Struts2 内部是如何工作的

**struts的核心原理就是通过拦截器来处理客户端的请求，经过拦截器一系列的处理后，再交给Action。**

下面先看看struts官方的工作原理图：

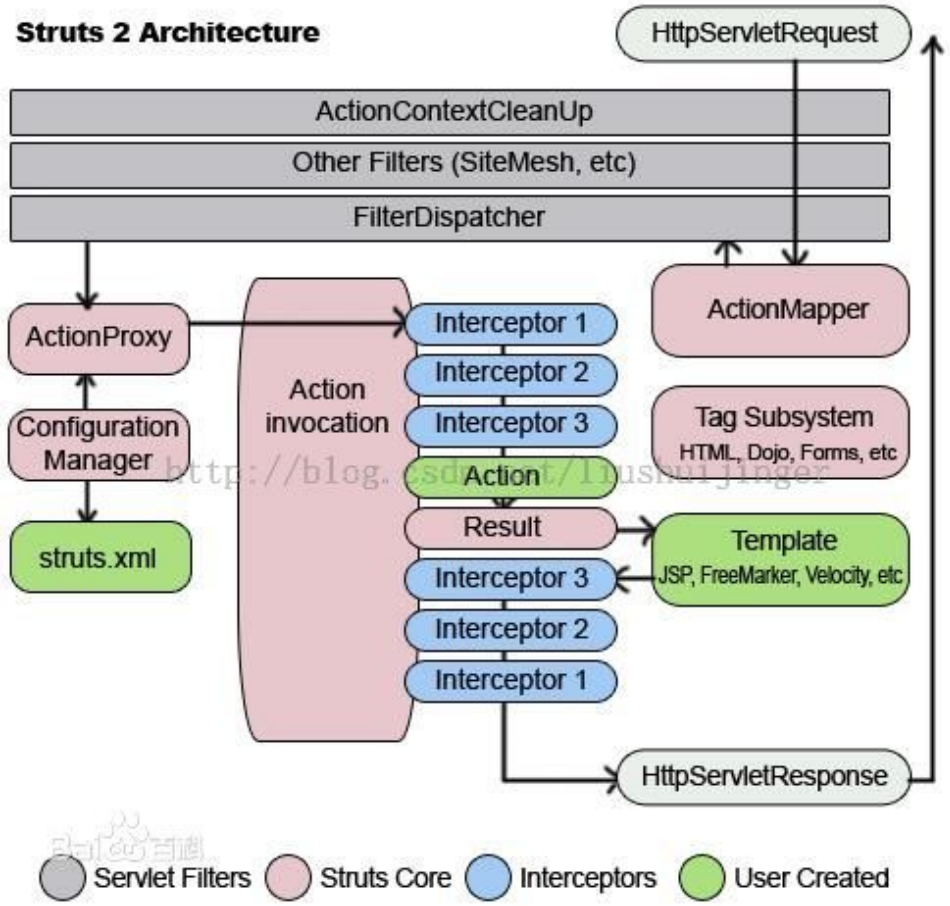


图1 struts原理图

简单分析一下：

首先客户端发来HttpServletRequest请求，传递给FilerDispatcher（ActionMapper是访问静态资源（struts的jar文件等）时用的，平时很少用），

然后FilerDispatcher会为我们创建一个ActionProxy，ActionProxy会通过ConfigurationManager获得struts.xml文件中的信息，ActionProxy拥有一个ActionInvocation实例，通过调用ActionInvocation的invoke()方法，来挨个处理Interceptor，最后处理Action，接着Result返回，再逆序经过Interceptor，最后得到HttpServletResponse返回给客户端。

