
```

Use the :static_cache_control setting (see below) to add Cache-Control header info to static files.

According to RFC 2616, your application should behave differently if the If-Match or If-None-Match header is set to *, depending on whether the resource requested is already in existence. Sinatra assumes resources for safe (like get) and idempotent (like put) requests are already in existence, whereas other resources (for instance post requests) are treated as new resources. You can change this behavior by passing in a :new_resource option:

```
get '/create' do
  etag '', :new_resource => true
  Article.create
  erb :new_article
end
```

If you still want to use a weak ETag, pass in a :kind option:

```
etag '', :new_resource => true, :kind => :weak
```