Caching Outside the Solution

Elton Stoneman
elton@sixeyed.com

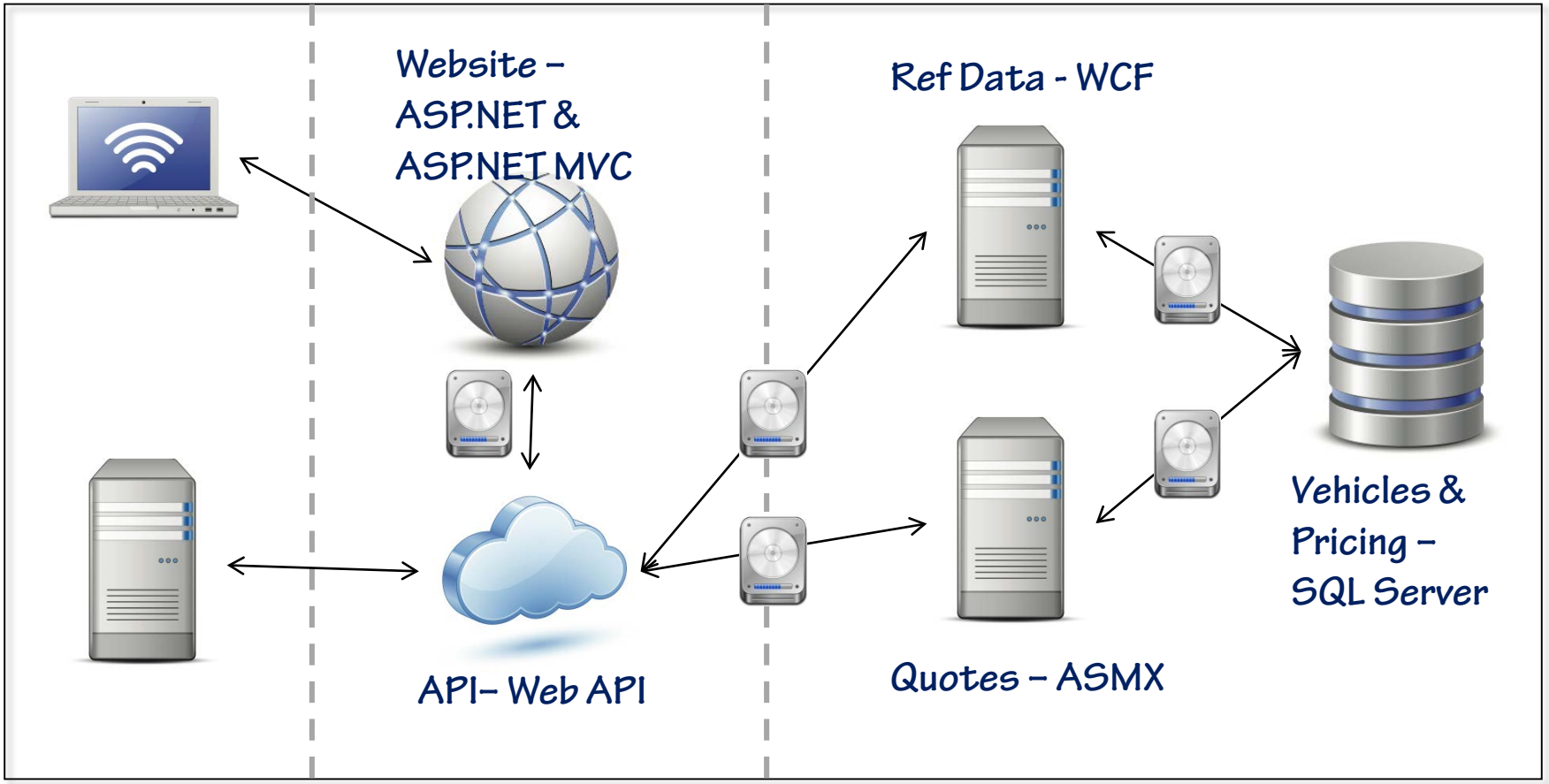
pluralsight
hardcore developer training

Outline

- **Caching inside-out**
  - Browser and proxy caches
- **Caching in HTTP**
  - Transactions and status codes
- **Validation**
  - Etag, Last-Modified and conditional GETs
- **Expiration**
  - Expires and Cache-Control
- **IIS, ASP.NET, Web API & WCF**
  - HTTP – web resources & REST, not SOAP

