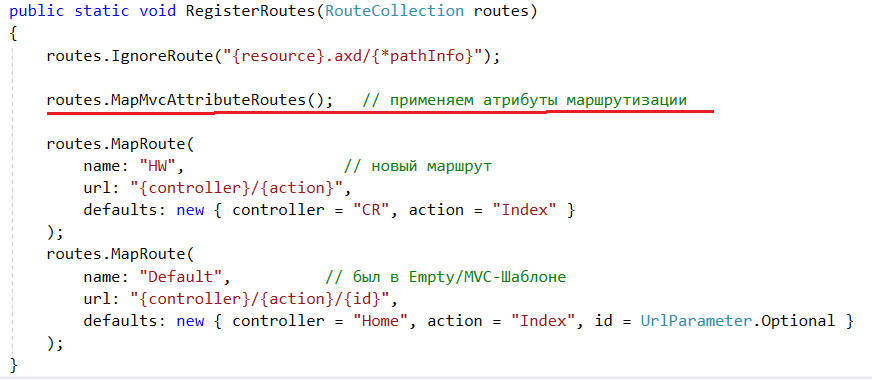
Авториза́ция — предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки данных прав при попытке выполнения этих действий.  
Аутентифика́ция — процедура проверки подлинности, например: проверка подлинности пользователя путём сравнения введённого им пароля с паролем, сохранённым в базе данных пользовательских логинов; подтверждение подлинности электронного письма путём проверки цифровой подписи письма по открытому ключу отправителя.

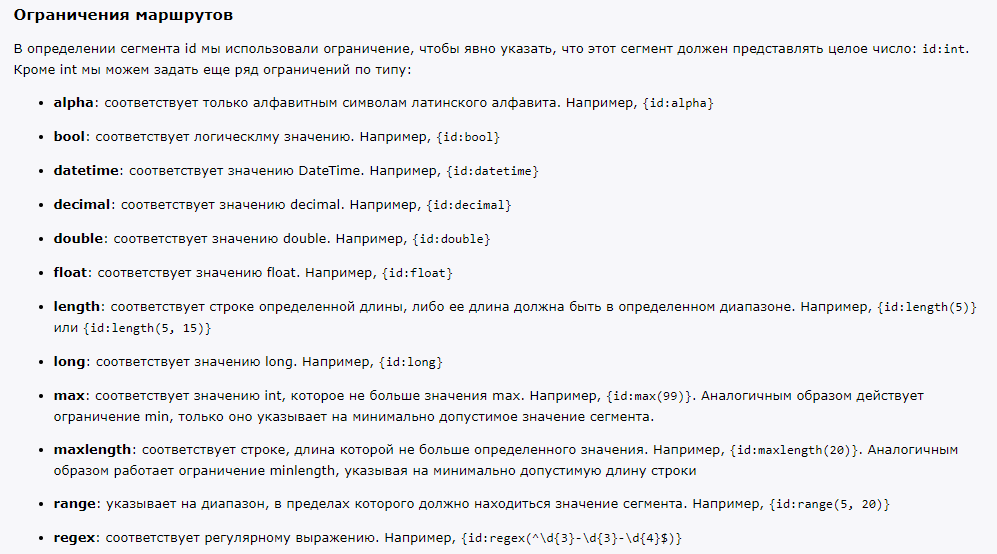
Кэш или кеш — промежуточный буфер с быстрым доступом к нему, содержащий информацию, которая может быть запрошена с наибольшей вероятностью.

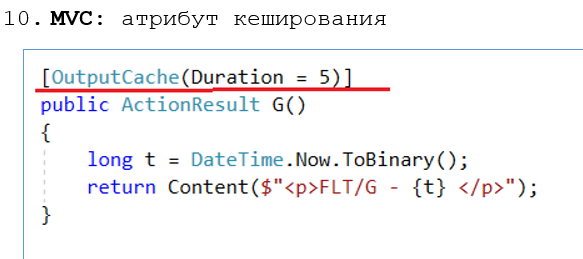
Буферизация — метод организации обмена, в частности, ввода и вывода данных в компьютерах и других вычислительных устройствах, который подразумевает использование буфера для временного хранения данных.