如果不太明白呢，那就看看下面这张时序图，也许你就懂了：

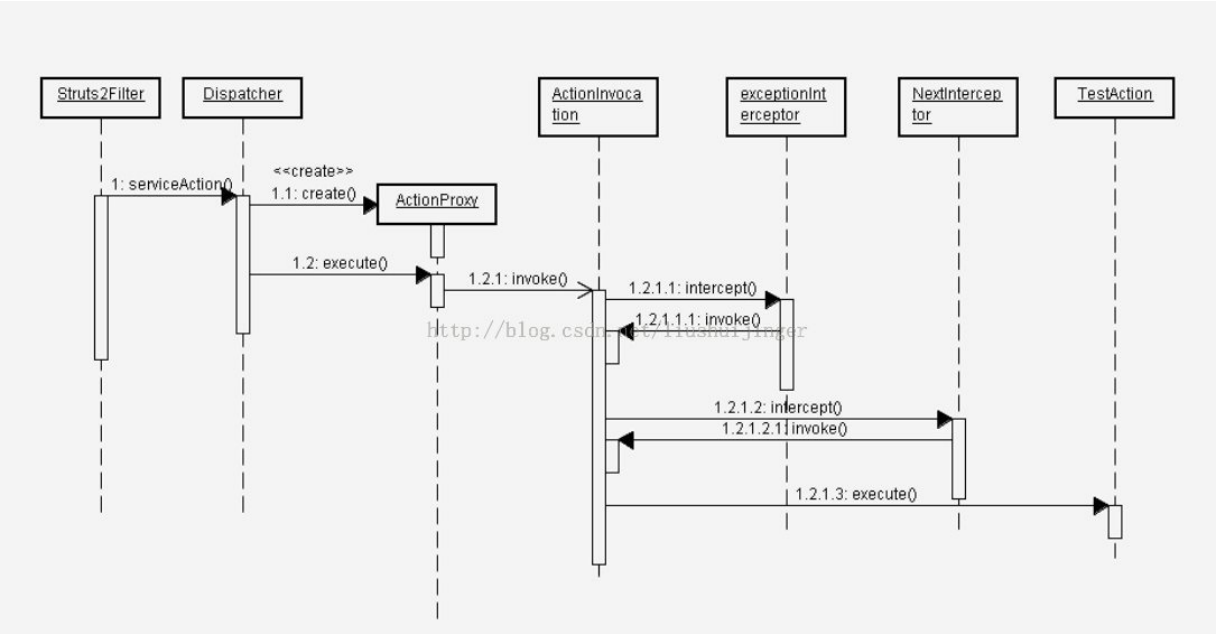


图2 struts原理时序图

上面的时序图逻辑就比较清晰了，我就不过多解释了。

看完struts的原理图，我们还是需要通过代码来进一步了解它具体是怎么实现的。

首先，我们需要一个ActionInvocation：

```
package com.tgb.struts;

import java.util.ArrayList;

import java.util.List;


public class ActionInvocation {

    List<Interceptor> interceptors = new ArrayList<Interceptor>();

    int index = -1;

    Action a = new Action();


    public ActionInvocation() {

        this.interceptors.add(new FirstInterceptor());

        this.interceptors.add(new SecondInterceptor());

    }


    public void invoke() {

        index ++;

        if(index >= this.interceptors.size()) {

            a.execute();

        }else {
```

```
            this.interceptors.get(index).intercept(this);
        }
    }
}
```

我们实现的ActionInvocation是将Interceptor写在里面的，但实际上是通过反射加载的，原理同之前写的Spring与Hibernate的博客，相同的代码就不在这里占用篇幅了，也没啥意思。不知道怎么实现的朋友请查看前面几篇博客。

接下来是我们的Interceptor接口以及两个简单的实现：

```
package com.tgb.struts;

public interface Interceptor {
    public void intercept(ActionInvocation invocation) ;
}
```

```
package com.tgb.struts;

public class FirstInterceptor implements Interceptor {

    public void intercept(ActionInvocation invocation) {
        System.out.println("FirstInterceptor Begin...");
        invocation.invoke();
        System.out.println("FirstInterceptor End...");
    }

}
```

```
package com.tgb.struts;

public class SecondInterceptor implements Interceptor {

    public void intercept(ActionInvocation invocation) {
        System.out.println("SecondInterceptor Begin...");
        invocation.invoke();
        System.out.println("SecondInterceptor End...");
```

```
    }

}
```

然后就是我们的Action：

```
[java] view plaincopy在CODE上查看代码片派生到我的代码片
package com.tgb.struts;

public class Action {
    public void execute() {
        System.out.println("Action Run...");
    }
}
```

最后是我们的客户端调用：

```
package com.tgb.struts;

public class Client {
    public static void main(String[] args) {
        new ActionInvocation().invoke();
    }
}
```

差点忘了，还有我们最后的执行结果：

```
FirstInterceptor Begin...
SecondInterceptor Begin...
Action Run...
SecondInterceptor End...
FirstInterceptor End...
```

通过上面的执行结果，我们可以很清楚的看到，请求来的时候会按照顺序被所有配置的拦截器拦截一遍，然后返回的时候会按照逆序再被拦截器拦截一遍。