javax.servlet.ServletConfig接口：

1、这是一个接口，是由Apache Tomcat服务器实现的Servlet规范之一，Tomcat专门写了一个ServletConfig接口的实现类，其完整的类名为：org.apache.catalina.core.StandardWrapperFacade。而javax.servlet.Servlet也是一个接口，而我们就是实现该接口的人员。

这里主要的重点：--明白大家都是面向接口开发，面向接口调用的思想。

怎么理解：面向接口开发，就是实现接口，实现里面的方法，接口中的方法是公开的，固定的；面向接口调用，不管你是怎么写里面的代码，有什么方法我只需要知道接口的就行了，而且我都可以用接口类型定义的引用调用，因为接口是父类，无论有多少个实现类该引用去调都编译通过，都可以调用，但运行时调用的方法由该引用具体的实例化的实现类对象决定。

2、javaweb程序员编程，面向ServletConfig接口完成调用，不需要关系实现类。不同的webapp放到不同的Web容器所实现的类的类名是不同的，这里放进的是Tomcat服务器。

3、Tomcat服务器就是一个实现了Servlet规范和JSP规范的容器

4、ServletConfig接口中由哪些方法？

--String getInitParameter(String name)

该方法表示通过名字name这个key来获取初始化参数value，ServletConfig对象保存的是web.xml文件中的，所以获取的是该文件中初始化参数标签的value位置的值，具体获取如下测试

--Enumeration getInitParameterNames()

该方法是获取所有初始化参数的name，获取key，返回的是一个集合，测试如下

--ServletContext getServletContext()

该方法是获取一个ServletContext对象（Servlet对象的上下文），具体分析下章节

--String getServletName()

该方法是获取xml文件中<servlet-name>标签中的值，一般没啥用，测试如下

5、ServletConfig的作用

从字面以上上理解，ServletConfig就是一个Servlet对象的配置信息对象，该对象里面封装的是Servlet对象的配置信息。而Servlet对象的配置信息是放到web.xml文件中的，我们知道该文件中有个<servlet></servlet>标签，每一个这种标签里的信息都被封装到ServletConfig对象里面了，就像下面的测试一样，xml文件中有两个servlet标签，Tomcat生成了两个信息配置对象来封装。servlet标签的数量对应ServletConfig对象的数量。当我们刷新浏览器，点击不同连接，这两个信息配置对象的输出都没动，原因是init方法只执行一次。

6、业务中可能需要信息配置对象或者如果我们希望在service方法中用ServletConfig对象，

--可以在类中定义一个私有的ServletConfig类型的变量config，在init方法中将接收的信息配置对象传进去。

--这样Servlet接口中返回ServletConfig类型对象的公开的方法getServletConfig()我们就将其返回值设置为config，该方法的作用是为了如果以后这个实现类有子类，子类可以通过该方法获取其ServletConfig对象。

我们在实现Servlet接口的时候，其中的init方法需要传入一个ServletConfig类型的对象参数，由于该方法是Web容器启动，该对象由容器提供，由该容器实现servlet规范中的ServletConfig接口，下面我们实现把参数输出一下。

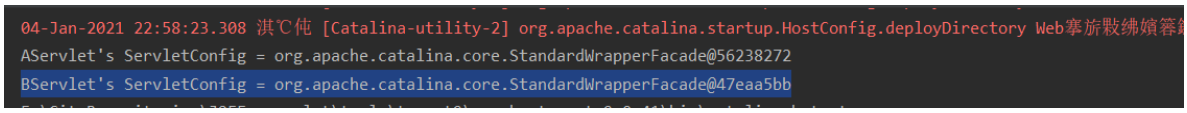
创建web项目模块b-ServletConfig，配置号相应环境，编写两个Servlet实现类AServlet和BServlet，

两个实现类中的initial方法中这样写，其他方法先不写

```
@Override
    public void init(ServletConfig servletConfig) throws ServletException {

        System.out.println("AServlet's ServletConfig =
"+servletConfig.toString());
        //AServlet's ServletConfig =
org.apache.catalina.core.StandardWrapperFacade@56238272

    }
```

```
@Override
    public void init(ServletConfig servletConfig) throws ServletException {

        System.out.println("BServlet's ServletConfig =
"+servletConfig.toString());
        //BServlet's ServletConfig =
org.apache.catalina.core.StandardWrapperFacade@47eaa5bb

    }
```

写一个index.html，里面分别是请求这两个实现类对象的超链接，启动Tomcat，分别点击两个连接输出如下：




```
AServlet's ServletConfig =
org.apache.catalina.core.StandardWrapperFacade@56238272
BServlet's ServletConfig =
org.apache.catalina.core.StandardWrapperFacade@47eaa5bb
```

将init方法中的局部变量赋值给一个Servlet实现类的私有ServletConfig类型的成员变量，利用该变量测试ServletConfig接口中每个方法的作用，这里先在AServlet的servlet标签xml文件中添加初始化参数标签以及相应信息

```
<?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">
     <!--第二个Servlet对象-->
    <servlet>
        <servlet-name>AServlet</servlet-name>
        <servlet-class>com.servlet.servletconfig.Aservlet</servlet-class>
        <!--初始化参数：被封装到ServletConfig对象中-->
        <init-param>
            <param-name>driver</param-name>
            <param-value>com.mysql.jdbc.Driver</param-value>
        </init-param>
        <init-param>
            <param-name>url</param-name>
```

