Caching Inside the Solution

Elton Stoneman
elton@sixeyed.com

pluralsight
hardcore developer training

Outline

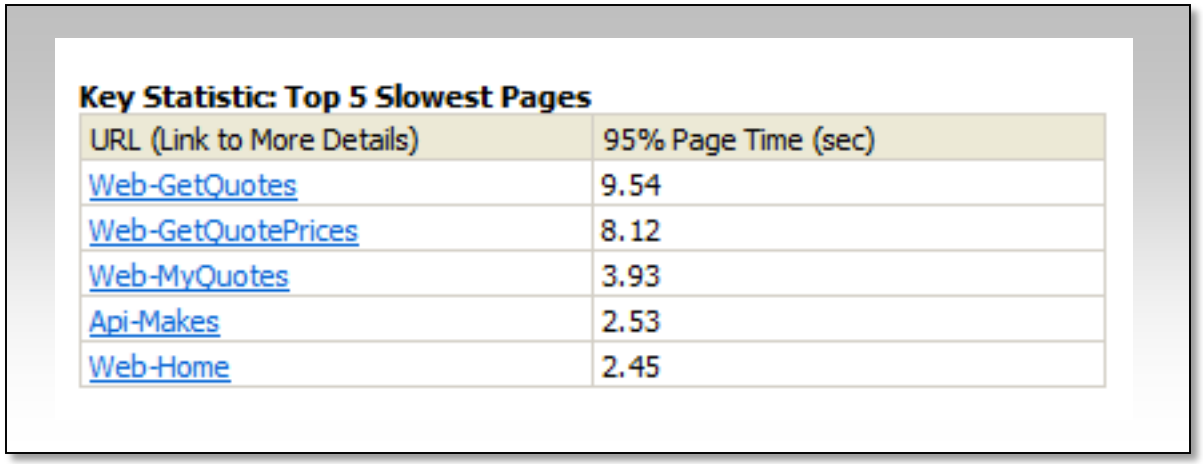
- **A tour of the demo solution**
  - Web & API
- **Caching method results explicitly**
  - Get/check/set pattern
- **Caching method results with AOP**
  - Deep-dive into the CacheCallHandler
  - An alternative AOP pattern
- **A decision matrix for cache items**
  - Guidelines for choosing the right technique and technology

A Tour of the Demo Solution

- Demo

A Tour of the Demo Solution

- **Repeat calls don't reduce latency**
- **Distributed architecture**
  - No caching - blocking calls between components
- **Acceptable performance**
  - For a single user
  - For 60 concurrent users, performance nose-dives

