# 《面向服务的软件系统》实验指导书

## ——实验一：服务构件/服务系统的设计与实现

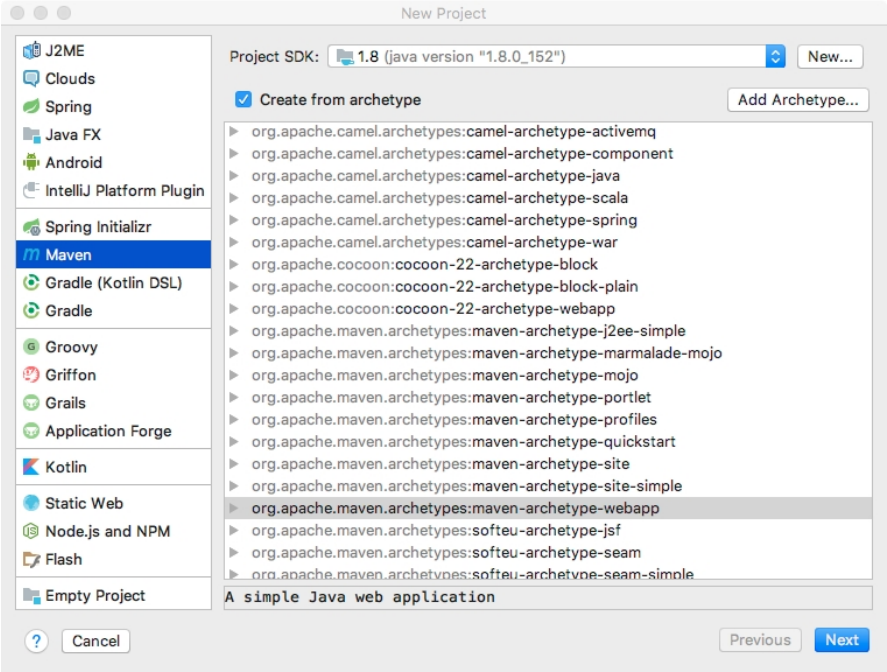
### ➢ 实验 1 基于 Apache CXF 框架的服务开发环境搭建

#### 1.1 基本开发环境准备

- 安装 IntelliJ IDEA，并注册学生账户以获得免费使用权。
- 安装 jdk1.8
- 安装 tomcat6.0 以上

#### 1.2 项目创建

- 利用 Maven 创建一个 Web 项目




- 项目 dependencies 修改

在 IDE 自动生成的 pom.xml 文件中添加一个 dependency。本次实验选用的是 Apache CXF 框架 3.2.6 版。同学们可以通过 https://mvnrepository.com/artifact/org.apache.cxf，查找最新版本使用。

```
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>apache-cxf</artifactId>
    <version>3.2.6</version>
    <type>pom</type>
</dependency>
```

配置好后，等待 Maven download 所有必须的 dependencies，其中，包括本实验后半段需要使用的 Spring 4.3.18 框架。下载完毕后，项目目录 External Libraries 中将出现一系列的项目相关的库文件，如下图所示。


1.3 服务接口创建

- 在工程目录的 src 下创建自定义的代码 package
- 随后编写服务接口程序，如下图所示的 IReaderService 接口，即定义该 web 服务的服务方法集合（PortType）。

```
@WebService
public interface IReaderService {
    public Reader getReader(@WebParam(name="name") String name, @WebParam(name="password") String password);
    public List<Reader> getReaders();
}
```

备注：接口中定义了返回值为 Reader，是一个包含了字符串类型属性 name 和 password 的类

1.4 服务接口实现

完成服务接口 IReaderService 中各个方法的逻辑具体实现。注意 @WebService 中关于 endpoint 和 serviceName 的定义。

```java
@WebService(endpointInterface= "com.hit.IReaderService",serviceName="readerService")
public class ReaderService implements IReaderService{
    public Reader getReader(@WebParam(name="name") String name, @WebParam(name="password") String password) {
        return new Reader(name,password);
    }

    public List<Reader> getReaders(){
        List<Reader> readerList = new ArrayList<Reader>();
        readerList.add(new Reader( name: "shun1", password: "123"));
        readerList.add(new Reader( name: "shun2", password: "123"));
        return readerList;
    }
}
```

