

接着上面笔记中的ServletConfig接口中最后一个方法，
我们新建一个web项目c-ServletContext，
该项目中的两个Servlet实现类为AServlet和BServlet，
将init中的局部变量ServletConfig对象赋值给一个私有的成员变量config，
在各自的service方法中用该对象调用ServletConfig接口中的获取Servlet对象上下文的方法getServletContext分别在两个实现类中获取，并输出到浏览器。其他配置就不赘述了。

web.xml文件

```
<?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">

    <servlet>
        <servlet-name>servletcontext1</servlet-name>
        <servlet-class>com.servlet.AServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>servletcontext1</servlet-name>
        <url-pattern>/a</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>servletcontext2</servlet-name>
        <servlet-class>com.servlet.BServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>servletcontext2</servlet-name>
        <url-pattern>/b</url-pattern>
    </servlet-mapping>
</web-app>
```

两个实现类中方法的代码

```
private ServletConfig config = null;

@Override
public void init(ServletConfig servletConfig) throws ServletException {
    this.config = servletConfig;
}
```

```

@Override
public ServletConfig getServletConfig() {
    return config;
}

@Override
public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {

    ServletConfig config = getServletConfig();
    ServletContext application = config.getServletContext();
    System.out.println(application);
}

@Override
public String getServletInfo() {
    return null;
}

@Override
public void destroy() {
}

```

部署运行，在浏览器输入URL转到index页面，点击其中的两个连接分别输出两个Servlet对象中获取的ServletContext对象，可以看到运行窗口输出如下

```

Connected to server
[2021-01-05 06:48:58,490] Artifact c-ServletContext:war exploded: Artifact is being deployed, please wait...
[2021-01-05 06:48:58,739] Artifact c-ServletContext:war exploded: Artifact is deployed successfully
[2021-01-05 06:48:58,739] Artifact c-ServletContext:war exploded: Deploy took 249 milliseconds
org.apache.catalina.core.ApplicationContextFacade@221945cc
org.apache.catalina.core.ApplicationContextFacade@221945cc
05-Jan-2021 18:49:08.243 洪℃侖 [Catalina-utility-2] org.apache.catalina.startup.HostConfig.deployDirectory 銅崧eb 辜旂穀綿嬪簪
05-Jan-2021 18:49:08.271 洪℃侖 [Catalina-utility-2] org.apache.catalina.startup.HostConfig.deployDirectory Web辜旂穀綿嬪簪鏽

```

两个Servlet对象输出的两个ServletContext对象内存地址相同，是同一个对象。并且该接口由Tomcat服务器实现，保存的地址是org.apache.catalina.core包下，实现类的类名是ApplicationContextFacade.java

结论：

--javax.servlet.ServletContext接口：

1、Tomcat服务器完成对这个接口的实现，完整的类名为：

org.apache.catalina.core.ApplicationContextFacade

javaweb程序员无需关心这个实现类，只需要面向ServletContext接口调用接口中的方法就行了。

2、ServletContext具体是什么？什么时候被创建？什么时候被销毁？创建几个？

--ServletContext翻译为：Servlet上下文

--一个webapp只有一个web.xml文件，该文件在服务器启动时被解析

--一个webapp只有一个ServletContext对象，该对象也是在服务器启动阶段被实例化

--ServletContext对象在服务器关闭时被销毁

--ServletContext对象对应的是web.xml文件，也可以说这个对象在项目运行时是web.xml文件的代表

--ServletContext对象是所有Servlet对象周围环境的代表【在同一个webapp中所有Servlet对象共享一个环境，也就是共享同一个ServletContext对象】

--当所有用户希望共享一个数据，可以将该数据存放到ServletContext对象中，但是放进ServletContext对象中的数据不建议涉及修改操作，因为该对象是多线程共享的，修改会有线程安全问题。

3、ServletContext接口中常用的重点方法

--Object getAttribute(String name)

获取ServletContext范围中的数据，在底层该方法是调用了集合的get方法，Object value=map.put(key,value)

--void removeAttribute(String name)

移除ServletContext范围中的数据，在底层该方法是调用了集合的remove方法，map.remove(key)

--void setAttribute(String name, Object object)

向ServletContext范围中添加数据，在底层该方法是调用了集合的put方法，map.put(key,value)

下面两个方法和ServletConfig接口中的一样，只不过获取web.xml文件中的信息是在<servlet></servlet>标签之外定义的上下文初始化参数标签<context-param>标签中的内容，具体xml文件代码如下面所示。

--String getInitParameter(String name)

--Enumeration getInitParameterNames()

--String getRealPath(String path)

该方法是获取文件的绝对路径，演示如下

4、Servlet、ServletConfig和ServletContext之间的关系

--一个Servlet对象对应一个ServletConfig对象

--一个webapp中的所有Servlet对象共享一个ServletContext对象

5、从测试将数据存储到ServletContext范围以及从中读取的操作中我们知道ServletContext范围可以跨用户传递，不同用户点击连接照样可以获取数据可以存储数据。

1、测试String getInitParameter(String name)和Enumeration getInitParameterNames()方法，xml文件中上下文配置参数如下

```
<?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">
```