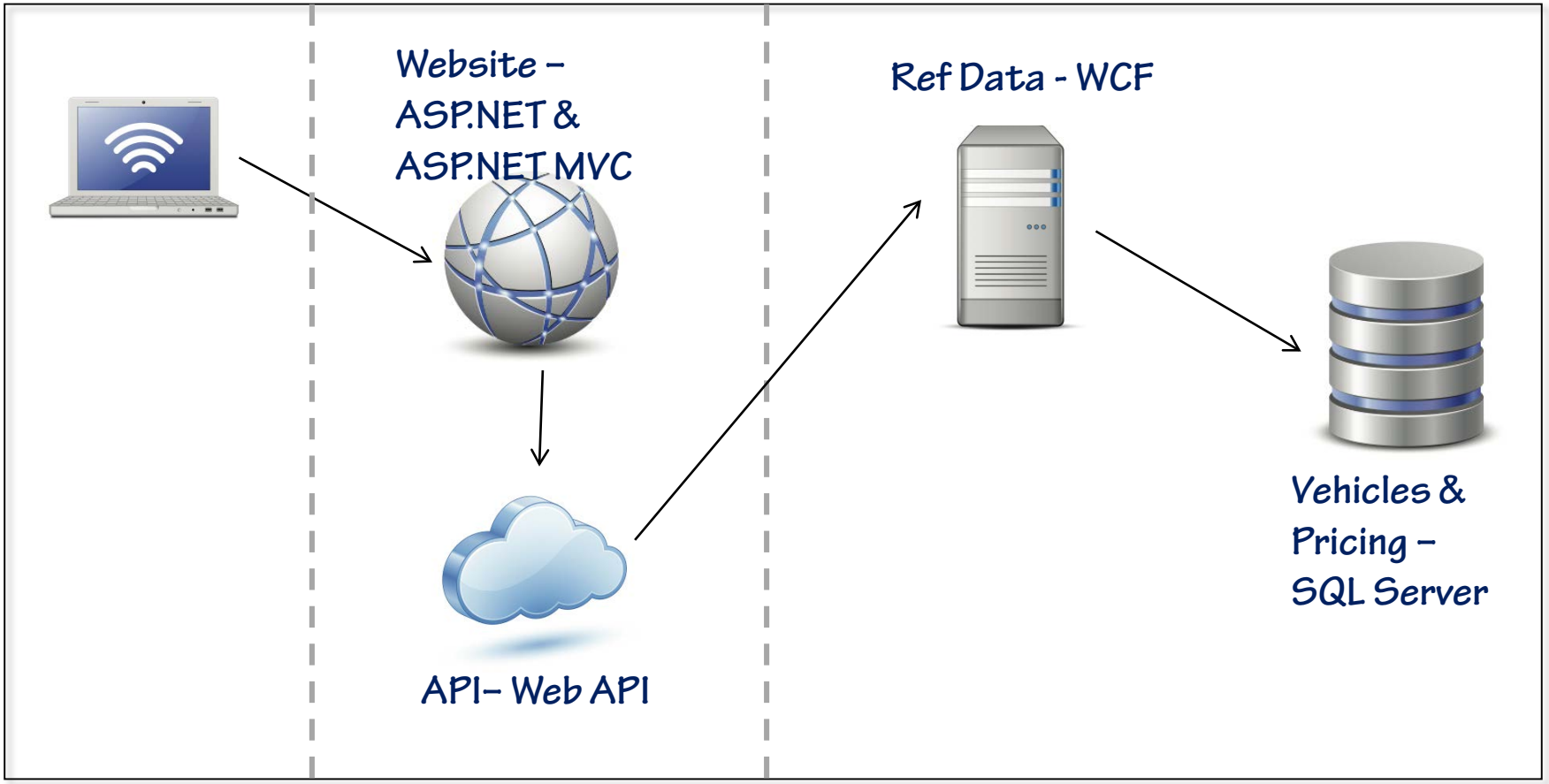


Key Statistic: Top 5 Slowest Pages

URL (Link to More Details)	95% Page Time (sec)
Web-GetQuotes	9.54
Web-GetQuotePrices	8.12
Web-MyQuotes	3.93
Api-Makes	2.53
Web-Home	2.45

