新建一个web项目
代码文件
web.xml
application.properties
JxDispatcherServlet.java
JxAutowired.java
JxController.java
JxMapping.java
JxParam.java
JxService.java
DemoController.java
IDemoService.java
DemoServiceImpl.java
FileUtil.java

新建一个web项目

目录结构如下

```
spring_base
├── spring_base.iml
├── src
|   ├── application.properties
|   └── com
|       └── jx
|           └── base
|               ├── JxDispatcherServlet.java
|               ├── annotation
|               |   ├── JxAutowired.java
|               |   ├── JxController.java
|               |   ├── JxMapping.java
|               |   ├── JxParam.java
|               |   └── JxService.java
|               ├── controller
|               |   └── DemoController.java
|               ├── service
|               |   ├── DemoServiceImpl.java
|               |   └── IDemoService.java
|               └── util
|                   └── FileUtil.java
└── web
    ├── WEB-INF
    |   └── web.xml
    └── index.jsp

```

代码文件

### web.xml

```xml
<?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/
         version="3.1">
    <display-name>base_spring</display-name>

    <servlet>
        <servlet-name>JxDispatcherServlet</servlet-name>
         <servlet-class>com.jx.base.JxDispatcherServlet</servlet-class>
         <init-param>
             <param-name>contextConfigLocation</param-name>
             <param-value>application.properties</param-value>
         </init-param>
     </servlet>
     <servlet-mapping>
         <servlet-name>JxDispatcherServlet</servlet-name>
         <url-pattern>/*</url-pattern>
     </servlet-mapping>
 </web-app>
```

application.properties

```
scanPackage=com.jx.base
```

JxDispatcherServlet.java

```java
package com.jx.base;

import com.jx.base.annotation.*;
import com.jx.base.service.DemoServiceImpl;
import com.jx.base.service.IDemoService;
import com.jx.base.util.FileUtil;

import java.io.File;
import java.io.IOException;
 import java.io.InputStream;
 import java.lang.reflect.Field;
 import java.lang.reflect.Method;
 import java.lang.reflect.Parameter;
 import java.net.URI;
 import java.net.URL;
 import java.util.*;

 import javax.servlet.ServletConfig;
 import javax.servlet.ServletException;
 import javax.servlet.http.HttpServlet;
 import javax.servlet.http.HttpServletRequest;
 import javax.servlet.http.HttpServletResponse;
```

```java


public class JxDispatcherServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;

   // 保持application.properties配置文件中的内容
   private Properties contextConfig = new Properties();

   // 保存所有扫描后存储的类名
   private List<String> classNameList = new LinkedList<>();

   // ioc容器
   private Map<String, Object> ioc = new HashMap<>();

   // 存储所有的URL 和 Method的映射关系
   private Map<String, Method> mappingHandler = new HashMap<>();

    public JxDispatcherServlet() {
        super();
    }

   @Override
   public void init(ServletConfig config) throws ServletException {

      try {
         // 加载配置文件
         doLoadConfig(config.getInitParameter("contextConfigLocation"));

         // 扫描类文件
         doScanner(contextConfig.getProperty("scanPackage"));

         // 初始化扫描到的类，并将他们放入 ioc 容器中
         doInstance();

         // 完成 依赖注入
         doAutowired();

         // 完成 Mapper映射
         doHandlerMapping();
      } catch (Exception e) {
         e.printStackTrace();
      }

   }

   private void doHandlerMapping() {
      if (ioc.isEmpty()) {
         return;
      }

      for (Map.Entry<String, Object> entry : ioc.entrySet()) {
         Class<?> clz = entry.getValue().getClass();
         if (!clz.isAnnotationPresent(JxController.class)){
            continue;
         }

        String baseUrl = "";
        if (clz.isAnnotationPresent(JxMapping.class)){
           JxMapping mapping = clz.getAnnotation(JxMapping.class);
           baseUrl = mapping.value();
           if (!baseUrl.startsWith("/")){
              baseUrl = "/" + baseUrl;
           }
        }

        for (Method method : clz.getMethods()) {
           if (!method.isAnnotationPresent(JxMapping.class)) {
              continue;
           }
           JxMapping mapping = method.getAnnotation(JxMapping.class);
           String mappingUrl = mapping.value();
           if (!mappingUrl.startsWith("/")){
              mappingUrl = "/" + mappingUrl;
           }
           baseUrl = baseUrl + mappingUrl;

           mappingHandler.put(baseUrl, method);
         }
      }
   }

   private void doAutowired() throws IllegalAccessException {
       if (ioc.isEmpty()){
          return;
      }
      for (Map.Entry<String, Object> entry : ioc.entrySet()) {
         Field[] fields = entry.getValue().getClass().getDeclaredFields();
         for (Field field : fields) {
            if (!field.isAnnotationPresent(JxAutowired.class)) {
               continue;
            }
            JxAutowired autowired = field.getAnnotation(JxAutowired.class);
            String beanName = autowired.value();
            if ("".equals(beanName)){
               beanName = field.getType().getName();
            }

            field.setAccessible(true);

            field.set(entry.getValue(), ioc.get(beanName));
         }
      }
   }

   private void doInstance() throws Exception {
      if (classNameList.isEmpty()) {
         return;
      }
      for (String className : classNameList) {
         Class<?> clz = Class.forName(className);

         if (clz.isAnnotationPresent(JxController.class)){
            Object instance = clz.newInstance();
            String beanName = FileUtil.toLowerFirstCase(clz.getSimpleName());
            ioc.put(beanName, instance);
         }else if (clz.isAnnotationPresent(JxService.class)){
            JxService jxController = clz.getAnnotation(JxService.class);
            String beanName = jxController.value();
            if ("".equals(beanName)){
               beanName = FileUtil.toLowerFirstCase(clz.getSimpleName());
            }
            Object instance = clz.newInstance();
            ioc.put(beanName, instance);

            for (Class<?> clzInterface : clz.getInterfaces()) {
               if (ioc.containsKey(clzInterface.getName())){
                   throw new Exception("the " + clzInterface.getName() + " is exists");
               }
               ioc.put(clzInterface.getName(), instance);
            }
         }
      }
   }



   private void doLoadConfig(String contextConfigLocation) {
      try (InputStream fis = this.getClass().getClassLoader().getResourceAsStream(contextConfigLoc
         contextConfig.load(fis);
      } catch (IOException e) {
         throw new RuntimeException(e.getMessage(), e);
      }
   }

   private void doScanner(String scanPackage) {
       String basePath = scanPackage.replaceAll("\\.", "/");
      URL url = this.getClass().getClassLoader().getResource("/" + basePath);
      List<String> fileList = new LinkedList<>();
      FileUtil.getAllFileName(url.getPath(), fileList);
      fileList.forEach(f -> {
         if (f.endsWith(".class")){
            String clazz = f.substring(f.indexOf(basePath)).replace(".class", "").replaceAll("/",
            classNameList.add(clazz);
         }
      });
   }

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletEx
      this.doPost(request, response);
   }

   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletE
      try {
         this.doDispatcher(request, response);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

   private void doDispatcher(HttpServletRequest request, HttpServletResponse response) throws Exce
      String url = request.getRequestURI();
      String context = request.getContextPath();
      url = url.replace(context, "").replaceAll("/+", "/");
      if (!this.mappingHandler.containsKey(url)) {
         response.getWriter().write("404");
         return;
      }

      Method method = (Method) this.mappingHandler.get(url);

      Map<String, String[]> paramMap = request.getParameterMap();

      Object[] params = new Object[method.getParameters().length];

      // 默认情况下是获取不到参数的 名称的，只能获取到 arg1 、 arg2 、arg3
      // 可以通过字节码增强工具 asm、bytebuddy、javassit都是字节码增强的库
      // 也可在编译ide 上进行设置，  idea 的设置Preferences->Build,Execution,Deployment->Compiler->java
      for (int i = 0; i < method.getParameters().length; i++) {
         Parameter parameter = method.getParameters()[i];
         if (parameter.getType() == HttpServletRequest.class){
            params[i] = request;
         }else if (parameter.getType() == HttpServletResponse.class){
            params[i] = response;
         }else if (parameter.getType() == String.class){
            JxParam jxParam = parameter.getAnnotation(JxParam.class);
            String s = "";
            if (jxParam != null){
               for (String s1 : paramMap.get(jxParam.value())) {
                   s = s1;
                   break;
               }
            }else {
               for (String s1 : paramMap.get(parameter.getName())) {
                   s = s1;
                   break;
               }
            }
            params[i] = s;
         }
      }

      String beanName = FileUtil.toLowerFirstCase(method.getDeclaringClass().getSimpleName());
      method.invoke(ioc.get(beanName), params);
   }

