Dubbo

1.单体应用

1.1 所有的代码全部集中在一个war文件

1.2 后端（dao service controller） 程序不易于扩展 service 处理 代码重复量比较大

1.3 功能迭代时 技术复杂度变得越来越高

1. 0 CRM JSP+Servlet +JSP(JSTL)
2. 0 CRM SSM +Freemarker 无权限控制
3. 0 CRM SSM+Freemarker+EasyUI+权限控制
4. 0 CRM SSM+Freemarker +EasyUi+Shiro + 任务调度

   当程序出现bug 时， bug 修复难度较大 项目部署时 复杂度变高

1.4 服务不可拆分

应用于应用之间相互独立（不同的tomcat） 代码重复量较大 通常是业务代码

1.5 单位时间内处理的请求量有限

单台tomcat 并发200-250（apache httpclient-get|post|put|delete）水平扩展（nginx）

2.面向服务的架构（SOA）

2.1 服务调用方（客户端-java 应用程序） 调用服务提供的service 获取对应的处理结果

2.2 服务提供方（提供大量的service ） 实现相关service (面向接口开发)

3. 微服务架构 SpringCloud

服务粒度的拆分更加细化

4.Dubbo 概念（基于分布式项目）

一个高性能的RPC框架，用于分布式项目开发与服务治理的框架（支持的是JAVA语言）

类似的RPC框架：thrift，grpc

特点：

a.面向接口的远程调用（屏蔽内部实现细节）

b.支持不同的均衡均衡策略（RR,URL Hash 公平分发 fr）

c.支持服务自动注册与发现

d.可视化服务治理与运维

5.RPC 概念

协议:跨越传输层与应用层，封装了底层数据传输的细节，对于程序员来说，不需要关注底层实现细节

客户机（客户端）-服务器模式：当请求没有到达服务端，服务端（全天候）处于休眠状态，当请求到达时，服务端程序会被唤醒，处理客户端请求（计算，图形处理 等），将结果响应给客户端（RPC Socket）

