```
[ApiController]
[ResponseCache(CacheProfileName = "120SecondsDuration")]
```

We have to mention that this cache rule will apply to all the actions inside the controller except the ones that already have the ResponseCache atribute applied.

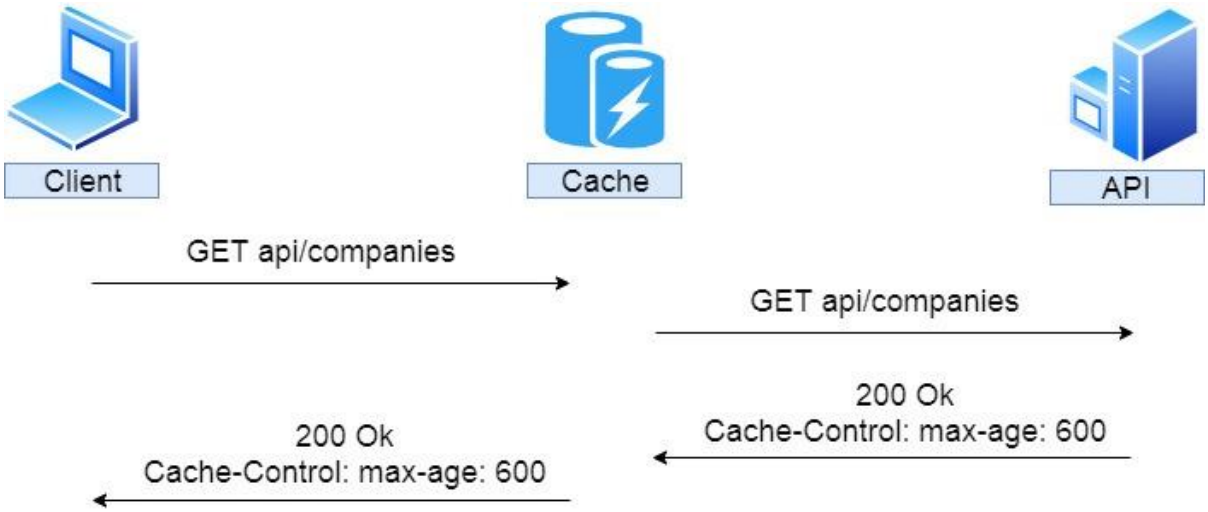
That said, once we send the request to `GetCompany`, we will still have the maximum age of 60. But once we send the request to `GetCompanies`:

There you go. Now, let's talk some more about the Expiration and Validation models.

Expiration Model

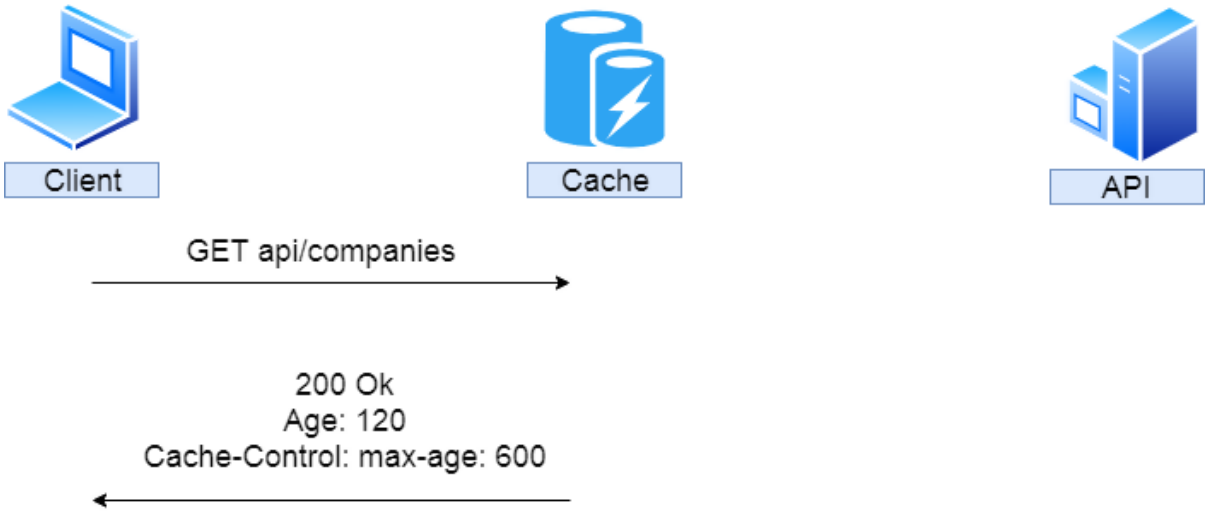
The expiration model allows the server to recognize whether or not the response has expired. As long as the response is fresh, it will be served from the cache. To achieve that, the Cache-Control header is used. We have seen this in the previous example.