<!--web.xml文件中配置上下文参数，该信息在服务器启动阶段解析xml文件时自动封装到ServletContext对象中-->

```
<context-param>
  <param-name>username</param-name>
  <param-value>admin</param-value>
</context-param>
<context-param>
  <param-name>password</param-name>
  <param-value>123</param-value>
</context-param>

<servlet>
```

```

        <servlet-name>servletcontext1</servlet-name>
        <servlet-class>com.servlet.AServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>servletcontext1</servlet-name>
        <url-pattern>/a</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>servletcontext2</servlet-name>
        <servlet-class>com.servlet.BServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>servletcontext2</servlet-name>
        <url-pattern>/b</url-pattern>
    </servlet-mapping>
</web-app>

```

service方法中的代码

```

@Override
public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {

    ServletConfig config = getServletConfig();
    ServletContext application = config.getServletContext();
    //System.out.println(application);

    //ServletContext接口中获取所有上下文初始化参数的name的方法
    Enumeration<String> names = application.getAttributeNames();
    //遍历集合，并获取上下文初始化参数name对应的value
    while (names.hasMoreElements()){
        String name = names.nextElement();
        String value = application.getInitParameter(name);
        System.out.println(name+" = "+value);
    }
}

```

结果如下

```

05-Jan-2021 21:38:06.221 洪℃饨 [Catalina-utility-2] org.apache.catalina.startup.HostConfig.deployDirecto
05-Jan-2021 21:38:06.296 洪℃饨 [Catalina-utility-2] org.apache.catalina.startup.HostConfig.deployDirecto
password = 123
username = admin

```

测试String getRealPath(String path)方法，获取文件绝对路径

```

@Override
public void service(ServletRequest servletRequest, ServletResponse
servletResponse) throws ServletException, IOException {

    ServletConfig config = getServletConfig();
    ServletContext application = config.getServletContext();
    //System.out.println(application);
}

```

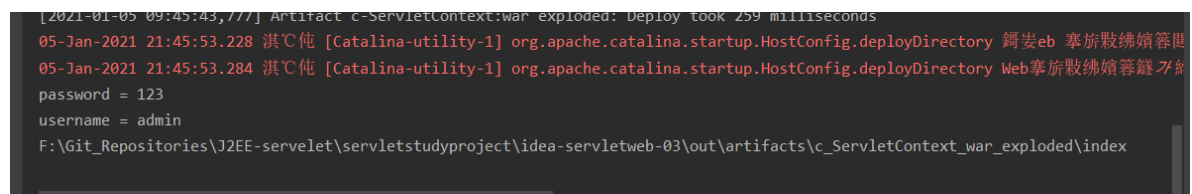
```

//ServletContext接口中获取所有上下文初始化参数的name的方法
Enumeration<String> names = application.getInitParameterNames();
//遍历集合，并获取上下文初始化参数name对应的value
while (names.hasMoreElements()){
    String name = names.nextElement();
    String value = application.getInitParameter(name);
    System.out.println(name+" = "+value);
}

//获取文件绝对路径方法
//下面这样写必须webapp根路径(即WEB-INF目录上一级)下有该文件
String absolutePath = application.getRealPath("/index.html");
}

```

结果



```

[2021-01-05 09:45:43,777] Artifact c-ServletContext:war exploded: Deploy took 259 milliseconds
05-Jan-2021 21:45:53.228 洪℃佬 [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployDirectory 鐳峇eb 辜旂敷绫嶃嶃嶃嶃
05-Jan-2021 21:45:53.284 洪℃佬 [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployDirectory Web辜旂敷绫嶃嶃嶃嶃
password = 123
username = admin
F:\Git_Repositories\J2EE-servlet\servletstudyproject\idea-servletweb-03\out\artifacts\c_ServletContext_war_exploded\index

```

测试void setAttribute(String name, Object object)和Object getAttribute(String name)方法，新建一个entity包，里面先新建一个实体用户类，代码如下

```

package com.entity;

public class User {

    private String usercode;

    private String username;

    public String getUsercode() {
        return usercode;
    }

    public void setUsercode(String usercode) {
        this.usercode = usercode;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

}

```

我们在*AServlet**类的service方法中用void setAttribute(String name, Object object)方法向ServletContext添加User对象中成员变量的数据。代码如下

```
//创建User对象，并且给成员变量赋值
User user = new User();
user.setUsercode("12335");
user.setUsername("高比例");

//给ServletContext范围存储数据
application.setAttribute("userObj",user);
```

我们在***BServlet***类的service方法中用Object getAttribute(String name)方法向ServletContext对象用name获取数据。

具体代码如下

```
//ServletContext对象范围中读取数据
Object value = application.getAttribute("usercode");
servletResponse.getWriter().print(value);//向浏览器中输出数据
```

写好后部署项目运行，进入索引页，当我们点即测试AServlet后返回页面再点击测试BServlet，可以看到如下结果输出



先点击向ServletContext范围存储数据的资源的执行结果，再点击获取数据的资源类执行结果，可以得到如下输出结果



以上操作顺序相反，当先点击下面连接获取不到数据，因为还没有存，得到如下执行结果



null