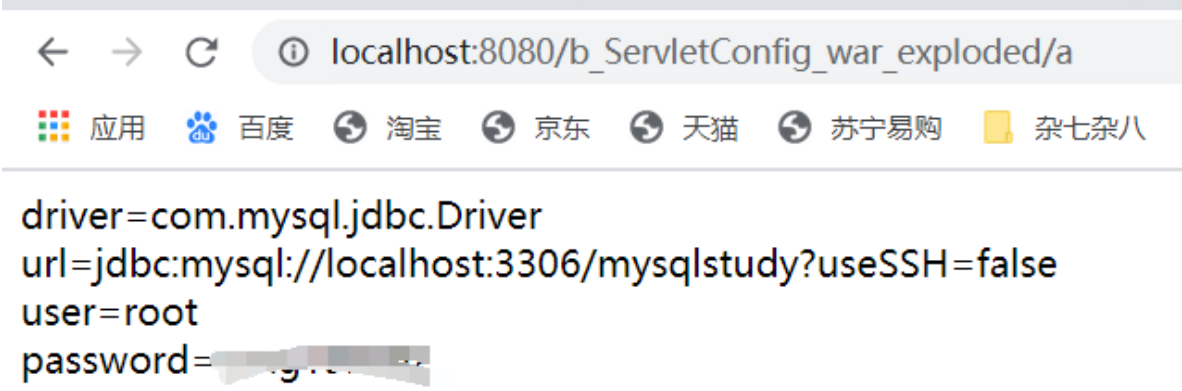
```
            <param-value>jdbc:mysql://localhost:3306/mysqlstudy?
useSSH=false</param-value>
        </init-param>
        <init-param>
            <param-name>user</param-name>
            <param-value>root</param-value>
        </init-param>
        <init-param>
            <param-name>password</param-name>
            <param-value>???</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>AServlet</servlet-name>
        <url-pattern>/a</url-pattern>
    </servlet-mapping>

    <!--第二个Servlet对象-->
    <servlet>
        <servlet-name>BServlet</servlet-name>
        <servlet-class>com.servlet.servletconfig.Bservlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>BServlet</servlet-name>
        <url-pattern>/b</url-pattern>
    </servlet-mapping>
</web-app>
```

然后AServlet类的service方法中测试 String getInitParameter(String name)方法，service方法中代码如下

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

        //更改输出格式类型
        servletResponse.setContentType("text/html;charset=UTF-8");
        //更改输出位置
        PrintWriter out = servletResponse.getWriter();

        //获取ServletConfig对象
        ServletConfig config = getServletConfig();

        //通过初始化参数的name获取value
        String driver = config.getInitParameter("driver");
        String url = config.getInitParameter("url");
        String user = config.getInitParameter("user");
        String password = config.getInitParameter("password");
        out.print("driver="+driver);
        out.print("<br>");
        out.print("url="+url);
        out.print("<br>");
        out.print("user="+user);
        out.print("<br>");
        out.print("password="+password);
```

```
    }
```

点击运行Tomcat部署项目，输入请求显示结果如下



开始测试 Enumeration getInitParameterNames()方法，代码如下

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

        //更改输出格式类型
        servletResponse.setContentType("text/html;charset=UTF-8");
        //更改输出位置
        PrintWriter out = servletResponse.getWriter();

        //获取ServletConfig对象
        ServletConfig config = getServletConfig();

        //通过初始化参数的name获取value
//        String driver = config.getInitParameter("driver");
//        String url = config.getInitParameter("url");
//        String user = config.getInitParameter("user");
//        String password = config.getInitParameter("password");
//        out.print("driver="+driver);
//        out.print("<br>");
//        out.print("url="+url);
//        out.print("<br>");
//        out.print("user="+user);
//        out.print("<br>");
//        out.print("password="+password);

        //获取初始化参数的name
        Enumeration<String> names = config.getInitParameterNames();
        //遍历该集合，并通过name调用获取value，虽然该集合对象的遍历方法名字不一样，但底层还是
一样的
        while (names.hasMoreElements()){

            String name = names.nextElement();
            String value = config.getInitParameter(name);
            out.print(name+" = "+value);
            out.print("<br>");
        }
```

```
    }
```

结果如下，可以知道存进集合中初始化参数name的顺序是随机的

password = [redacted]
driver = com.mysql.jdbc.Driver
user = root
url = jdbc:mysql://localhost:3306/mysqlstudy?useSSH=false

开始测试没啥卵用的String getServletName()方法，这里只贴出代码，

```
        //获取Servlet Name
        String servletName = config.getServletName();
        out.print("<servlet-name>"+servletName+"</servlet-name>");//AServlet

        //获取Servlet上下文的方法
        ServletContext application = config.getServletContext();

 out.print(application);//org.apache.catalina.core.ApplicationContextFacade@6b2c
87d6
```

最后一个获取ServletContext对象的方法，代码如上。结果如下

AServlet
org.apache.catalina.core.ApplicationContextFacade@5c663f13