Let's look at the diagram to see how caching works:



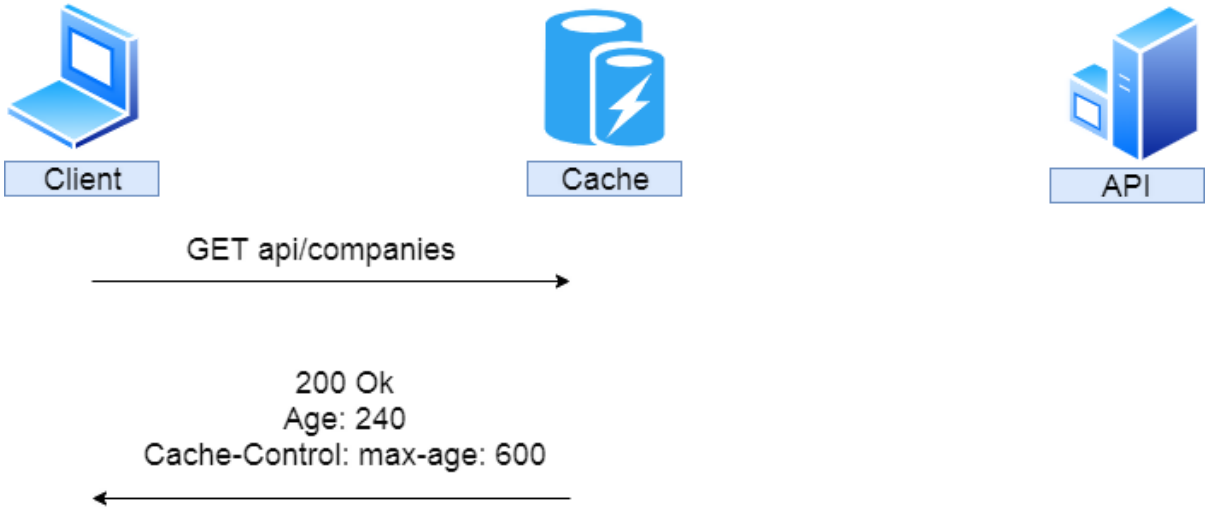
So, the client sends a request to get companies. There is no cached version of that response; therefore, the request is forwarded to the API. The API returns the response with the Cache-Control header with a 10-minute expiration period; it is being stored in the cache and forwarded to the client.

If after two minutes, the same response has been requested:



We can see that the cached response was served with an additional Age header with 120 seconds or two minutes. If this is a private cache, that is where it stops. That's because the private cache is stored in the browser and another client will hit the API for the same response. But if this is a

shared cache and another client requests the same response after an additional two minutes:



The response is served from the cache with an additional two minutes added to the Age header.