1.5 服务发布程序的编写

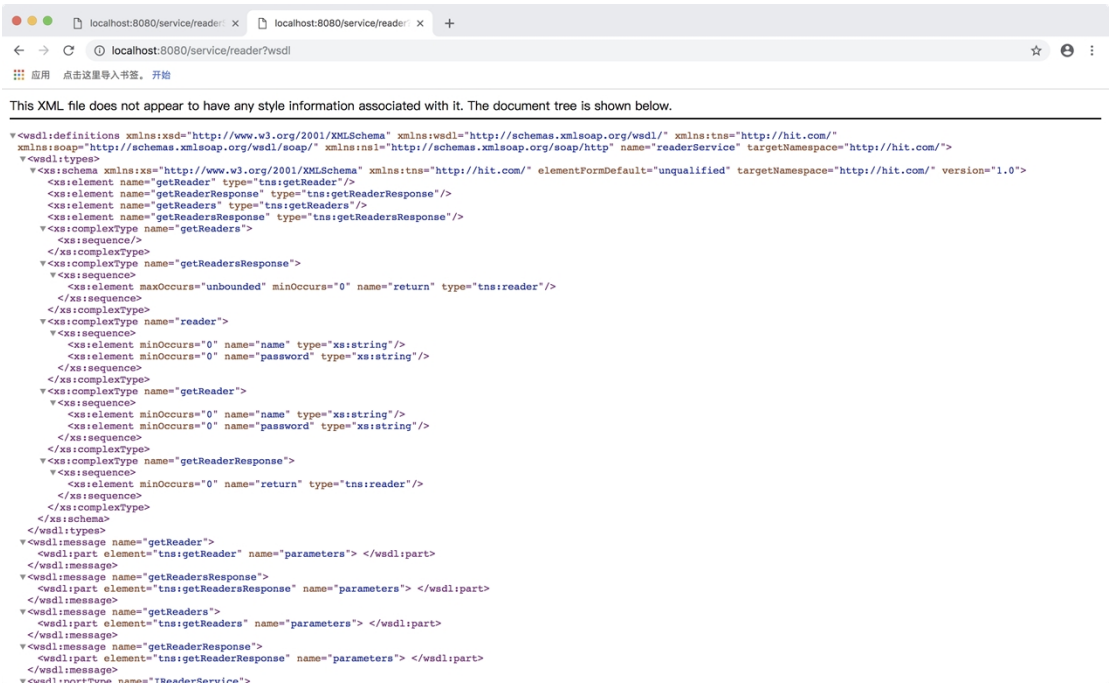
按下图的代码编写服务发布程序。

```java
public class ServiceRun {
    public static void main(String[] args) {
        JaxWsServerFactoryBean factory = new JaxWsServerFactoryBean();
        factory.setServiceClass(ReaderService.class);
        factory.setAddress("http://localhost:8080/service/reader");
        Server server = factory.create();
        server.start();
    }
}
```

编写完成后，编译并运行该程序，并在浏览器中输入

http://localhost:8080/service/reader?wsdl

如果出现下列 WSDL 描述的页面，则服务发布成功。




1.6 客户端程序编写

按照下图中代码编写代码进行测试，如果得到“[Name:shun1,Password:123, Name:shun2,Password:123]”的结果，则客户端测试成功。

```java
package com.hit;

import org.apache.cxf.jaxws.endpoint.dynamic.JaxWsDynamicClientFactory;

public class ClientRun {
    public static void main(String[] args) {
        JaxWsDynamicClientFactory dcf = JaxWsDynamicClientFactory.newInstance();
        org.apache.cxf.endpoint.Client client = dcf.createClient( wsdlUrl: "http://localhost:8080/service/reader?wsdl");
//        org.apache.cxf.endpoint.Client client = dcf.createClient("http://localhost:8080/service/readerService2?wsdl");
        Object[] objects;
        try {
//            objects = client.invoke("getReaders", "桑博");
            objects = client.invoke( s: "getReaders");
            //输出结果
            System.out.println(objects[0].toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

1.7 基于 Spring 在 Tomcat 下发布服务

- Spring 配置

在项目下 add Framework support 选项中，选定 Spring 4 的支持。由于 Apache CXF 3.2.6 中自带 Spring 4 的相关库文件，所以不用重新下载，选定已下载的库文件做支持。同时，生成 Spring 的配置文件。具体内容如下图所示：

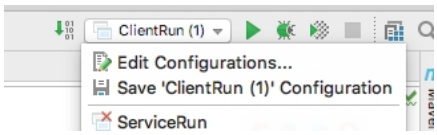
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jaxws="http://cxf.apache.org/jaxws"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://cxf.apache.org/jaxws
       http://cxf.apache.org/schemas/jaxws.xsd">

    <import resource="classpath:META-INF/cxf/cxf.xml" />
    <!--import resource="classpath:META-INF/cxf/cxf-extension-soap.xml"/-->
    <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />

    <jaxws:endpoint id="readerServicce2"
                    implementor="com.hit.ReaderService" address="/readerService2" />
</beans>
```