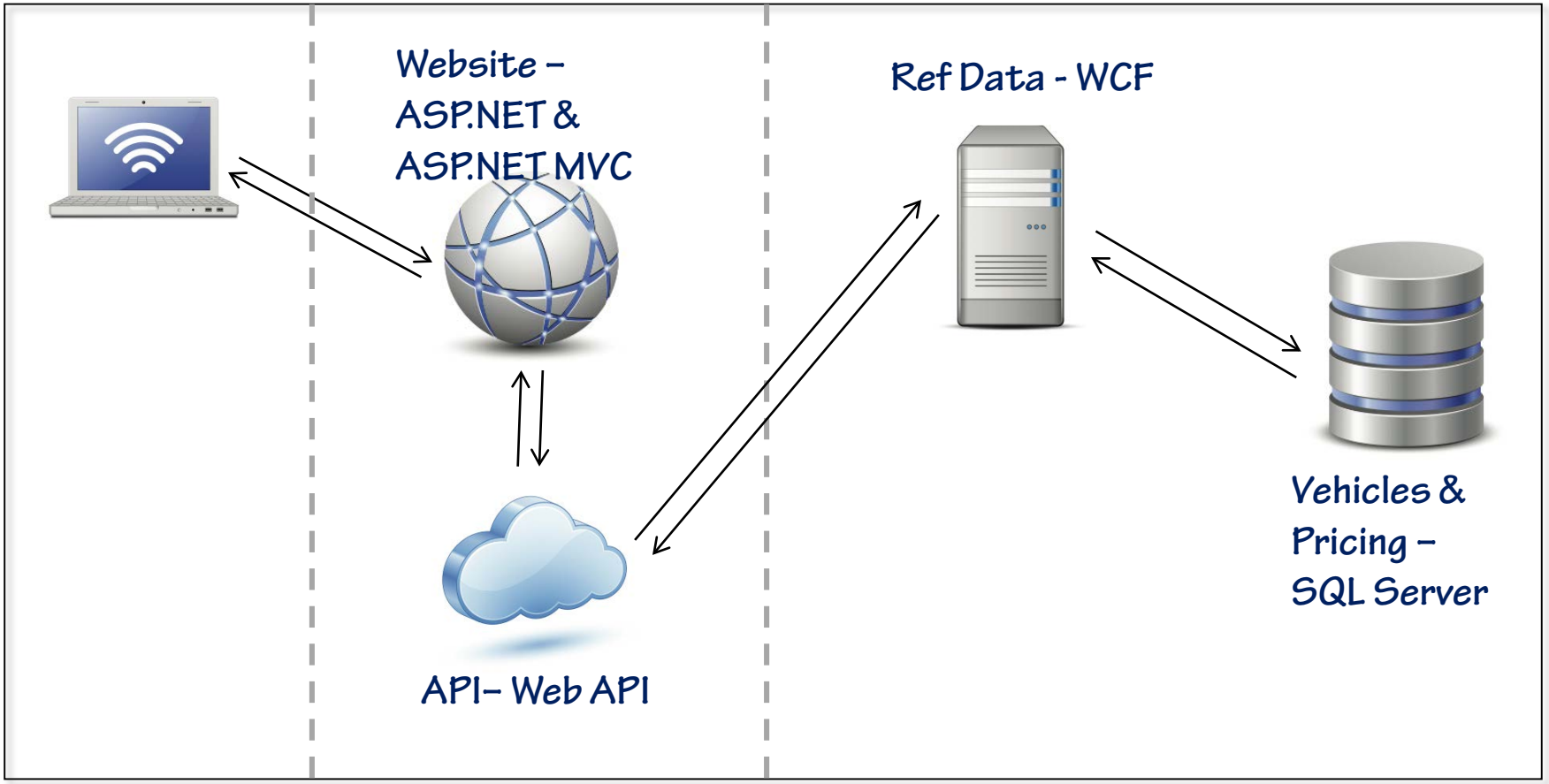
A Tour of the Demo Solution

- Caching to reduce contention & improve scalability



A Tour of the Demo Solution

- Caching to reduce contention & improve scalability



Caching Method Results Explicitly

- Demo

Caching Method Results Explicitly

- **Simple get/check/set pattern**
- **Details in the explicit implementation**
  - Cache key
  - Cache type
  - Lifespan
- **Hard-coded choices**

