私塾在线《高级软件架构师实战培训阶段二》

——跟着cc学架构系列精品教程

本部分课程概览

- 根据实际的应用需要，学习Web缓存和静态化的相关知识，以快速上手、理解并掌握Web缓存和静态化，大致包括：

1：什么是Web缓存、以及Web缓存的类型
2：为何要Web缓存
3：基本概念：重验证、更新
4：接下来要分每种类型来谈
（1）工作机制
（2）如何控制缓存和不缓存
（3）如何清除缓存，主动和被动
5：什么是静态化
6：为何需要静态化
7：静态化分类
8：常见静态化的实现，比如：访问数据多，并发高的，如首页， 比如大部分内容一样，少部分跟访问用户相关的，如列表页，详细页等等

Web缓存基础

- 什么是Web缓存

Web缓存指的是从发起请求的客户端，到执行功能的web服务器之间的，用来保存服务器输出结果的副本，并提供给发起请求的客户使用，这样的一些缓存的统称。

- Web缓存的类型

1：浏览器缓存
2：CDN（Content Delivery Networks）
3：反向代理服务器缓存
4：Web服务器端的缓存，通常是应用程序来实现的动态页面缓存

- 为何要使用Web缓存

1：加快Web应用响应速度，提高用户满意度
2：减少服务端的处理压力，以同样的资源来支撑更大的访问量和高并发

- 重验证

通常存放在缓冲中的内容是有时效性的，因此需要经常对缓存中的内容做有效性检查，也叫“重验证”。

- 更新

更新操作主要是为了保证缓存中的内容与服务器中的内容保持一致

浏览器缓存-1

- 工作机制

  浏览器缓存主要是通过在Http头部增加：Last-Modified，If-Modified-Since，Expires，Cache-Control等标识，和服务器进行协商，以确定是否采用浏览器的缓存。

- Http中的更新途径：定义文档过期时间、执行重验证

一：文档过期时间

1：在服务端返回的Http头设置：Expires

2：在服务端返回的Http头设置： Cache-Control: max-age

3：同时具有的时候，max-age具有更高的优先级

4：要注意：就算是过期了，并不标志内容一定不对了，只是提醒要求重验证

5：在Java中的设置示例：response.setHeader("Cache-Control","max-age=100 “)

二：执行重验证

1：重验证的目的：是与服务器交互，判断缓存中的文档是否已经改变。如已经改变，就重新下载一份最新的文档，来代替缓存内容；如没有做修改，则只需获取新的HTTP头部（可能包含新的过期时间），并更新缓存中的头部。

浏览器缓存-2

2：在request的Http头设置：If-Modified-Since

如果服务端返回的报头中有“Last-Modified”，那么客户端在下一次请求报头中会包含“If-Modified-Since”，这两个头部是相互对应的。

当服务器收到客户端请求的“If-Modified-Since”之后，服务器通过比较这两个时间，若“Last-Modified”更大，表明客户端缓存中的内容已经过时，服务器会将最新的文档（新的Header）返回给客户端，状态码为200；否则认为客户端缓存中的内容仍然是最新的，只需向客户端返回304状态码，同时包含最新的HTTP头部。

3：在request的Http头设置：If-None-Match

这是一种比较文档标签（Entity tags，Etags）的方式。基本思想是为每一个文档生成一个Etag，它可以是某个序列号、版本号或者检验码。同样“If-None-Match”头部是与服务器端的“Etag”头部相对应的，这样服务器端只需要比较标签号就可以判断出客户端缓存中的文档是否是最新的。

浏览器缓存-3

- 浏览器缓存控制

HTTP规范中定义了服务器如何约束、限制客户端缓存的头部，按照优先级分别有：

1：Cache-Control: no-store ：禁止缓存，浏览器会删除缓存的内容

2：Cache-Control: no-cache ：禁止缓存，浏览器可能会缓存，只是限制本地缓存不能在没有与服务器执行一致性检查的前提下直接响应用户

3：Cache-Control: must-revalidate ：要求缓存在响应用户请求之前一定要先保证缓存中的文档副本是最新的

4：Cache-Control: max-age ：设置过期时间，若max-age=0，则表示不能缓存文档或者每次访问缓存前必须执行一致性检查

5：Expires

6：在Java中设置示例：

```
response.setHeader("Cache-Control", "no-cache");
response.addHeader("Cache-Control", "no-store");
response.setHeader("Pragma", "no-cache");
response.setDateHeader("Expires", -1);
response.setDateHeader("max-age", 0);
```