- web.xml 配置

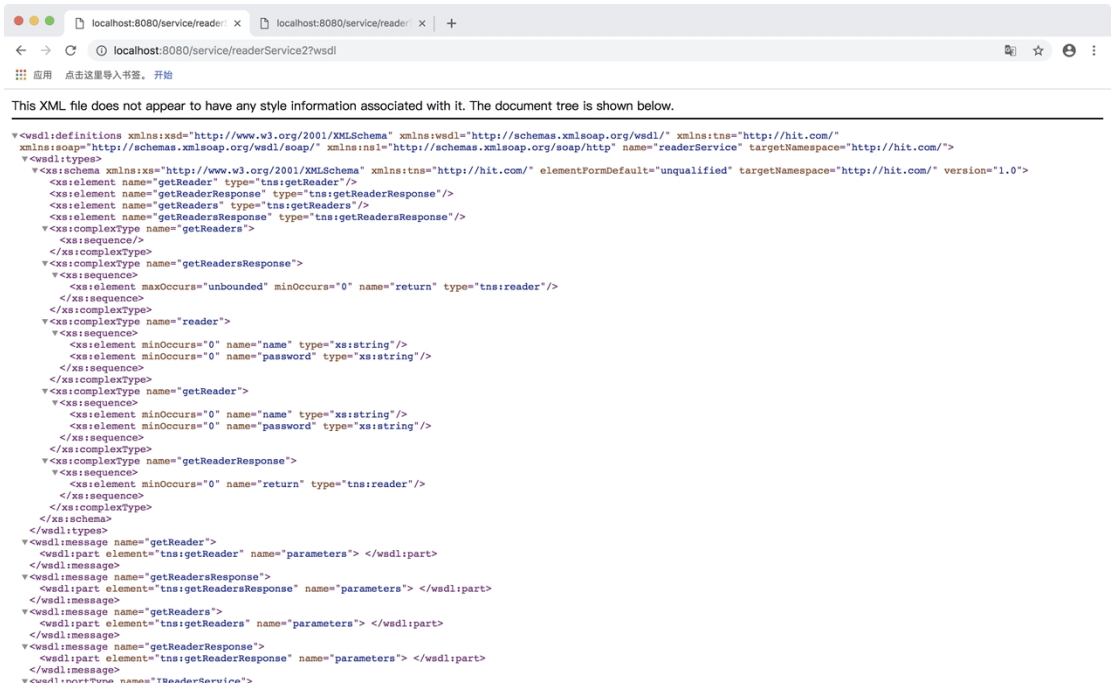
如果 IDE 没自动生成 web.xml 文件，可手动在 WEB-INF 文件夹下创建一个 web.xml 文件。具体内容配置如下图所示：

```
<!DOCTYPE web-app PUBLIC
        "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
        "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>cxfservice</display-name>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>WEB-INF/spring-config.xml</param-value>
  </context-param>
  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
  <servlet>
    <servlet-name>cxf</servlet-name>
    <servlet-class>
      org.apache.cxf.transport.servlet.CXFServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>cxf</servlet-name>
    <url-pattern>/service/*</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

- 服务下 tomcat 下的部署




在 edit configuration 中添加一个 tomcat server, 然后选择已安装好的 tomcat 作为服务器, 并将本工程的 artifacts 部署到该服务器, 并启动。如果输入以下 URL 后: http://localhost:8080/service/readerService2?wsdl, 能得到如下图所示的 wsdl 描述页面，则服务发布成功。




## 1.8 客户端程序编写

按照下图中代码编写代码进行测试，如果得到“[Name:shun1,Password:123, Name:shun2,Password:123]”的结果，则客户端测试成功。