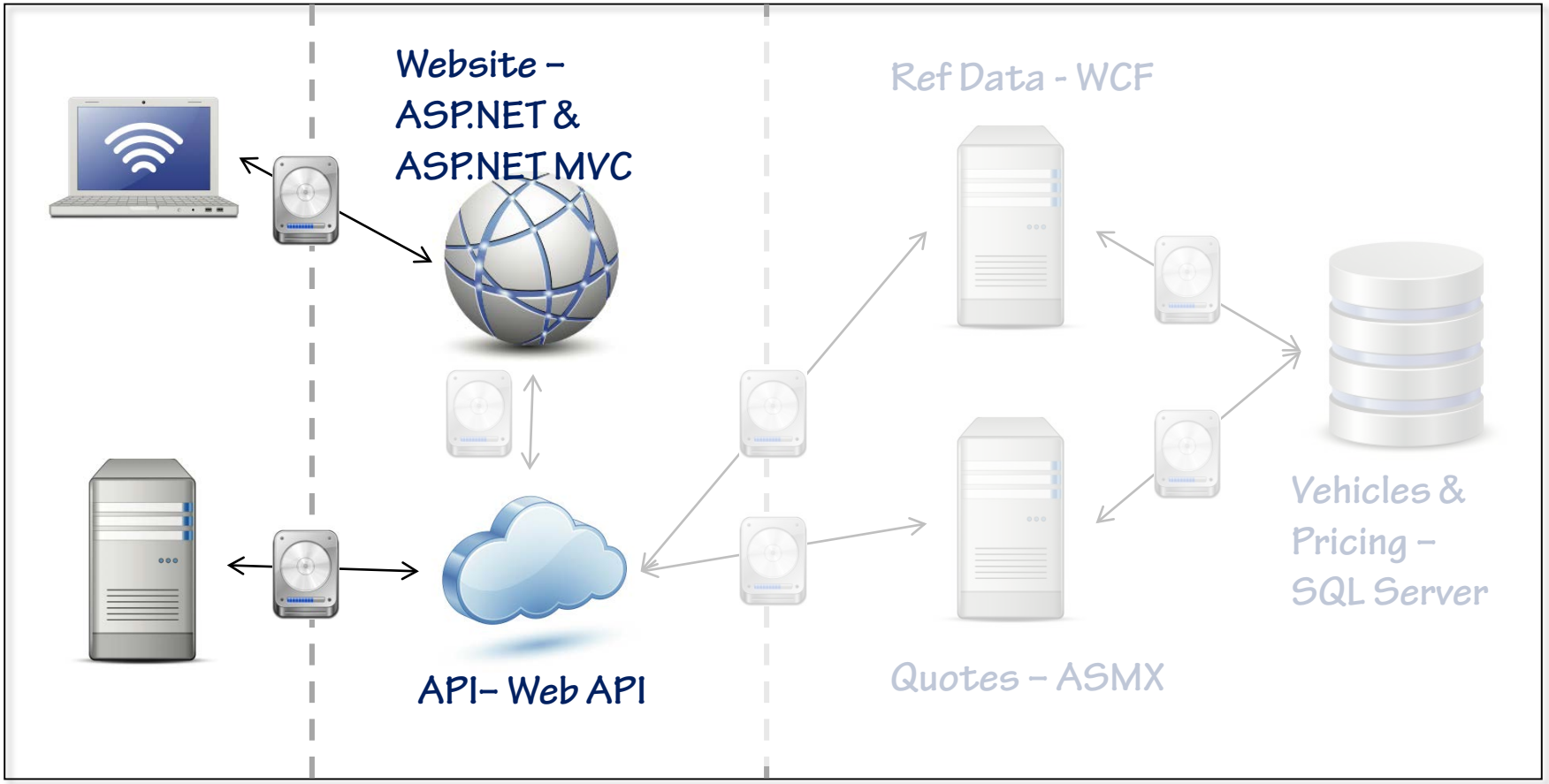
Inside-Out

- Caching inside the solution



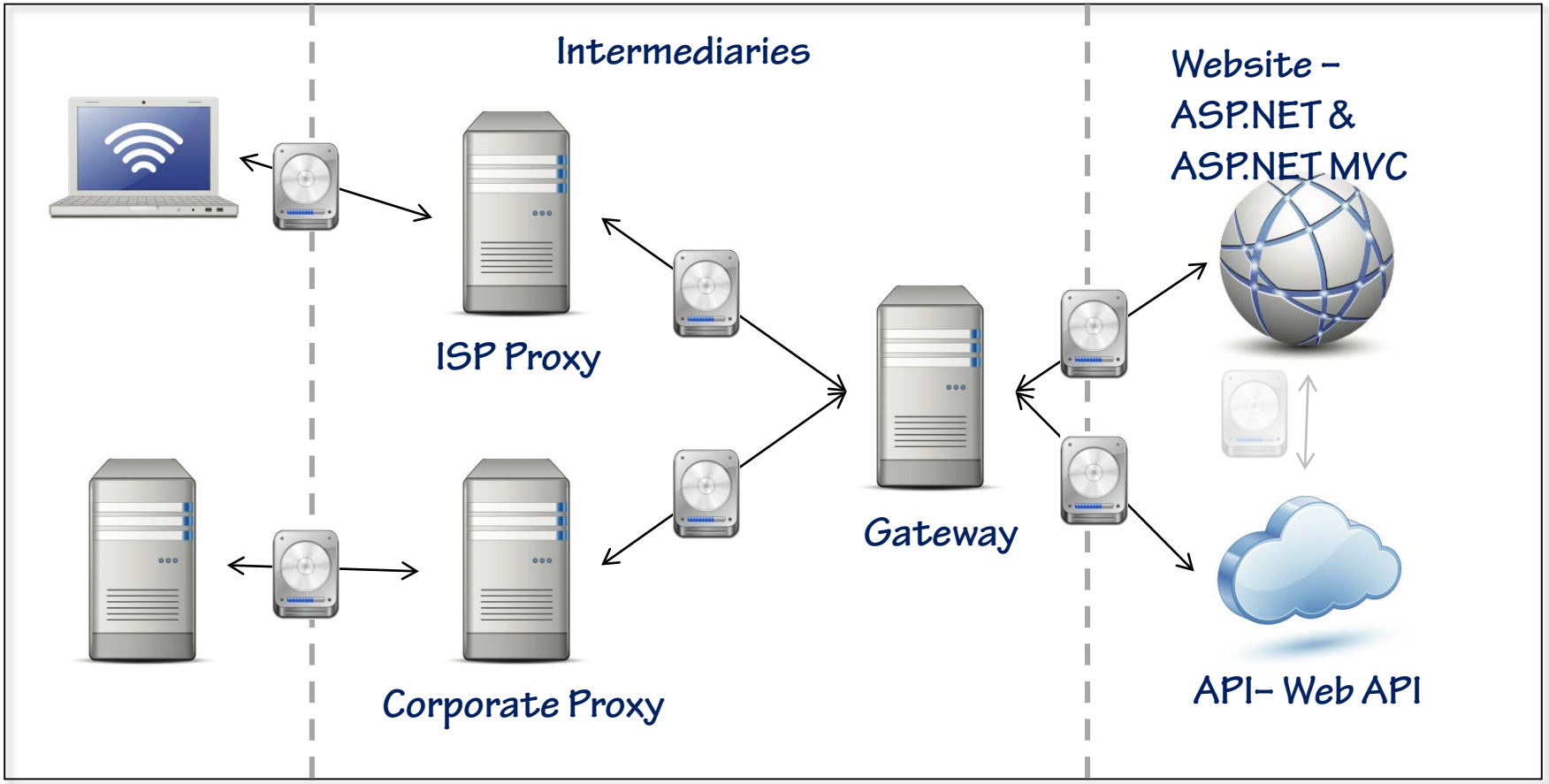
Inside-Out

- Caching outside the solution



Inside-Out

- Client and proxy caches



Caching in HTTP

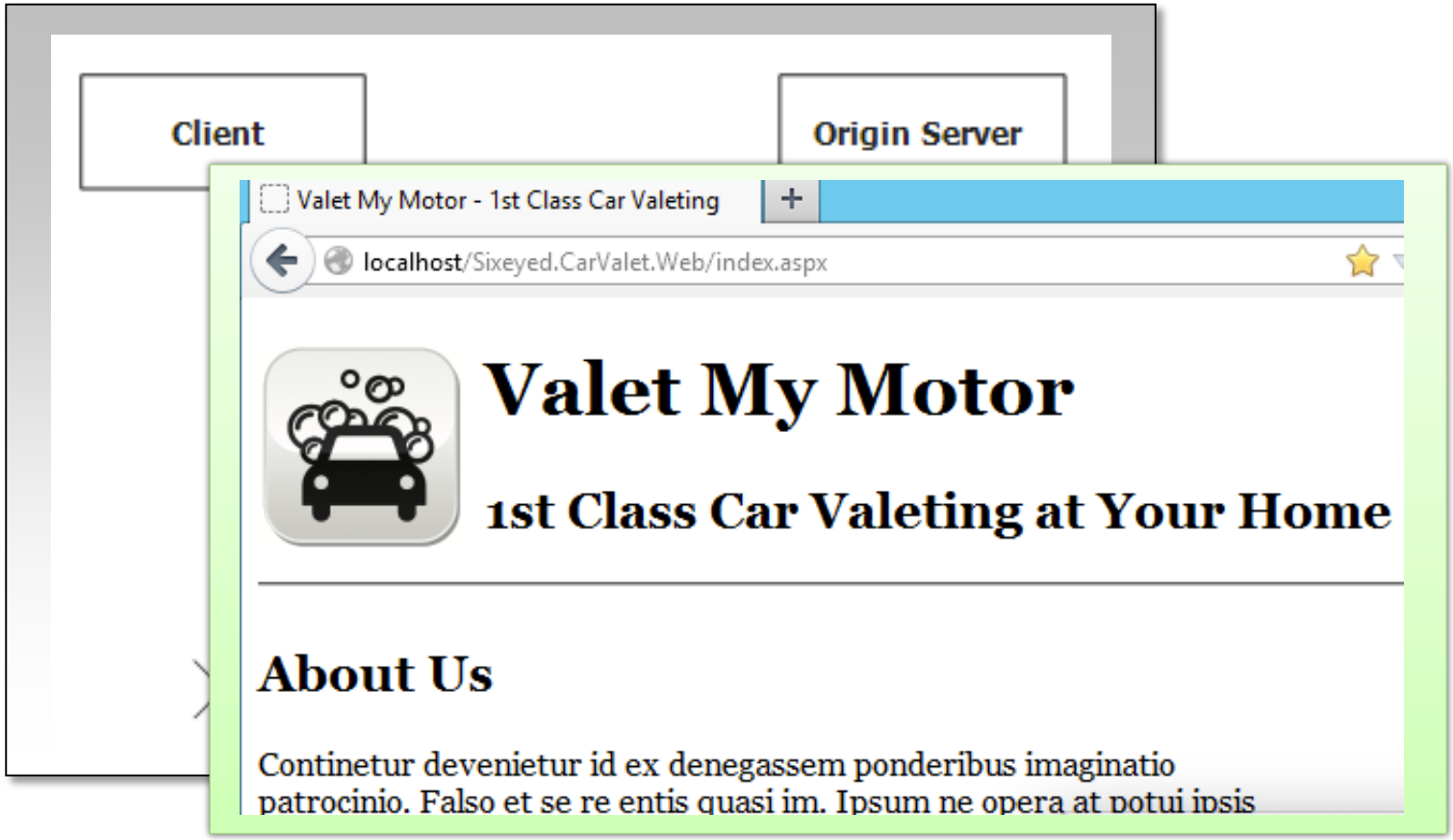
> “The goal of caching in HTTP/1.1 is
> to eliminate the need to send requests in many cases,
> and to eliminate the need to send full responses in many other cases”

Caching in HTTP

- **Request**
  - URL
  - Method
    - GET, POST (+ HEAD, PUT, DELETE etc.)
  - Headers
    - Accept: application/json
  - {Body}
- **Response**
  - {Body}
  - Headers
    - Status: 200 OK
    - Content-Type: application/json

