性能问题诊断

当系统出现性能缓慢现象时，首先应分析是什么原因导致缓慢再分析如何改进。

1. 信息收集

首先应当确定是交互次数、还是客户端、服务器性能瓶颈引起的，在这个过程中可以应该使用Charles跟踪工具跟踪整个交互过程。

2. 问题解决方案

2.1 交互次数过多

由于每一次IE等浏览器与服务器进行一次交互都需要建立HTTP连接，这个过程会花费一定的时间，应该通过以下一些手段避免。最理想的情况下每一次用户操作只调用1～2个服务器方法：

1. 合并多个CSS文件为单一文件，例如Smartbi会通过bof_merge_css.jsp返回所有的CSS
2. 使用jsloader.resolveMany方法预加载多个JavaScript文件，例如QueryView.js会一次性请求28个JavaScript类
3. 一次用户点击操作会调用很多次服务器方法时，应当在bof.composite.CompositeModule模块中增加方法，并在前端JavaScript中通过util.compositeInvoke一次性调用服务器方法并返回

2.2 服务器缓慢

服务器缓慢可以通过JProfiler跟踪，应该确认是否有方法调用次数过多、找到花费时间最多的方法进行优化：

1. 当一个方法会在一次用户点击操作重复调用多次时，应当考虑使用Cache（bof.cache.CacheBuilderFactory）。也可以使用RequestAttribute缓存调用结果，例如bof.catalogtree.CatalogTreeModule中权限判断都有使用RequestAttribute避免删除一个节点时重复判断
2. 当一个方法调用花费非常多时间时，应当考虑是否代码有优化余地。例如：

a) 避免重复创建对象

b) 在拼接字符串数量较多时由调用者传递StringBuilder到当前类中，并始终使用这个StringBuilder进行添加字符串操作而不应使用另外的字符串加号操作，例如以下红色部分应当避免：

```
public void toXML(StringBuilder buff) {
    buff.append(id).append(name);
    buff.append(id + name);
}
```

c) 操作DOM子节点时，尽量减少parent. getElementsByTagName(“name”).items(0)这样的全DOM树搜索的操作。可以使用bof.util.XmlUtility.getChildElementByTagName减少搜索深度

2.3 客户端缓慢

由于受到浏览器JavaScript引擎的限制，客户端是目前Smartbi性能瓶颈，在不少项目中都会出现效率不理想的情况。客户端的性能跟踪可以使用IE 8 Developer Bar自带的“探查器”，根据这个分析结果优化花费时间最多的方法。优化的方法有以下：

1. 遍历DOM节点时应该使用firstChild、nextSibling而不是childNodes

```
for(int i = 0, len = node.childNodes.length; i < len; i++) {
        var child = node.childNodes[i];
        …
}
```

应当优化为：

```
var child = node.firstNode;
while(child) {
        …
        child = child.nextSibling;
}
```

2. 避免操作DOM节点的style属性，无论读取、更改style属性都会花费大量时间

动态往TABLE中添加行、列时应当使用createElement(“TBODY”)、createElement(“TR”) 、createElement(“TD”)，例如：

```
for(var i = 0; i < 300; i++) {
        var row = table.insertRow(-1);
        for(var j = 0; j < 20; j++) {
                var cell = row.insertCell(-1);
                cell.innerText = i + ":" + j;
        }
```

```
}
```

这个代码花费时间是5.5秒。在优化后花费为0.4秒：

```
var tBody = table.tBodies[0];
for(var i = 0; i < 300; i++) {
        var row = document.createElement("TR");
        tBody.appendChild(row);
        for(var j = 0; j < 20; j++) {
                var cell = document.createElement("TD");
                row.appendChild(cell);
                cell.innerText = i + ":" + j;
        }
}
```

3. 性能优化

查看性能优化章节。