Caching Method Results with AOP

- **The .Core library**
  - Log – wrapper over log4net

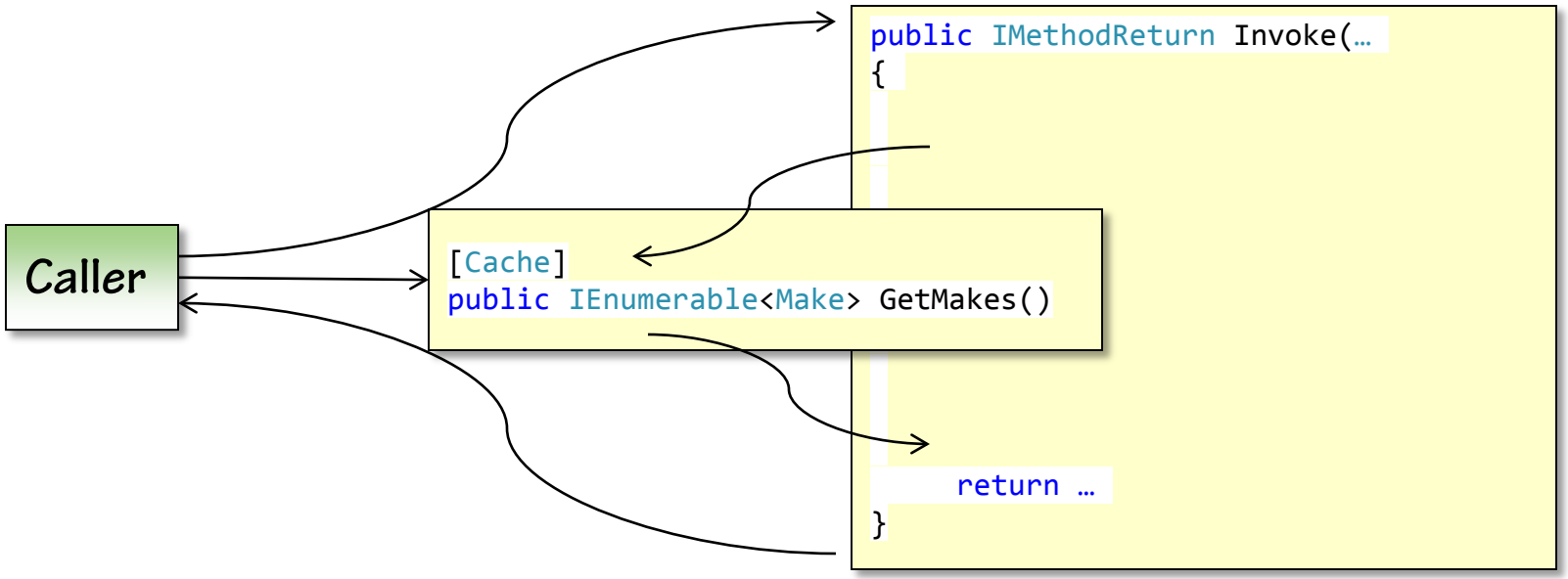
```
Log.Debug("VehicleRepository.GetMakes called");
```

  - Cache – wrapper over cache stores: memory, AppFabric etc.

```
Cache.Memory.Get<List<Make>>("GetMakes");
```

Caching Method Results with AOP

- **About AOP**
  - ”Aspect Oriented Programming in .NET” – Donald Belcham
- **Hooks, interceptors and the Container**



Caching Method Results with AOP

- Demo

Caching Method Results with AOP

- **Add the AOP hook**
  - `[Cache]` attribute
  - `virtual` method
- **Fetch instances from the AOP framework**
  - `Container.Register<>`
  - `Container.Get<>`
- **Generic implementation**
  - Abstracts the details

Deep-dive into the CacheCallHandler

- Demo

Deep-dive into the CacheCallHandler

- **CacheAttribute**
  - Inherits HandlerAttribute
  - Returns CacheCallHandler
- **CacheCallHandler**
  - Per-item configuration
  - Generate cache key
  - Get/check/set pattern
  - Serialization to fit any cache store
- **Container.Register<>**
  - InterceptionConstructor
  - VirtualMethodInterceptor

Deep-dive into the CacheCallHandler

- Register by type

```
Container.Register<VehicleRepository>(Lifetime.Singleton);
```

- Register by interface

```
Container.RegisterAll<IRepository>();
```

An Alternative AOP Pattern

- Demo

An Alternative AOP Pattern

- **Two patterns for AOP method-level caching**
  - Consumer uses Container explicitly
  - Target internalizes Container usage
- **Simple, generic caching patterns**
  - Reduce execution time
  - Remove blocking calls
- **Work to be done**
  - Deciding on the right cache store

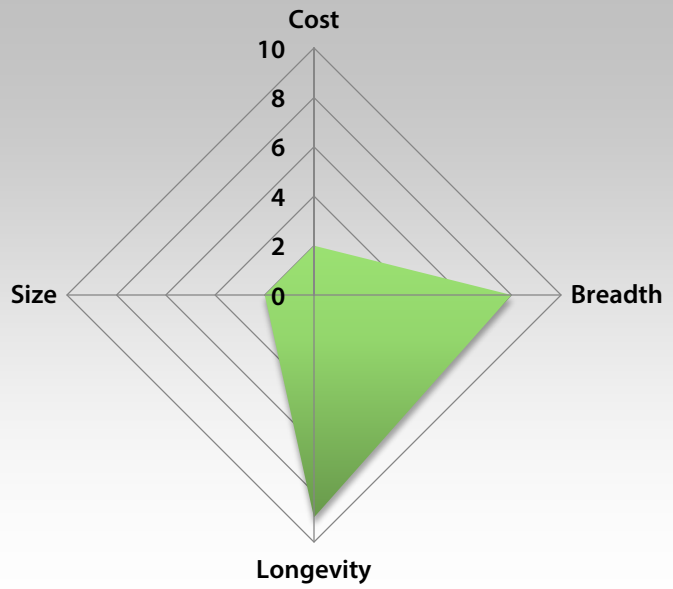
A Decision Matrix for Cache Items

- **Guideline criteria**
  - Cost – of fetching from the owner
  - Breadth – reuse throughout the solution
  - Longevity – how long the data stays fresh
  - Size – of the cached object
- **The right balance**
  - Identify suitable cache types
  - Maximise effectiveness