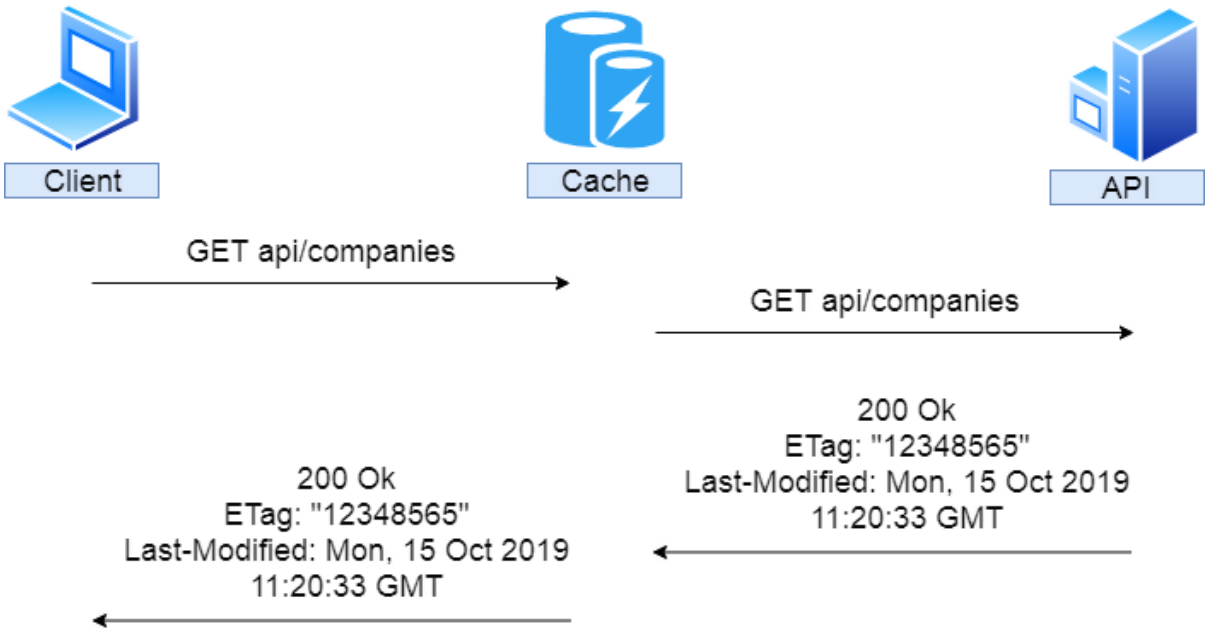
We saw how the Expiration model works, now let's inspect the Validation model.

Validation Model

The validation model is used to validate the freshness of the response. So it checks if the response is cached and still usable. Let's assume we have a shared cached GetCompany response for 30 minutes. If someone updates that company after five minutes, without validation the client would receive the wrong response for another 25 minutes — not the updated one.

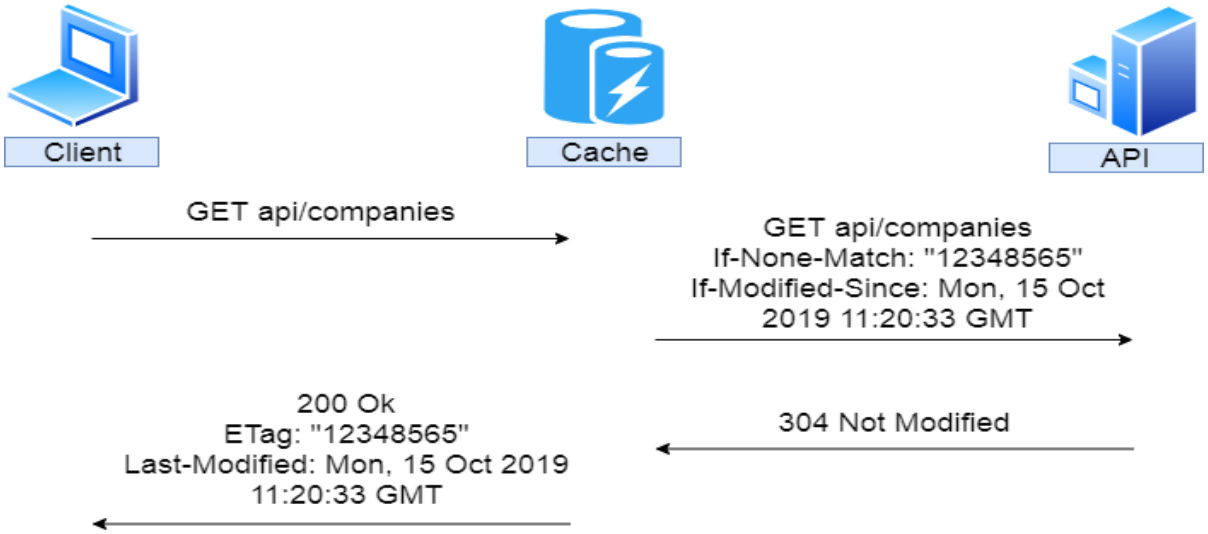
To prevent that, we use validators. The HTTP standard advises using Last-Modified and ETag validators in combination if possible.

Let's see how validation works:



So again, the client sends a request, it is not cached, and so it is forwarded to the API. Our API returns the response that contains the Etag and Last-Modified headers. That response is cached and forwarded to the client.

After two minutes, the client sends the same request:



So, the same request is sent, but we don't know if the response is valid. Therefore, the cache forwards that request to the API with the additional headers If-None-Match — which is set to the Etag value — and If-