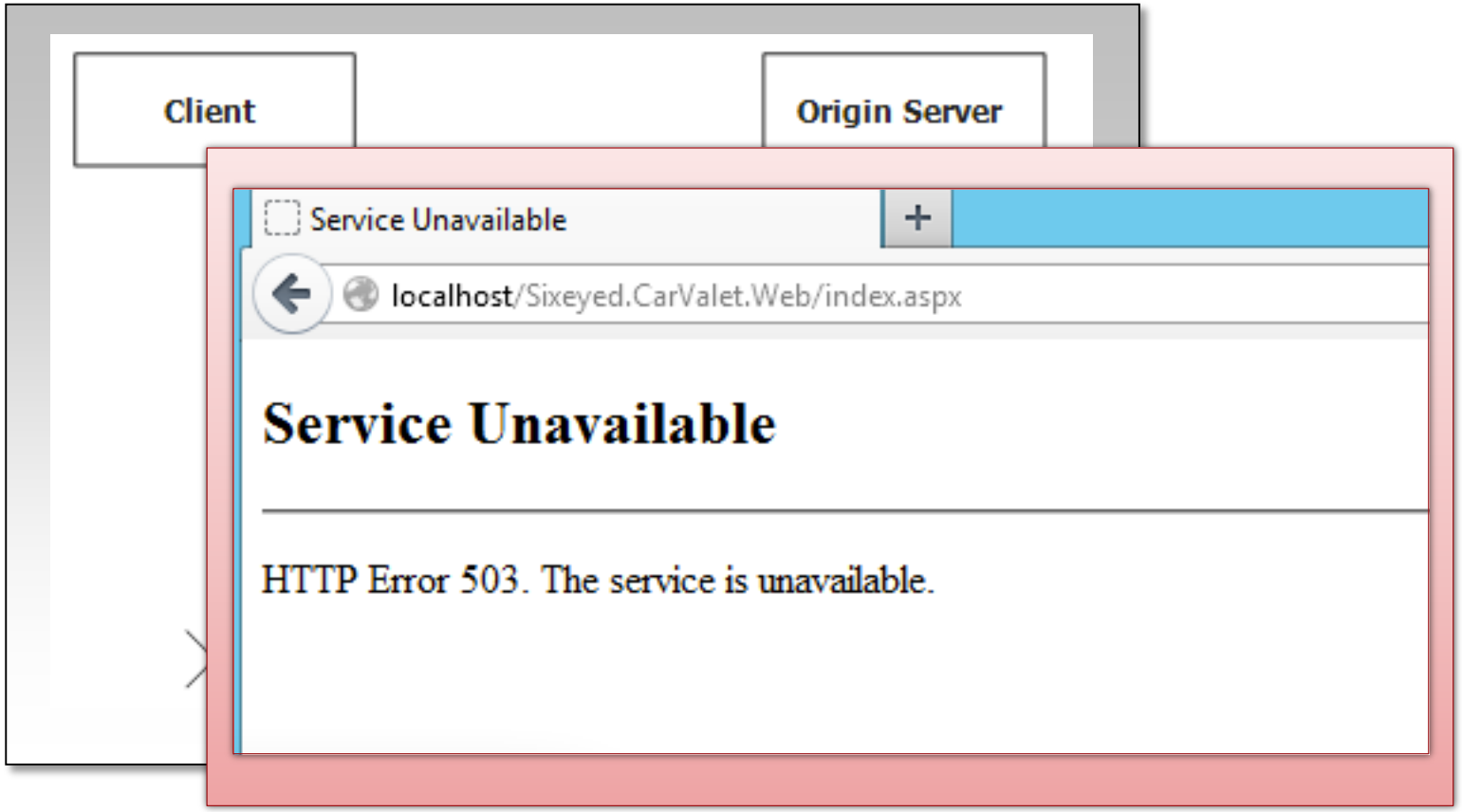
Caching in HTTP

- Simple GET with success response



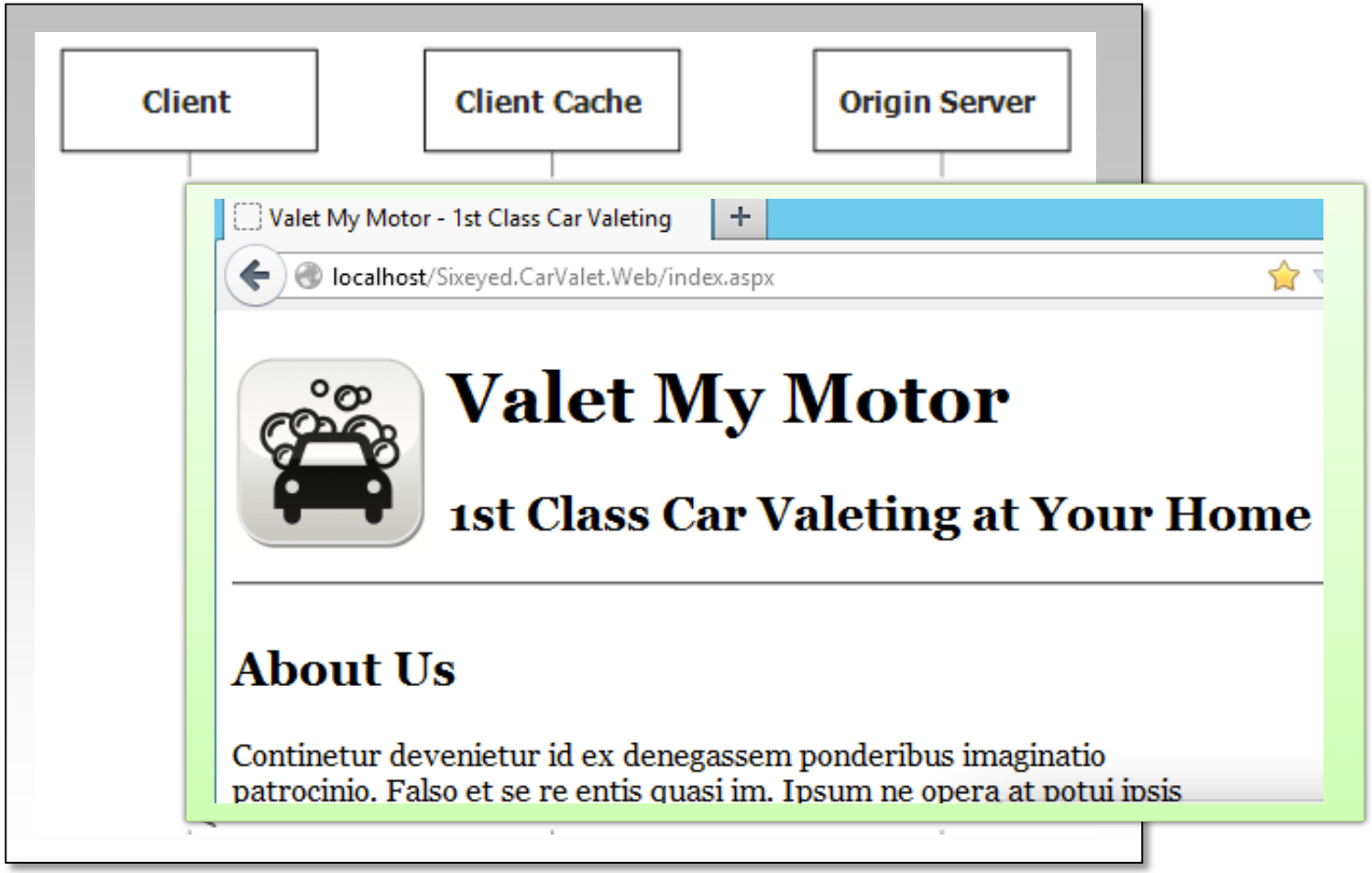
Caching in HTTP

- Simple GET with failure response



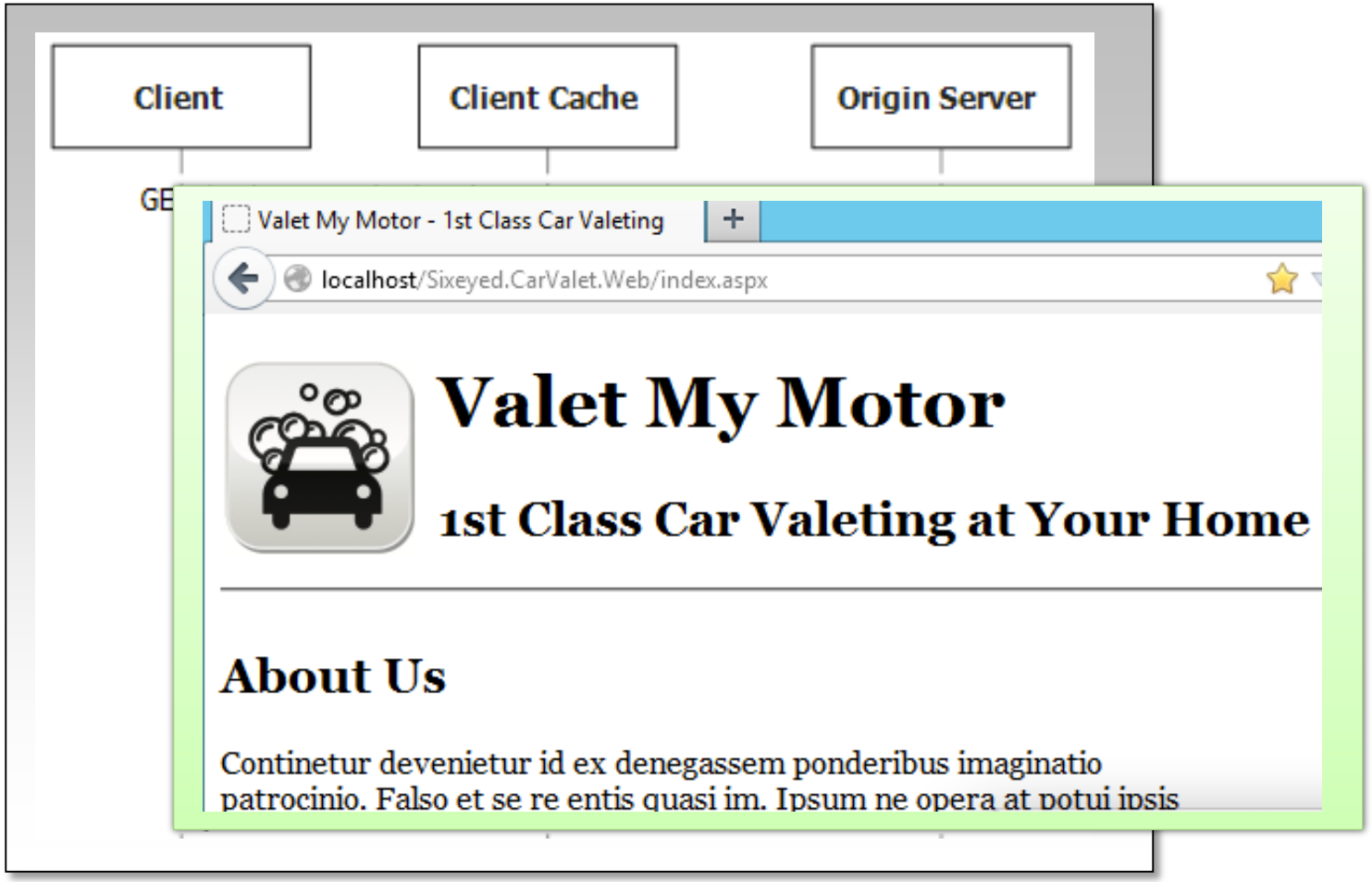
Caching in HTTP

- Simple GET with success response + cache header



Caching in HTTP

- Repeat GET + cache-header with not-modified response



Validation

- **Attribute check**
  - Origin server returns validation metadata
  - Client caches response
    - Validates repeat request with origin server
- **Etag**
  - Resource state
  - MD5 hash of body
  - Client sends if-none-match
- **Last-Modified**
  - Change timestamp
  - Client sends if-modified-since
- **Etag takes precedence**

Validation

- Demo

Validation

- **IIS – static content**
  - Last-Modified
    - Modified timestamp from filesystem
    - Sun, 21 Apr 2013 18:51:56 GMT
  - Etag
    - Computed – consistent algorithm
    - "43ab214cc13ece1:0"
- **ASP.NET – WebForms and MVC**
  - Last-Modified (and Expiration)

```
<%@ OutputCache Duration="30"
                Location="ServerAndClient"
                VaryByParam="none"  %>
```

```
[OutputCache(Duration=30,
             Location=OutputCacheLocation.ServerAndClient)]
```