A Decision Matrix for Cache Items

- Vehicle Makes

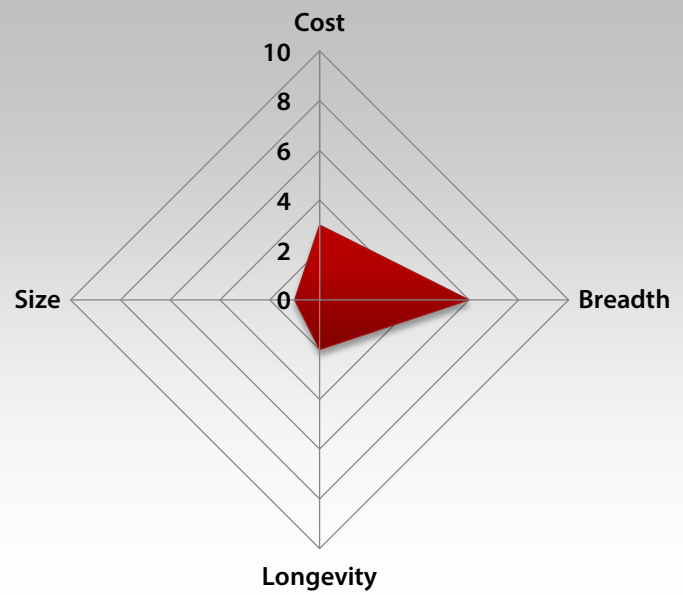
Cost	2
Breadth	8
Longevity	9
Size	2



A Decision Matrix for Cache Items

- CMS Content

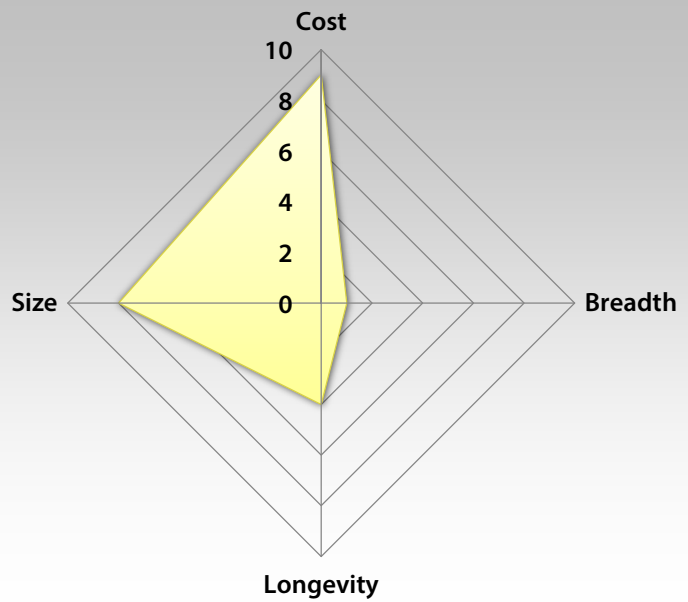
Cost	3
Breadth	6
Longevity	2
Size	1



A Decision Matrix for Cache Items

- Quote Prices

Cost	9
Breadth	1
Longevity	4
Size	8



A Decision Matrix for Cache Items

- **Attributes of the cache store**
  - Cost – of reading/writing to the store
  - Breadth – how widely available is the store
  - Longevity – how long can items live in the store
  - Size – physical limit to the store & cleanup policy

Summary

- **A tour of the demo solution**
  - Performance concerns – speed & resource use
- **Caching method results explicitly**
  - Core library & memory cache
- **Caching method results with AOP**
  - Using Unity IoC Container
- **A decision matrix for cache items**
  - Next module: local cache stores