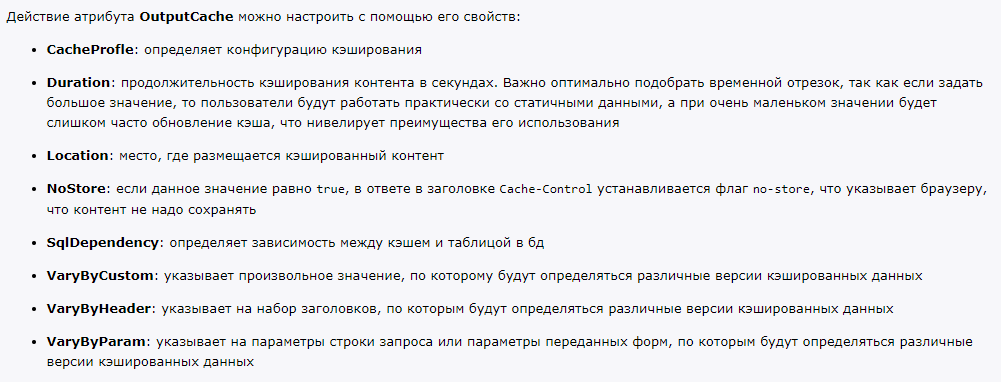
Кэш — это промежуточный буфер, содержащий информацию. Он ускоряет повторную загрузку фото, видео, приложений, программ и прочего. **Кэшированные данные** находятся в этом буфере и быстро воспроизводятся по запросу.

Основное различие между буфером и кешем заключается в том, что буферная память используется для того, чтобы справляться с различной скоростью между отправителем и получателем потока данных, тогда как кеш - это память, в которой хранятся данные, так что скорость доступа может быть увеличена для многократно используемых данных.  
Буфер всегда несет исходные данные для отправки получателю. Тем не менее, кеш несет копию оригинальных данных.  
Буфер всегда реализуется в основной памяти, но кэш может быть реализован как в RAM, так и в Disk.









Атрибут [ActionName] **позволяет для метода задать другое имя действия**

1. **Cache**: предназначен для хранения кэшируемых данных в форме имя/значение, но можно для каждого элемента хранения определить специализированные политики истечения срока и зависимости.
2. **Кэширование:** техника хранения в памяти копий некоторой информации для повторного применения.
3. **Кэширование:** два вида: **кэширование вывода** и **кэширование данных**.
4. **Кэширование вывода**: можно сохранить копию сгенерированной страницы, отправленной клиенту; следующий запрос на эту же страницу не потребует ее генерации – используется страница из кэша.
5. **Кэширование данных:** программист может сохранить в кэше данны; страницы могут проверять наличие данных в кэше и использовать ее.

**кэшировать надо по заголовкам запроса, а не ответа**

1. Кэширование части страницы: **фрагментарное кэширование** (используется для пользовательских элементов управления), **послекэшевая постановка** (не может быть использована для пользовательских элементов управления.
2. **Если в объект Session ничего не записано, то при новом запросе создается новая сессия**
3. Для кэширования данных применяется объект **Cache**.

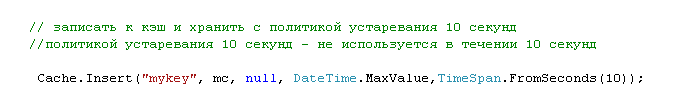
* Cache **безопасный для потоков**;
* элементы Cache удаляются **автоматически**;
* элементы Cache поддерживают **зависимости**;
* Cache **привязан к домену** приложения (после перезапуска домена – новый кэш);
* Cache **не разделяется в серверном кластере**.