Validation

- **WebAPI**
  - Custom ActionFilter - generate Etag from hashed response

```
message.Headers.ETag = new EntityTagHeaderValue(eTag, false);
```

```
[ComputeETag]
public IEnumerable<string> GetByName(string startsWith)
```

- **WCF**
  - Explicitly add & check in service operation

```
var response = WebOperationContext.Current.OutgoingResponse;
if (response != null)
{
    response.SetETag(eTag);
```

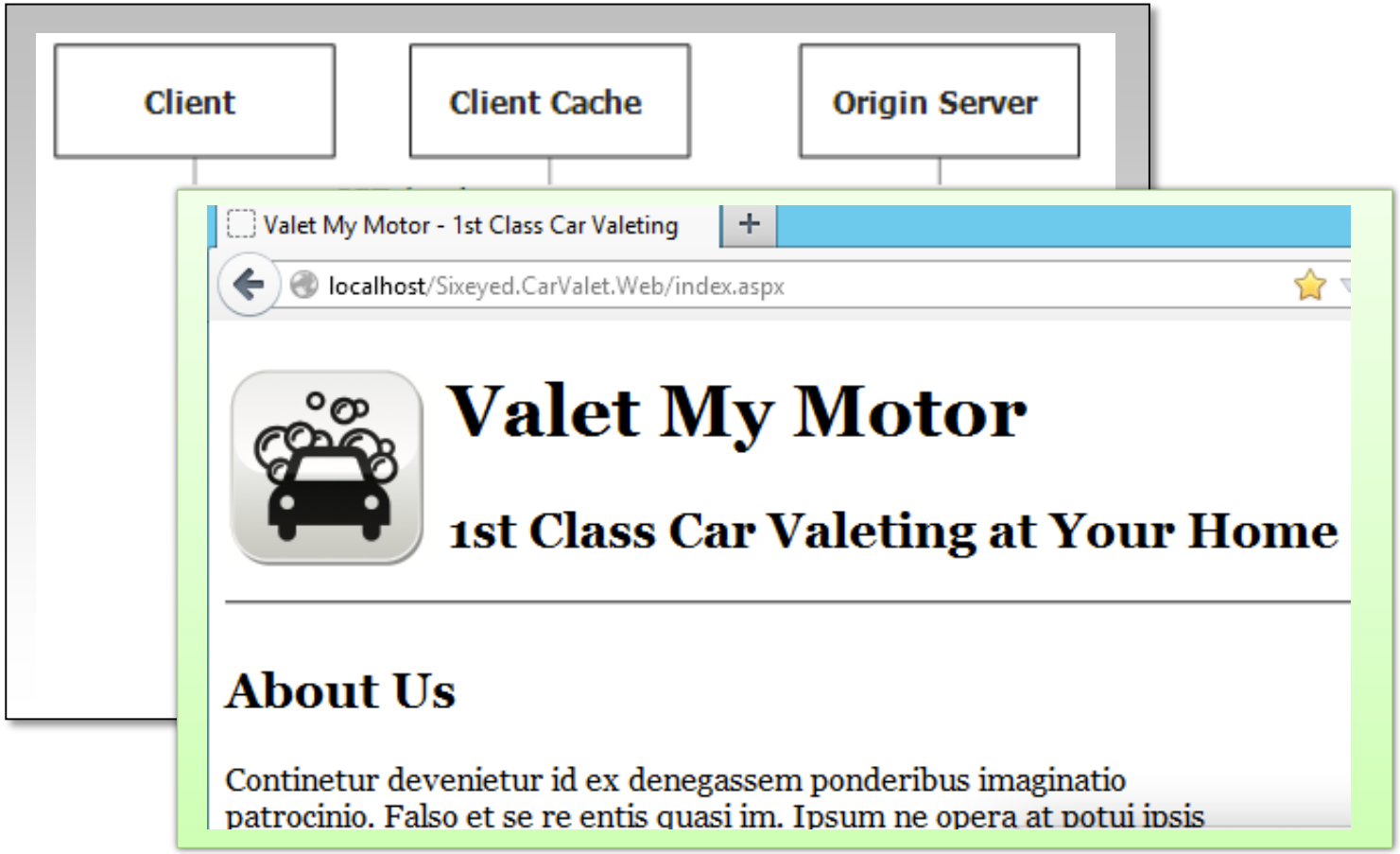
```
var request = WebOperationContext.Current.IncomingRequest;
if (request.Method == "GET" || request.Method == "HEAD")
{
    request.CheckConditionalRetrieve(eTag);
```

Expiration

- Simple GET with expires response

Expiration

- Repeat GET with expires response



Expiration

- **Lifespan**
  - Origin server returns expiration metadata
  - Client caches response
    - Uses cached response until expired
    - No repeat call to origin server
- **Expires (HTTP 1.0)**
  - Stated, fixed lifespan
- **Cache-control (HTTP 1.1)**
  - Computed, fixed lifespan
  - Highly configurable
- **Expiration and validation work together**
  - Expiration takes precedence
  - Validation may become expiration

Expiration

- Demo

Expiration

- **IIS – static content**
  - Expires or Cache-Control

```
<staticContent>
  <clientCache httpExpires="Sun, 29 Mar 2020 00:00:00 GMT"
               cacheControlMode="UseExpires" />
</staticContent>
```

- **ASP.NET- WebForms and MVC**
  - Output cache
  - Expires and Cache-Control (and Validation)

```
<%@ OutputCache Duration="30"
                Location="ServerAndClient"
                VaryByParam="none"  %>
```

```
[OutputCache(Duration=30,
             Location=OutputCacheLocation.ServerAndClient)]
```

Expiration

- **WebAPI**
  - Custom ActionFilter – specify lifespan

```
response.Content.Headers.Expires = DateTimeOffset.Now +
                                        TimeSpan.FromSeconds(Seconds);
```