```

```
}
```

JxAutowired.java

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface JxAutowired {
    String value() default "";
}
```

JxController.java

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface JxController {
    String value() default "";
}
```

JxMapping.java

```
@Target({ ElementType.TYPE, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface JxMapping {
    String value() default "";
}
```

JxParam.java

```
@Target(ElementType.PARAMETER)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface JxParam {
    String value() default "";
}
```

JxService.java

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface JxService {
    String value() default "";
}
```

DemoController.java

```
@JxController
@JxMapping("demo")
public class DemoController {

    @JxAutowired
    private IDemoService demoService;

    @JxMapping("query")
    public void query(HttpServletRequest request, HttpServletResponse response, @JxParam("name") Str
        String result = demoService.getName(name);
        try {
            response.getWriter().write(result);
        } catch (IOException e) {
```

```java
            e.printStackTrace();
        }
    }

}
```

IDemoService.java

```java
public interface IDemoService {
    String getName(String name);
}
```

DemoServiceImpl.java

```java
@JxService
public class DemoServiceImpl implements IDemoService{

    public String getName(String name) {
        return "my name is " + name;
    }

}
```

FileUtil.java

```java
public class FileUtil {

    /**
     * 获取一个文件夹下的所有文件全路径
     *
     * @param path
     * listFileName
     */
    public static void getAllFileName(String path, List<String> listFileName) {
        File file = new File(path);
        File[] files = file.listFiles();
        String[] names = file.list();
        if (names != null) {
            String[] completNames = new String[names.length];
            for (int i = 0; i < names.length; i++) {
                completNames[i] = path + names[i];
            }
            listFileName.addAll(Arrays.asList(completNames));
        }
        for (File a : files) {
            if (a.isDirectory()) {
                getAllFileName(a.getAbsolutePath() + "/", listFileName);
            }
        }
     }

     /**
      * 加字符首字母 转小写
      *
      * @param str
      * @return
      */
     public static String toLowerFirstCase(String str){
        char[] chars = str.toCharArray();
        // 大小写 Ascii 相差 32
        chars[0] += 32;
        return String.valueOf(chars);
    }
}
```