浏览器缓存-4

- 要注意：不同浏览器对这些功能的实现可能不同
- F5和Ctrl+F5 刷新的实现也不同：F5通常是驱使浏览器去执行一次一致性检查；而“Ctrl+F5”则是在删除本地缓存的前提下，去执行一致性检查

- 如何主动通知浏览器缓存失效

这个问题其实也是静态资源版本更新的问题，基本的解决思路：

1：页面引入js、css等统一做成单独的jsp

2：页面上使用ESI来引入这些jsp

3：如果后端修改了这些静态资源的内容，那么在这些单独的jsp上，修改链接，给这些链接加上“?版本号”或者“?时间戳”等，反正要修改一下这些链接

4：然后主动请求PURGE，以让Varnish里面缓存这些jsp的内容失效，这样Varnish会从后端重新获取这些jsp的内容，拼接好新的内容后返回给客户端

5：客户端浏览器会发起新的请求，去重验证资源是否有更新，发出的请求是按照这些新的链接发出的，那么服务器端是没有这些链接对应的Etag的，就会重新下载新的资源了

6：当然，由于更新资源和缓存内容之间会有一个时间差，对于大型的高并发的应用，这种不同步可能会造成一点小问题，这就需要更专业的工具来同步更新服务器的静态资源了。

反向代理服务器缓存

- Varnish缓存

1：Varnish本身就是设计来做高速缓存的

2：一般按照请求的url进行缓存，可以订制自己的Hash策略

3：缓存的时候，按照内容的类型不同，设置好不同的ttl

4：如果在缓存期内，后端内容发生了改变，后端可以主动发出PURGE请求，让Varnish清除对应的缓存

5：对于动态页面的缓存，如果里面有需要根据请求的用户进行变化的内容，可以采用ESI技术，也可以采用Ajax的方式来获取动态内容

6：对于热点内容、高并发的内容，可以采用Varnish缓存 + 后端对动态内容静态化的组合方案来解决

Web应用对页面的缓存

- Web应用对页面的缓存

1：前面使用了Varnish缓存的话，一般Web应用不需要对页面进行缓存

2：如果要做的话，常见思路是：

做一个Filter，检查请求的URL是否有对应的缓存（一般就放到内存中），或者是对应的生成好的静态Html，如果有，就获取内容直接返回；如果没有，就调用静态化的程序，生成一份要返回的Html的内容，设置到缓存中，然后返回。

3：当然，如果Web应用自己做页面内容缓存的话，那么缓存的过期，还有缓存的清除、更新等功能，就都得自己来做了

静态化-1

- 什么是静态化

  就是把动态内容转化成为静态的、可以直接返回给客户端使用的过程

- 为何需要静态化

1：加快网站的运行速度，提升客户体验

2：减少对后端服务器的压力

3：为了SEO

- 静态化分类

  一般分成真正静态化 和 伪静态化

- 实现真正静态化

  基本的思路大都是：根据模板，从后端服务器和数据库动态获取数据，然后组合起来，形成静态页面的内容。

- 热点或高并发内容实现真正静态化的方案

1：Varnish配置一个缓存的时间，比如3分钟

2：后端程序利用这个时间，在这段时间之内，调用静态化功能生成静态页面

静态化-2

- 静态化的一些说明

1：要结合应用场景分析哪些页面，哪些内容适合静态化
2：跟客户信息相关的内容，不要包含在静态化内容里面
3：经常变化的内容，最好不要包含在静态内容里面
4：更新频率过高的内容，不适合静态化
5：后端实现静态化的时候，通常都需要配套做静态化的管理，比如设置哪些页面需要静态化；多长时间做一次静态化；静态化后的内容存放路径等等
6：从某种意义上说，Varnish的缓存做的就是静态化的功能