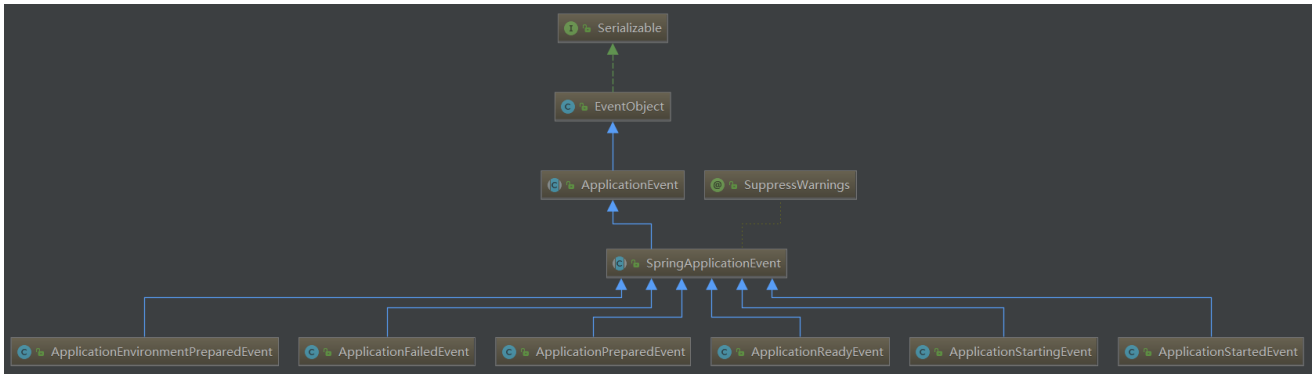
事件机制 之 SpringApplicationEvent

概述

Spring的事件为Bean与Bean之间的通信提供了支持，当我们系统中某个Spring管理的Bean处理完某件事后，希望让其他Bean收到通知并作出相应的处理，这时可以让其他Bean监听当前这个Bean所发送的事件。

类图



源码解析

```
package org.springframework.boot.context.event;

import org.springframework.boot.SpringApplication;
import org.springframework.context.ApplicationEvent;

/**
 * 与SpringApplication相关的ApplicationEvent基类
 *
 */
@SuppressWarnings("serial")
public abstract class SpringApplicationEvent extends ApplicationEvent {

    private final String[] args;

    public SpringApplicationEvent(SpringApplication application, String[] args) {
        super(application);
        this.args = args;
    }

    public SpringApplication getSpringApplication() {
        return (SpringApplication)this.getSource();
    }

    public final String[] getArgs() {
        return this.args;
    }

}
```

使用

要实现事件的监听，我们要做两件事：
1：自定义事件，继承ApplicationEvent接口
2：定义事件监听器，实现ApplicationListener
3：事件发布类

```
/**
```

```
 * 自定义事件
 */
public class MessageEvent extends ApplicationEvent {

    /**
     * 序列化
     */
    private static final long serialVersionID = 1L;

    /**
     * 收件人
     */
    public String receiver;

    /**
     * 收件内容
     */
    public String content;

    public MessageEvent(Object source) {
        super(source);
    }

    public MessageEvent(Object source, String receiver, String content) {
        super(source);
        this.receiver = receiver;
        this.content = content;
    }

    public void output(){
        System.out.println("I had been sand a message to " + this.receiver);
    }
}

/**
 * 定义监听事件
 */
@Component
public class MessageLisenter implements ApplicationListener<MessageEvent> {

    @Override
    public void onApplicationEvent(MessageEvent messageEvent) {
        messageEvent.output();
        System.out.println(messageEvent.receiver + "received msg : " + messageEvent.content );
    }
}

@Component
public class Publisher {

    @Autowired
    ApplicationContext applicationContext;

    public void publish(Object source, String receiver, String content) {
        applicationContext.publishEvent(new MessageEvent(source, receiver, content));
    }
}


@Controller
@ResponseBody
@RequestMapping("/")
public class HelloController {
```

```
    @Autowired
    public Publisher publisher;

    @RequestMapping("hello")
    public void Hello() {
        publisher.publish("Hello,World!","Mr.Lensen", "I Love U");
    }
}
```