```
package com.hit;

import org.apache.cxf.jaxws.endpoint.dynamic.JaxWsDynamicClientFactory;

public class ClientRun {
    public static void main(String[] args) {
        JaxWsDynamicClientFactory dcf = JaxWsDynamicClientFactory.newInstance();
//        org.apache.cxf.endpoint.Client client = dcf.createClient("http://localhost:8080/service/reader?wsdl");
        org.apache.cxf.endpoint.Client client = dcf.createClient( wsdlUrl: "http://localhost:8080/service/readerService2/?wsdl");
        Object[] objects;
        try {
//            objects = client.invoke("getReaders", "桑博");
            objects = client.invoke( s: "getReaders");
            //输出结果
            System.out.println(objects[0].toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

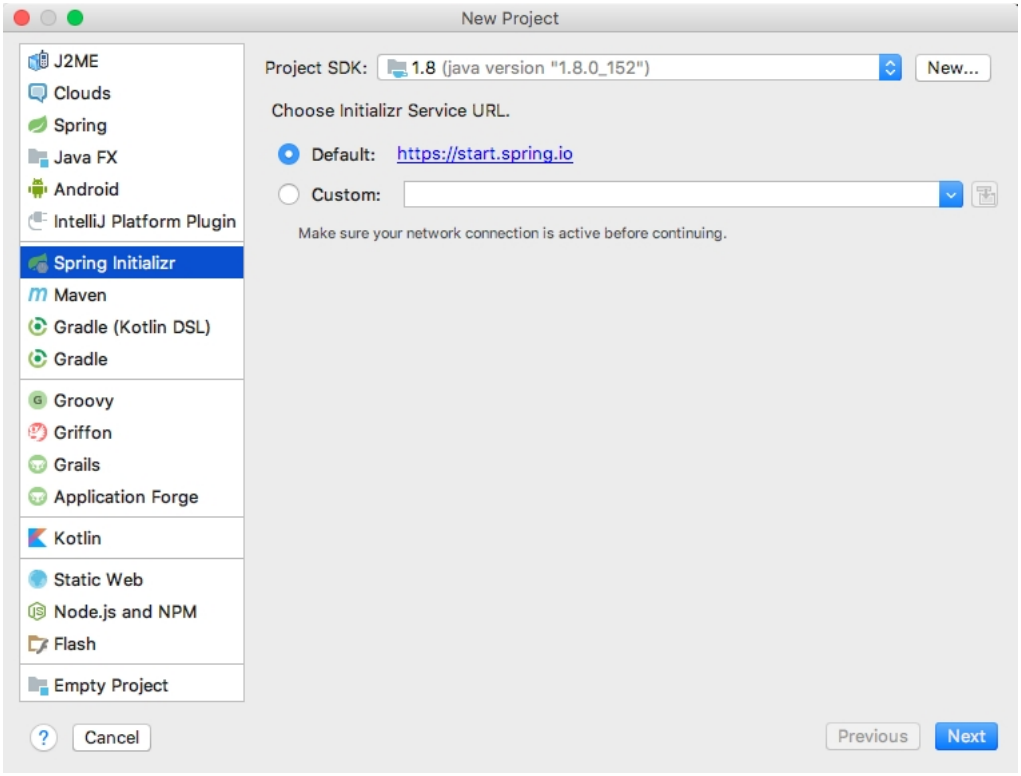
## ➢ 实验 2 基于 SpringBoot 框架的服务开发环境搭建

## 2.1 基本开发环境准备

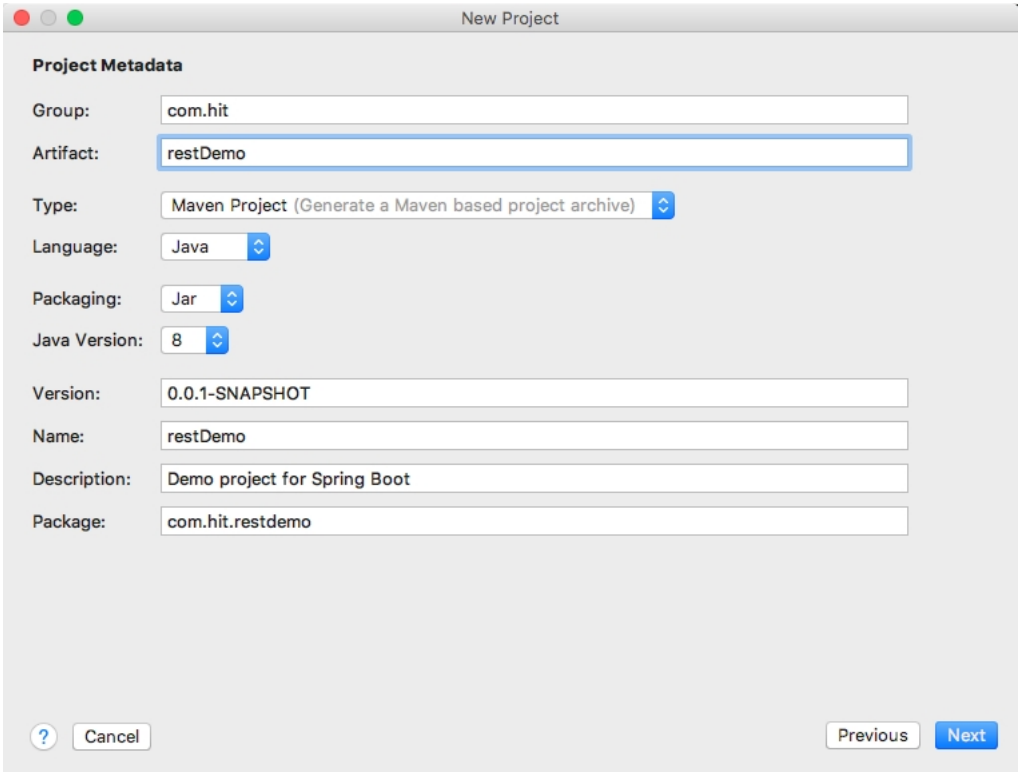
- 同 1.1

## 2.2 项目创建

- 打开 idea，选择 Create New Project
- 项目配置，选择 Spring Initializr，Project SDK 选择 1.8，URL 选择默认，不做修改。如下图所示，然后选择 Next