OSI 七层网络模型：

应用层：Http (Https),ftp,smtp,pop3

表示层

会话层

传输层：tcp|udp

网络层

数据链路层

物理层

RPC协议-开源框架

Dubbo 框架 （基于JAVA语言），google （GRPC）, facebook(thrift)，小米，华为

RPC 内部运行解决的问题

a.网络通讯问题--》Socket

b.网络寻址问题--》本地存根（字典）

c.数据序列化与反序列化

RPC 简单实现-基于RMI

客户端

服务端

Maven多模块构建

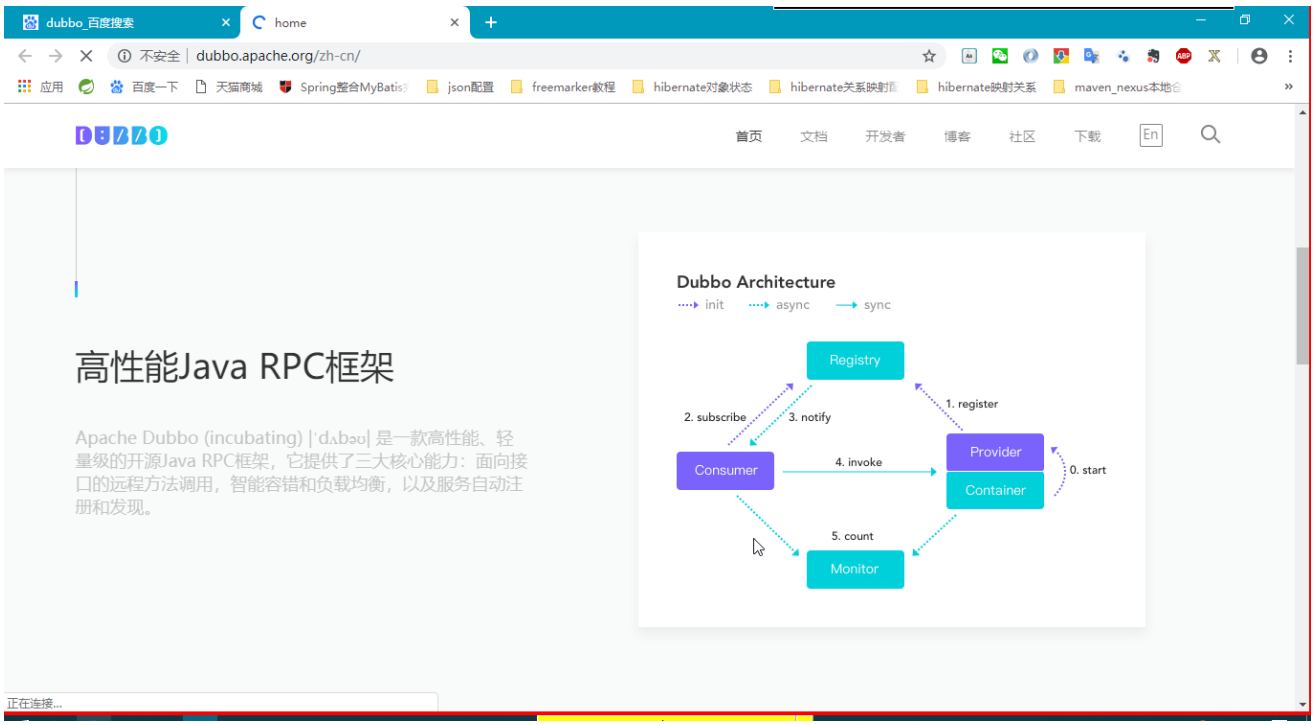
rmi_par 父工程

rmi_client 客户端

rmi_server 服务端

rmi_api api接口定义模块

6.Dubbo 流程介绍



1.服务提供方(Provider):

环境：Spring 环境

应用启动时:

服务初始化

服务注册（将服务信息写入到远程注册中心：ip：port、项目名/uri?属性参数配置&）

2.注册中心(Registry):

第三方软件实现（zookeeper） 一种树形的目录服务

2.1.保存服务提供方提供的服务信息

2.2.支持服务变更推送

3.服务消费方（Consumer）

环境：Spring 环境

应用启动时:

订阅注册中西提供的服务列表信息

执行远程服务调用（RPC）

4.监控中心（Monitor）

负责项目运行信息的监控操作（消费方运行信息 提供方运行信息）

7.Dubbo 快速入门环境配置

dubbo_par 父工程（pom）

dubbo_api 服务api定义模块(普通工程)

dubbo_provider 服务提供方(普通)

dubbo_consumer 服务消费方（普通）

8.Dubbo 常用标签配置

dubbo:application :应用名称配置（与项目名一致） （提供方与消费方）

dubbo:registry 注册中心配置 可以配置多个 一般情况配置一个 通常使用zookeeper 必须配置（提供方和消费方）

dubbo:protocol 服务注册使用协议 官方建议 dubbo 默认端口：20880 (提供方)

dubbo:service 注册服务 配置 （提供方）

dubbo:reference 服务订阅 （消费方）

9.Dubbo 注册中心

推荐使用Zookeeper 注册中心

配置步骤

zookeeper 配置

下载：http://mirror.bit.edu.cn/apache/zookeeper/zookeeper-3.4.13/ 并解压

目录：

conf:zoo_sample.cfg

单机版配置（windows 或者linux）

dataDir=C:/java/zk_3.4.13/data dataLogDir=C:/java/zk_3.4.13/log

bin: zkServer.cmd zkCli.cmd | zkServer.sh zkCli.sh

启动（windows）：

执行zkServer.cmd 启动服务端

执行zkCli.cmd 连接zookeeper 服务端

启动（windows）：

执行zkServer.cmd 启动服务端

执行zkCli.cmd 连接zookeeper 服务端

启动（linux）：

cd /home/sorfware/zookeeper-3.4.13/bin

执行./zkServer.sh start |stop|restart 启动服务端

执行./zkCli.sh 连接zookeeper 服务端

应用程序配置

添加坐标

```
<!--
         zookeeper 坐标添加
       -->
       <dependency>
           <groupId>org.apache.zookeeper</groupId>
           <artifactId>zookeeper</artifactId>
           <version>3.3.3</version>
       </dependency>

       <dependency>
           <groupId>com.github.sgroschupf</groupId>
           <artifactId>zkclient</artifactId>
           <version>0.1</version>
       </dependency>
```

服务提供方 与 消费方 xml配置

dubbo_provider.xml(dubbo_consumer.xml 同理)

```
<dubbo:registry address="zookeeper://192.168.132.10:2181"></dubbo:registry>
```

10.SSM集成Dubbo

项目基于多模块进行构建-用户信息查询为例

ssm_dubbo_par 父工程（pom 工程）

ssm_dubbo_api 服务定义模块(普通工程)

ssm_dubbo_server 服务提供方(web工程)

ssm_dubbo_admin 消费方(web工程)

