# 一、定义

```
    Web三大组件之一，在Web中有很多监听器。如ServletRequestListener、HttpSessionListener、ServletContextListener.
```

```
其中重点掌握ServletContextListener,此监听器主要用于监听ServletContext对象的创建和销毁。
```

# 二、ServletContextListener接口的方法

## 1、创建

```
void contextInitialized(ServletContextEvent sce)：ServletContext对象创建后会调用该方法。
```

## 2、销毁

```
void contextDestroyed(ServletContextEvenet sce)：ServletContext对象被销毁之前会调用。
```

# 三、所有监听器操作步骤

## 1、定义一个类，实现接口

## 2、复写方法

## 3、配置

### （1）注解配置

### （2）web.xml配置

## 4、实现

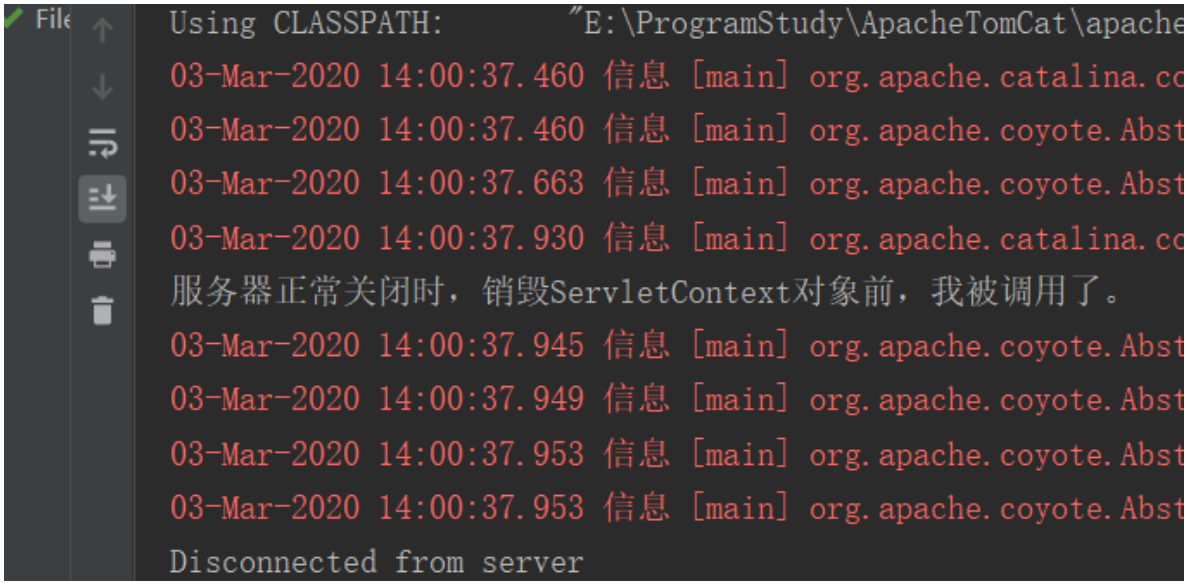
### （1）web.xml配置法

```
定义类，实现ServletContextListener接口，复写方法，在web.xml中定义一个listener标签，在其子标签listener-class中写上定义类的全类名，整个流程完成。
```

```java
package Listener;

import javax.servlet.Servlet;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class ContextListenerForText implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent servletContextEvent) {
        System.out.println("服务器创建时，创建ServletContext对象，所以我就被调用了。");
    }

    @Override
    public void contextDestroyed(ServletContextEvent servletContextEvent) {
        System.out.println("服务器正常关闭时，销毁ServletContext对象前，我被调用了。");
    }
}
```

```xml
<web-app xmlns="http://xmlns.jcp.org/xml/ns/java
         xmlns:xsi="http://www.w3.org/2001/XMLSc
         xsi:schemaLocation="http://xmlns.jcp.or
         version="4.0">

    <!--配置监听器-->
    <listener>
        <listener-class>
            Listener.ContextListenerForText
        </listener-class>
    </listener>

</web-app>
```

```
Output
03-Mar-2020 14:00:16.987 警告 [RMI TCP Connection(3)-127.0.0.1] org.apache.tom
服务器创建时，创建ServletContext对象，所以我就被调用了。
[2020-03-03 02:00:17,126] Artifact FileterListener:war exploded: Artifact is d
[2020-03-03 02:00:17,126] Artifact FileterListener:war exploded: Deploy took 3
```




### （2）注解配置法

在定义的类上加一个`@WebListener`

```
@WebListener
public class ContextListenerForText implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent servletContextEvent) {
        System.out.println("服务器创建时，创建ServletContext对象，所以我就被调用了。");
    }

    @Override
    public void contextDestroyed(ServletContextEvent servletContextEvent) {
        System.out.println("服务器正常关闭时，销毁ServletContext对象前，我被调用了。");
    }
}
```

## 四、应用

### 1、加载资源文件

### （1）思路

获取`ServletContext`的另一种方式：`servletContextEvent`中调用`getServletContext`方法。

思路：我们要加载资源文件，以便全局使用。在这个过程中，我们在程序中不应该写死资源路径，而应该采用动态的方式。正好`ServletContext`对象中有`getInitParameter`方法，可以在`web.xml`的`<context-param>`标签中通过键获取值。其中值为真实的路径。我们就实现了动态获取。

### （2）实现

#### ①Listener代码

```
package Listener;

```

```
import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import java.io.FileInputStream;
import java.io.FileNotFoundException;

public class ContextListener implements ServletContextListener {

    /*用于监听ServletContext对象的创建。由于ServletContext是服务器启动后自动创建，所以这个方法是服务器启动后调用*/
    @Override
    public void contextInitialized(ServletContextEvent servletContextEvent) {

        //1、使用servletContextEvent获取ServletContext
        ServletContext context = servletContextEvent.getServletContext();

        /*2、加载资源文件,这种资源文件是全局的资源文件，即整个项目都要使用的资源文件。不应该写死，而是让开发者指定加载的资源路径，而不修改代码，正好ServletContext里有
       getInitParameter方法.使用键获取值。我们只需要在web.xml配置键值参数即可.*/
        String resourceLocation = context.getInitParameter("resourceLocation");

        //3、获取真实路径
        String realPath = context.getRealPath(resourceLocation);

        //4、加载进内存
        try {
            FileInputStream fis = new FileInputStream(realPath);
            //如果能打印出来，说明资源加载成功，试验完毕
            System.out.println(fis);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }


    /*在服务器关闭后，ServletContext对象被销毁，当服务器正常关闭后该方法被调用*/
    @Override
    public void contextDestroyed(ServletContextEvent servletContextEvent) {
        System.out.println("ServletContext对象被销毁了。");
    }
}

```

**②xml代码**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <!--配置监听器-->
    <listener>
        <listener-class>
```

```
            Listener.ContextListener
        </listener-class>
    </listener>

    <!--配置资源路径-->
    <context-param>
        <param-name>resourceLocation</param-name>
        <param-value>/WEB-INF/classes/a.txt</param-value>
    </context-param>

</web-app>
```

事件监听机制可能理解的错误，以后再来改。

事件监听机制

事件：一件事情。相当于ServletContext对象被创建了

事件源：事情发生的地方。事件源相当于Tomcat

监听器：一个对象，相当于定义的类实例化.

注册监听：将事件、事件源、监听器绑定在一起。相当于在配置中配置了监听器。

比如说：就拿javascript举例，一个按钮，注册一个单击事件，来执行一个function函数。