- 继续项目配置



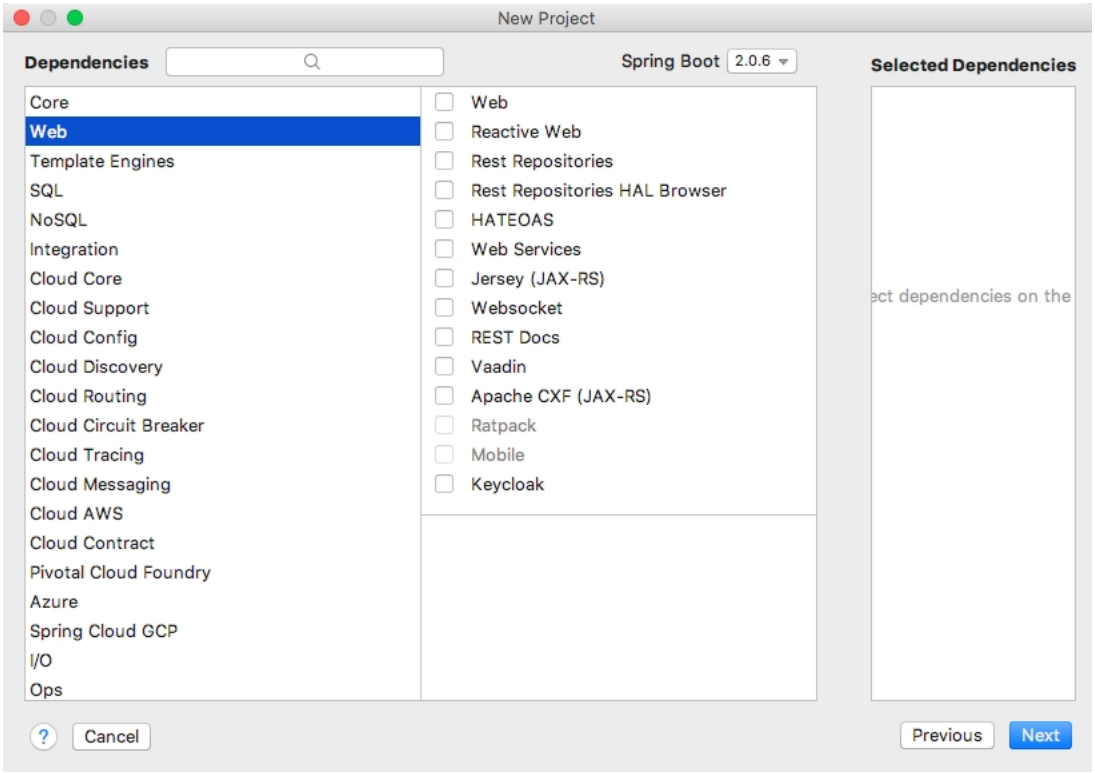

这里基本都已经自动生成了，简单介绍下：

✓ Group：对应 pom 文件中的 groupId，项目组织唯一标识，对应 Java 包的结构

✓ Artifact：对应 pom 文件的 artifactId，项目唯一标识，对应项目名称

- ✓ Type：我们是 Maven 构建的，那么选择第一个 Maven Project
- ✓ Language：开发语言，选择 Java
- ✓ Packaging：打包类型，打包成 Jar 文件
- ✓ Java Version：jdk 版本，选择 1.8/8
- ✓ Version：项目版本，对应 pom 文件的 version
- ✓ Name：项目名称
- ✓ Description：项目描述，对应 pom 文件的 description
- ✓ Package：包名

- 进一步配置：




- ✓ 选择 Spring Boot 的版本。
- ✓ 其他 dependencies 可以根据需要进行勾选，本实验基础选择 Web 下的 Web，Web Services，Apache CXF (JAX-RS)。其他选项可以课下自行进行尝试。
- ✓ 按照该配置最终创建项目之后，目录中存在.mvn 文件夹，mvnw、mvnw.cmd 文件是无关文件，可以选择删除。

- 服务程序的编写

按照下图中的代码实现服务代码编写，随后运行 DemoApplication.java//文件名字和前期配置有关系，该文件中有注解@ SpringBootApplication，表示它是 Springboot 项目的启动类。

```java
package com.hit.demo;

import org.springframework.web.bind.annotation.*;
import test.testEntity;

@RestController
public class HalloControl {
    @RequestMapping(value="/hallo", method= RequestMethod.GET)
    public String say(){return "hello!";}

    @RequestMapping(value="/hallo/{id}", method= RequestMethod.GET)
    public testEntity say1(@RequestParam(value = "qid", required = true) String qid, @PathVariable("id") String id){
        testEntity t = new testEntity(id,qid);
        return t;
    }
}
```

如果在控制台出现下图，则项目正式在 tomcat 下运行了，端口 8080，没有上下文的前置路径。

```
2018-10-21 12:56:03.406  INFO 17306 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2018-10-21 12:56:03.485  INFO 17306 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type [class org.sp
2018-10-21 12:56:03.691  INFO 17306 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.web.servlet.
2018-10-21 12:56:03.850  INFO 17306 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/hallo/{id}],methods=[GET]}" onto public test.testEntity c
2018-10-21 12:56:03.851  INFO 17306 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/hallo],methods=[GET]}" onto public java.lang.String com.h
2018-10-21 12:56:03.853  INFO 17306 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEnt
2018-10-21 12:56:03.854  INFO 17306 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframe
2018-10-21 12:56:03.874  INFO 17306 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type [class org.spring
2018-10-21 12:56:03.874  INFO 17306 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class org.springframewor
2018-10-21 12:56:04.103  INFO 17306 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2018-10-21 12:56:04.157  INFO 17306 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2018-10-21 12:56:04.161  INFO 17306 --- [           main] com.hit.demo.DemoApplication             : Started DemoApplication in 3.868 seconds (JVM running for 5.053)
```

在浏览器中输入 http://localhost:8080/hallo，如果看到浏览器中显示下图，则服务发布成功，可访问。

hello!