集成步骤:

构建父工程

构建三个子模块（ssm_dubbo_api， ssm_dubbo_server，ssm_dubbo_admin)

添加坐标依赖

在父工程pom 中添加坐标依赖

```
    <dependencies>

        <dependency>
```

```
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>



    <!--
       ssm 基本坐标
    -->
    <!-- spring 核心jar -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.3.2.RELEASE</version>
    </dependency>

    <!-- spring 测试jar -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>4.3.2.RELEASE</version>
    </dependency>

    <!-- spring jdbc -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>4.3.2.RELEASE</version>
    </dependency>

    <!-- spring事物 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>4.3.2.RELEASE</version>
    </dependency>

    <!-- aspectj切面编程的jar -->
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>1.8.9</version>
    </dependency>


    <!-- c3p0 连接池 -->
    <dependency>
        <groupId>c3p0</groupId>
        <artifactId>c3p0</artifactId>
        <version>0.9.1.2</version>

    </dependency>
```

```
<!-- mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.1</version>
</dependency>

<!-- 添加mybatis与Spring整合的核心包 -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.0</version>
</dependency>
<!-- mysql 驱动包 -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.39</version>
</dependency>

<!-- 日志打印相关的jar -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.2</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.2</version>
</dependency>

<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper</artifactId>
    <version>4.1.0</version>
</dependency>


<!-- spring mvc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>4.3.2.RELEASE</version>
</dependency>

<!-- web servlet -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.0.1</version>

</dependency>
```

```
<!-- 添加json 依赖jar包 -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.7.0</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.7.0</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.7.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.1</version>
</dependency>
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.2</version>
</dependency>


<!--
   freemarker 坐标配置
-->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context-support </artifactId>
    <version>4.3.2.RELEASE</version>
</dependency>
<!-- Template Language -->
<dependency>
    <groupId>org.freemarker</groupId>
    <artifactId>freemarker</artifactId>
    <version>2.3.21</version>
</dependency>


<!-- https://mvnrepository.com/artifact/com.alibaba/fastjson -->
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.47</version>

</dependency>
```

```
        <!--
           dubbo 坐标配置
        -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>dubbo</artifactId>
            <version>2.5.6</version>
        </dependency>



        <!--
          zookeeper 坐标添加
        -->
        <dependency>
            <groupId>org.apache.zookeeper</groupId>
            <artifactId>zookeeper</artifactId>
            <version>3.3.3</version>
        </dependency>

        <dependency>
            <groupId>com.github.sgroschupf</groupId>
            <artifactId>zkclient</artifactId>
            <version>0.1</version>
        </dependency>

    </dependencies>
```

api 模块添加服务接口（IUserService）

```
package com.shsxt.api.service;

import com.shsxt.api.vo.User;

public interface IUserService {
    public User queryUserByUserId(Integer userId);
}
```

Server 模块

Server 代码实现

```
@Service
public class UserServiceImpl implements IUserService {
    @Autowired
    private UserDao userDao;
    @Override
    public User queryUserByUserId(Integer userId) {
        return userDao.queryUserByUserId(userId);
    }
}
```