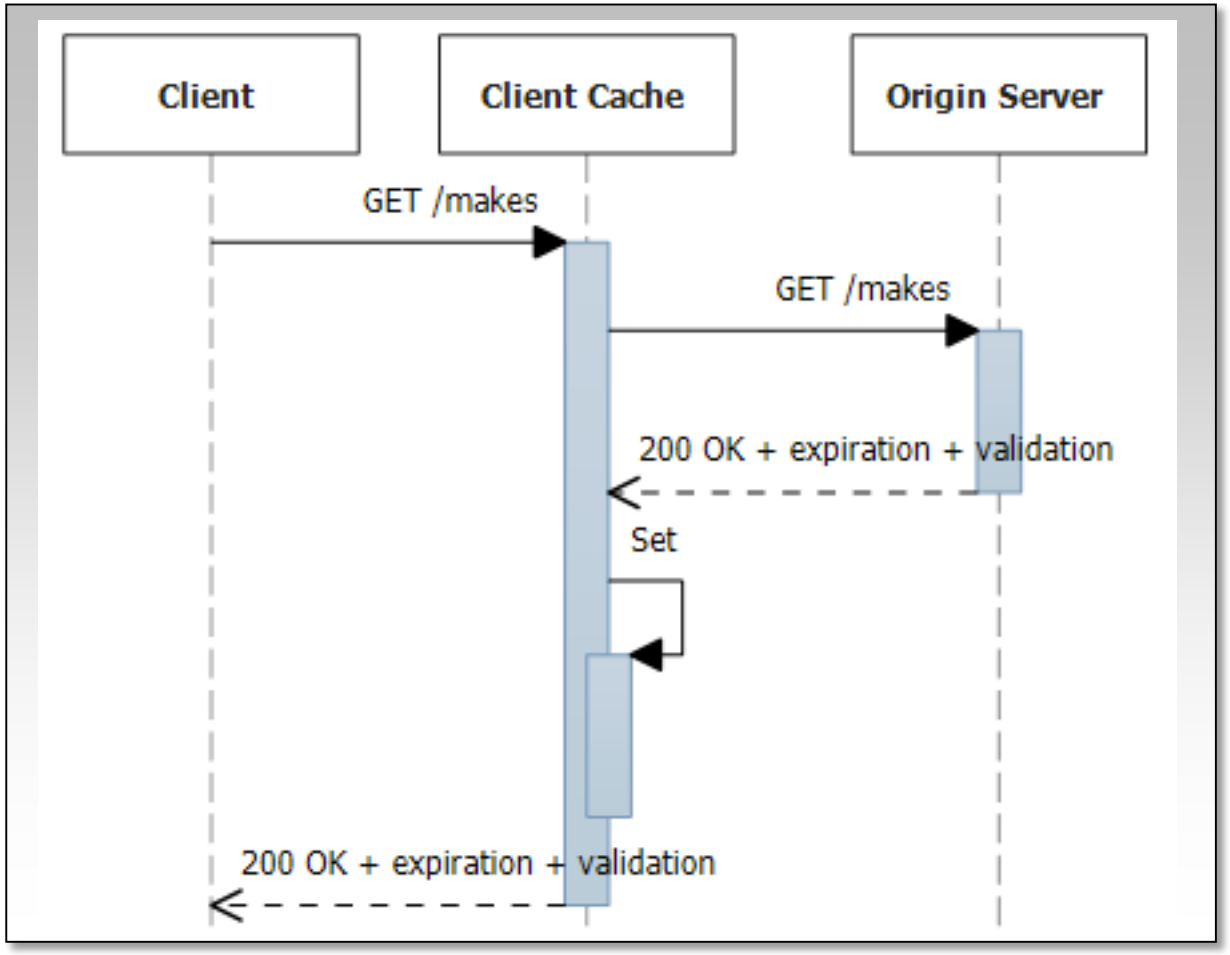
```
[Expires(Seconds = 30)]
public IEnumerable<string> GetByName(string startsWith)
```

- **WCF**
  - Explicitly added in service operation

```
var response = WebOperationContext.Current.OutgoingResponse;
if (response != null)
{
    var expires = DateTime.UtcNow.AddSeconds(seconds).ToExpiresString();
    response.Headers.Add(HttpResponseHeader.Expires, expires);
```

