## 前端缓存规则

1. 目的

用于提高数据加载效率，可考虑的优化点例如：

* 懒加载。如具有主从结构的列表数据，一般先加载主体列表数据，然后通过缓存机制懒加载对应的从属数据。
* 缓存用于检索的字典数据。

1. 整体结构

缓存作为客户端（浏览器）和服务器之间的中间层，用于缓冲对数据的访问。目前的存储介质分为**浏览器内存**和**浏览器本地文件**。客户端需指定其中一种方式。前者的作用范围是**页面**，后者的作用范围是**域**，前者的时间范围是**当前会话（**关闭页面就释放**）**，后者的时间范围为**永久**。两者具有相同的[缓存策略](#缓存策略)。缓存唯一性通过**url**和**请求参数**（get方式对应查询参数，post方式对应请求体的请求参数）来确定。

1. 机制

通过在url上附加相关参数，拦截ajax请求，注入缓存逻辑。

1. 缓存策略

* expire：缓存最后一次刷新到当前经过的时间。用户可指定一个值，比如：10000，意思是如果缓存在最近的10s内没有刷新，那么缓存失效，同时删除缓存；否则，缓存有效；
* liveTime：如果不主动访问缓存，那么缓存将一直驻留在存储介质中，这个参数用于指定缓存自动失效的时间。比如：指定：30000，那么缓存将在最后一次刷新的30s后自动失效。

1. 接口

* 使用缓存需要在对应的服务定义里加入缓存的支持参数：cacheable: true
* Service方法的第二个参数为缓存参数，包括：

{

local: true/false, // 可选。 true：使用本地文件缓存，false：使用内存缓存。不指定时使用内存缓存

expire: 10000, // 必须。查看[缓存策略](#缓存策略)

liveTime: 30000 // 可选，默认10分钟。查看[缓存策略](#缓存策略)

}

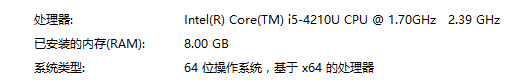
* Service url方法的第二个参数：同上

1. 控件配合的优化和支持

* ComboBox 增加queryLimit属性：当检索结果数据量很大的时候，控件渲染会很慢，指定这个参数后，只渲染检索结果前queryLimit条数据。
* ComboBox 增加queryFilter属性。该属性是个函数，用于实现自定义的数据过滤，第一个参数是数据项，第二个参数是用户输入的检索值，返回true表示符合检索条件。
* ComboBo增加applyKeyBeforeFilter。如果指定true，queryFilter传入的数据项事先经过默认的检索条件过滤，否则完全由queryFilter进行所有数据的过滤。默认值为true。

七．注意事项

* 目前本地缓存通过indexedDB来存储数据，由于indexedDB没有提供同步方式的api来访问数据，所以，**当使用本地文件缓存的时候，Service方法将无法进行同步调用**。
* 本地文件缓存可在不同的登录用户之间共享，由于缓存是通过url及其参数来区分的，所以**不建议使用本地文件的方式来缓存和当前登录信息相关的数据**。
* **个人建议：**前端缓存只是辅助作用，不建议**过度存储**大量数据。关键的性能提升还在于服务器端。下面是一些前端存储相关的性能测试数据，供开发人员使时的考量：
* 配置：



* 数据量：100000 行 \* 20 字段
* 针对以上数据的下列操作
* JSON.stringify: 400ms
* JSON.parse: 1600ms
* 本地文件存储json值：1000ms
* 本地文件获取json值：500ms
* 本地文件存储数据：1500ms
* 本地文件获取数据：1000ms
* 内存进行两个字段的字符串包含检索（or）：50ms
* indexedDB在索引字段平衡的情况下，针对以上数据的操作：
* 存储：20s
* 获取：2s
* 全表检索：

5秒(queryLimit=null)，

150ms(queryLimit=500) 130(queryLimit=200) 100ms(queryLimit=100)

* 应用索引：

1000ms(queryLimit=null)

150ms(queryLimit=500) 130(queryLimit=200) 100ms(queryLimit=100)