- политика скользящего устаревания;

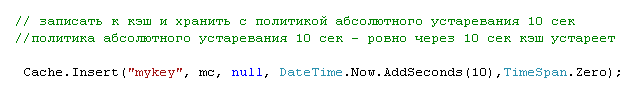
- политика абсолютного устаревания.

**Модель устаревания HTTP кэша** была разработана для того, чтобы клиенты не делали постоянные запросы к изначальному серверу, а получали актуальные HTTP ответы от промежуточных узлов.

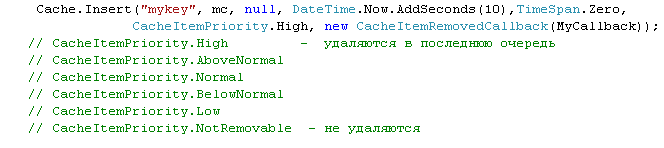
1. **Политика скользящего устаревания:** задается время неиспользования элемента хранения



1. **Политика абсолютного устаревания**

****

1. **Приоритеты кэша и процедура обратного вызова**

****

1. **Зависимости кэша**:

1) от других кэшированных элементов;

2) от файлов или папок;

3) от запросов к базе данных.

1. **Пользовательская зависимость кэша**

- с помощью таймера;

- с помощью отдельного потока;

- с помощью обработчика других событий.

Существует несколько видов кэширования:

##### 1. Браузерное кэширование или клиентское кэширование

Представляет собой составление для браузера команды использовать имеющуюся кэшированную копию. Работа такого кэширования основана на том, что при повторном посещении, браузеру отдаётся заголовок 304 Not Modified, а сама страница или картинка загружаются из локального польз-ого кэша.

Веб-страницы состоят из множества различных элементов: картинок, css и js файлов и т.п. Часть этих элементов используются на нескольких страницах сайта. Под клиентским кэшированием понимают способность браузеров сохранять копии файлов (ответов сервера), чтобы не загружать их повторно.

Идея заключается в том, что сервер добавляет заголовок Last-modified к ответу, кот. он отдает браузеру.

Теперь браузер знает, что файл был создан (или изменен). В следующий раз, когда браузеру понадобится тот же файл, он отправит запрос с заголовком if-Modified-Since.

Если файл не изменялся, сервер отправляет браузеру пустой ответ со статусом 304 (Not Modified). Тогда, браузер знает, что файл не обновлялся и мб отобразить копию, которую он сохранил в прошлый раз.

https://html5.by/blog/cache/#:~:text=Под%20клиентским%20кэшированием%20понимают%20способность,также%20снизить%20нагрузку%20на%20сервер.

**1.1 Кэширование файлов и картинок**

**1.2 Кэширование https**

**1.3 Кэширование центра сертификации**

**1.4 Кэширование страниц**

##### 2. Серверное кэширование

Под серверным кэшированием понимаются все виды кэширования, при котором данные хранятся на серверной стороне (DataSet). Эти данные не доступны клиентским браузерам. Кэш создаётся и хранится по принципу «один ко многим» (многие, в данном случае, — это клиентские устройства).

Суть кэширования на стороне сервера состоит в том, чтобы **записать все, что происходит на сервере, в файл, сохранить его и при последующем обращении этого, либо любого другого пользователя к этой странице просто выдать ему статичную копию**.

**2.1 Кэширование страницы целиком**

**2.2 Кэширование результатов компиляции php-файлов**

**2.3 Кэширование отдельных блоков страницы**

**2.4 Кэширование php на основе неразделяемых ресурсов**

**2.5 Кэширование php на основе общих ресурсов**

**2.6 Кэширование mysql на основе query cache**

**2.7 Кэширование mysql результатов работы, агрегирующие таблицы**

<https://habr.com/ru/company/zerotech/blog/316316/>

Существует несколько видов кешей, разеленные на две основные категории: приватные кеши и кеши совместного использования. В кешах совместного использования хранятся копии, которые могут направляться разным пользователям. Приватный кеш предназначен для отдельного пользователя

# Четыре уровня кэширования в сети: клиентский, сетевой, серверный и уровень приложения