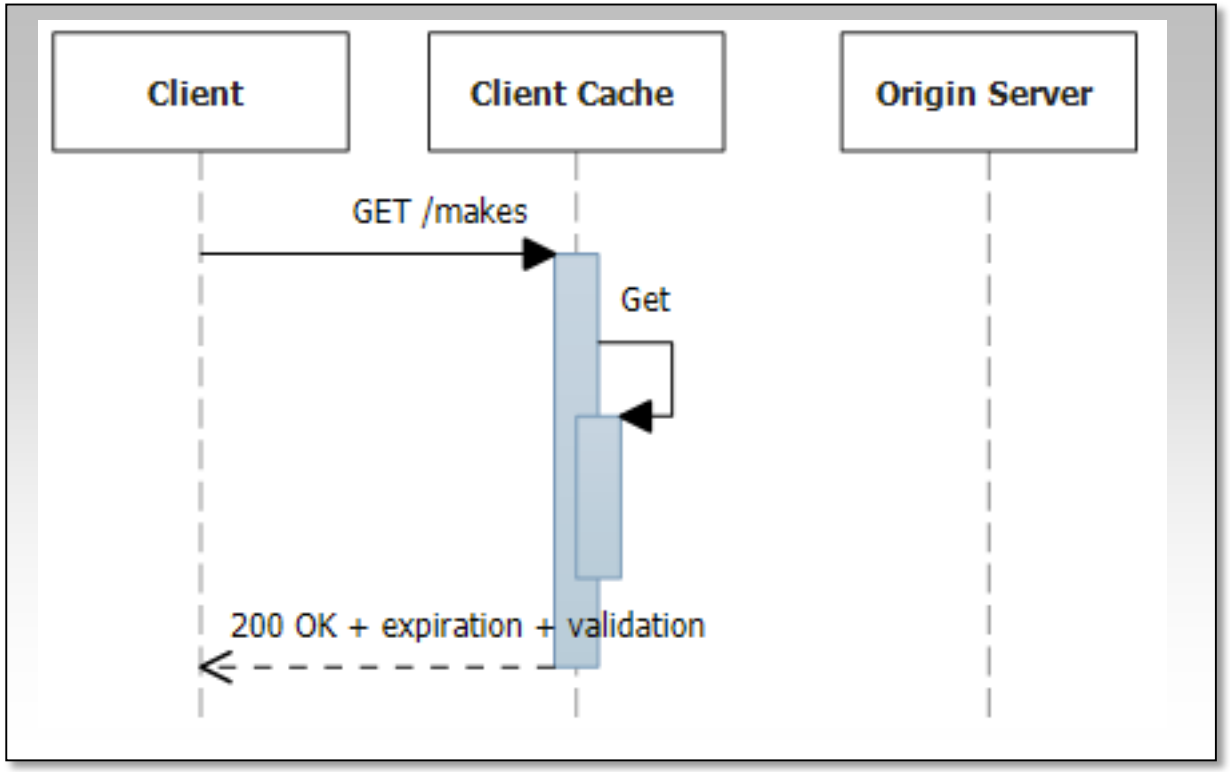
Expiration and Validation

- First GET: Expires/Cache-Control + Etag/Last-Modified



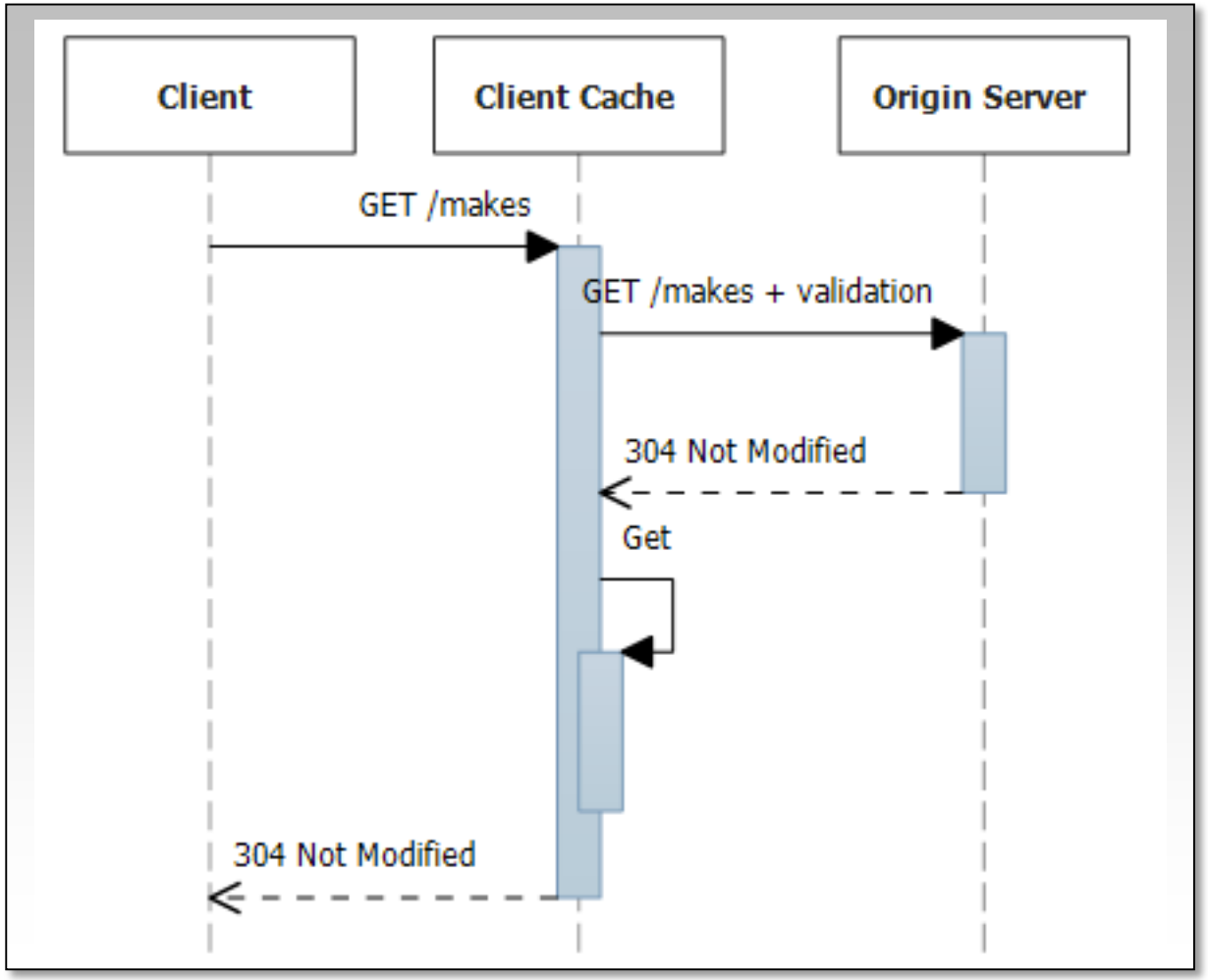
Expiration and Validation

- Subsequent GET (within expiration)



Expiration and Validation

- Subsequent GET (expired but unmodified)



Summary

- **Client caching in HTTP**
  - HTTP protocol
- **IIS, ASP.NET, Web API & WCF**
- **Validation**
  - Etag, Last-Modified
  - Conditional GETs – headers only
- **Expiration**
  - Expires and Cache-Control
  - No GETs
- **Expiration and Validation**