Server 模块资源文件添加

jdbc.properties,mybatis.xml ,log4j.properties,spring.xml

以spring.xml为例

```
<?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:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd
    http://code.alibabatech.com/schema/dubbo
    http://code.alibabatech.com/schema/dubbo/dubbo.xsd">
    <context:component-scan base-package="com.shsxt.server"></context:component-scan>

    <!--
         加载jdbc.properties 文件
      -->
    <context:property-placeholder location="classpath:jdbc.properties"></context:property-
placeholder>
    <!--
         数据源c3p0
        -->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="${jdbc.driver}"></property>
        <property name="jdbcUrl" value="${jdbc.url}"></property>
        <property name="user" value="${jdbc.user}"></property>
        <property name="password" value="${jdbc.password}"></property>
    </bean>

    <!--
     开启aop 环境
```

```
    -->
      <aop:aspectj-autoproxy/>

     <!--
      事物管理器
    -->
     <bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
         <property name="dataSource" ref="dataSource"></property>
     </bean>

     <!--
          配置事物通知
        -->
     <tx:advice id="txAdvice" transaction-manager="txManager">
         <tx:attributes>
             <!--
               对更新方法进行增强:引入事物控制
             -->
             <tx:method name="save*" propagation="REQUIRED"/>
             <tx:method name="add*" propagation="REQUIRED"/>
             <tx:method name="update*" propagation="REQUIRED"/>
             <tx:method name="del*" propagation="REQUIRED"/>
             <tx:method name="query*" read-only="true"></tx:method>
             <tx:method name="get*" read-only="true"></tx:method>
         </tx:attributes>
     </tx:advice>



     <!--
       aop 基本配置
     -->
     <aop:config>
         <!--
           通常配置的拦截规则:service 方法
         -->
         <aop:pointcut id="cut" expression="execution (* com.shsxt.server.service..*.*(..))">
</aop:pointcut>
         <aop:advisor advice-ref="txAdvice" pointcut-ref="cut"></aop:advisor>
     </aop:config>


     <!-- 配置 sqlSessionFactory-->
     <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
         <property name="dataSource" ref="dataSource"></property>
         <property name="configLocation" value="classpath:mybatis.xml" />
         <property name="mapperLocations" value="classpath:com/shsxt/server/db/mappers/*.xml" />
     </bean>

     <bean id="mapperScanner" class="org.mybatis.spring.mapper.MapperScannerConfigurer">
         <!-- 扫描com.shsxthsxt.dao这个包以及它的子包下的所有映射接口类 -->

         <property name="basePackage" value="com.shsxt.server.db.dao" />
```

```
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />
    </bean>



    <!--
       dubbo 配置
    -->
    <dubbo:application name="ssm_dubbo_server"></dubbo:application>
    <dubbo:registry address="zookeeper://192.168.132.10:2181"></dubbo:registry>
    <dubbo:protocol name="dubbo" port="20880"></dubbo:protocol>
    <dubbo:service interface="com.shsxt.api.service.IUserService" ref="userServiceImpl">
</dubbo:service>

</beans>
```

程序执行入口文件 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="3.0"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring.xml</param-value>
  </context-param>
  <!-- 启用spring容器环境上下文监听 -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
</web-app>
```

配置maven 执行命令

1.安装命令

clean compile install -Dmaven.test.skip=true

2.启动项目命令

jetty:run -Djetty.port=9091

admin 模块

controller 核心代码

```
@RestController
public class UserController {
    @Resource
    private IUserService userService;

    @GetMapping("user/{userId}")
    public User queryUserByUserId(@PathVariable Integer userId){
        return userService.queryUserByUserId(userId);
    }
}
```

资源文件配置

log4j.properties,servlet-context.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:mvc="http://www.springframework.org/schema/mvc"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
       xsi:schemaLocation="
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd
        http://code.alibabatech.com/schema/dubbo
        http://code.alibabatech.com/schema/dubbo/dubbo.xsd">
    <!-- 扫描com.shsxt.crm 下包 -->
    <context:component-scan base-package="com.shsxt.admin" />
    <aop:aspectj-autoproxy />
    <bean id="freemarkerConfig"
          class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">
        <property name="templateLoaderPath" value="/WEB-INF/views/" />
        <property name="defaultEncoding" value="UTF-8" />
    </bean>
    <bean
            class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver">
        <property name="prefix" value="" />
        <property name="suffix" value=".ftl" />
        <property name="contentType" value="text/html;charset=UTF-8" />
    </bean>

    <!-- mvc 注解驱动 并添加json 支持 -->
    <mvc:annotation-driven>
        <mvc:message-converters>
```

```
            <!-- 返回信息为字符串时 处理 -->
            <bean class="org.springframework.http.converter.StringHttpMessageConverter"></bean>
            <!-- 将对象转换为json 对象 -->
            <bean

 class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter"></bean>
        </mvc:message-converters>
    </mvc:annotation-driven>

    <!-- 静态资源文件的处理放行 配置方式一 -->
    <mvc:default-servlet-handler />



    <!--
       dubbo 配置
    -->
    <dubbo:application name="ssm_dubbo_admin"></dubbo:application>
    <dubbo:registry address="zookeeper://192.168.132.10:2181"></dubbo:registry>
    <dubbo:reference interface="com.shsxt.api.service.IUserService" id="userService">
</dubbo:reference>


</beans>
```

核心 web.xml配置

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="3.0"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

  <!-- 编码过滤 utf-8 -->
  <filter>
    <description>char encoding filter</description>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>UTF-8</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>



  <servlet>
```

```
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:servlet-context.xml</param-value>
    </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

</web-app>
```

配置启动命令

jetty:run -Djetty.port=9092

启动并测试（zookeeper 先启动）

1.启动server 模块

2.启动admin 模块

测试最终结果：