Programación Distribuida

Spring Framework - Remoting

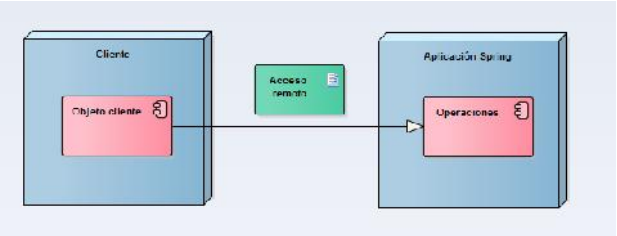
JAIME SALVADOR

AGENDA

- **Introducción**
- RMI
- HTTP Invoker
- Hessian, Burlap
- Web Services
- AMQP/JMS

Introducción

Spring soporta acceso a los servicios a través de clientes remotos



Introducción

Spring soporta acceso a los servicios a través de clientes remotos

- Métodos estándar
  - RMI
  - Web services
  - AMQP, JMS

Introducción

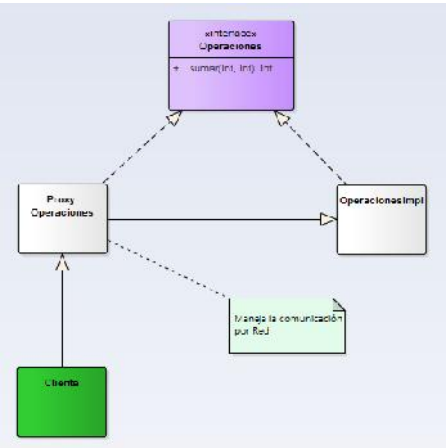
Spring soporta acceso a los servicios a través de clientes remotos

- Métodos propietarios
  - HTTP Invoker
  - Hessian, Burlap

Introducción

- El esquema general para acceso desde clientes remotos es:

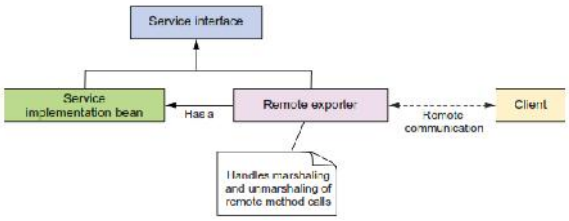
Proxy Factory Bean



Introducción

Los componentes Spring pueden ser exportados utilizando *Remote Exporters*

- El cliente se comunica con el Proxy
- No es necesario escribir código Java adicional



Introducción

Para las secciones siguientes se utilizará el ejemplo del servicios Operaciones

Interface

```
package com.distribuida.rmi.servicios;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ServicioOperaciones extends Remote {

    public int sumar( int n1, int n2 ) throws RemoteException;
}
```

Introducción

Para las secciones siguientes se utilizará el ejemplo del servicios Operaciones

Interface

```
package com.distribuida.rmi.servicios;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ServicioOperaciones extends Remote {

    public int sumar( int n1, int n2 ) throws RemoteException;
}
```

Implementación

```
package com.distribuida.rmi.servicios;

import org.springframework.stereotype.Component;

@Component(value="servicioOp")
public class ServicioOperacionesImpl implements ServicioOperaciones {

    public int sumar( int n1, int n2 ) {

        return n1 + n2;
    }
}
```

Introducción

Para las secciones siguientes se utilizará el ejemplo del servicios Operaciones

Interface

```
package com.distribuida.rmi.servicios;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ServicioOperaciones extends Remote {

    public int sumar( int n1, int n2 ) throws RemoteException;
}
```

Implementación

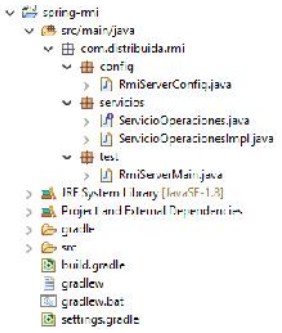
```
package com.distribuida.rmi.servicios;

import org.springframework.stereotype.Component;

@Component(value="servicioOp")
public class ServicioOperacionesImpl implements ServicioOperaciones {

    public int sumar( int n1, int n2 ) {

        return n1 + n2;
    }
}
```

Configuración

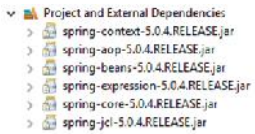
```
@Configuration
@ComponentScan(basePackages="com.distribuida.rmi.servicios")
public class RmiServerConfig {
```

Introducción

Para las secciones siguientes se utilizará el ejemplo del servicios Operaciones



Esquema del proyecto



AGENDA

- Introducción
- **RMI**
- HTTP Invoker
- Hessian, Burlap
- Web Services
- AMQP/JMS

RMI

- Utilizado cuando el cliente remote está escrito en Java
- Si el cliente NO es una aplicación Spring, el servicio debe seguir las reglas de RMI

```
package com.distribuida.rmi.servicios;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ServicioOperaciones extends Remote {

    public int sumar( int n1, int n2 ) throws RemoteException;
}
```

RMI

- Para exporter un servicio Spring a través de RMI se debe registrar un nuevo servicio

```
package com.distribuida.rmi.servicios;

import org.springframework.stereotype.Component;

@Component(value="servicioOp")
public class ServicioOperacionesImpl implements ServicioOperaciones {

    public int sumar( int n1, int n2 ) {

        return n1 + n2;
    }
}
```

```
@Bean(name="operacionesRmi")
public RmiServiceExporter operaciones( ServicioOperaciones servicioOperaciones ) {

    RmiServiceExporter exporter = new RmiServiceExporter( );

    exporter.setServiceName( "Operaciones" );
    exporter.setService( servicioOperaciones );
    exporter.setServiceInterface( ServicioOperaciones.class );

    return exporter;
}
```

```
SERVICE_NAME = "rmi://localhost:1099/Operaciones"
```

RMI

Cliente RMI

```
package com.distribuida.rmi.test;

import java.rmi.Naming;

import com.distribuida.rmi.servicios.ServicioOperaciones;

public class RmiClientMain {

    public static final String SERVICE_NAME = "rmi://localhost:1099/Operaciones";

    public static void main( String args[] ) throws Exception {

        ServicioOperaciones servicio = (ServicioOperaciones )Naming.lookup( SERVICE_NAME );

        System.out.println( "servicio: " + servicio.getClass() );

        int respuesta = servicio.sumar( 1, 3 );

        System.out.println( "respuesta: " + respuesta );
    }
}
```

RMI

Cliente Spring

- Acceso al servicio como un component Spring

```
public static final String SERVICE_NAME = "rmi://localhost:1099/Operaciones";

@Configuration
public static class RmiClientConfig{

    @Bean
    public RmiProxyFactoryBean operaciones( ) {

        RmiProxyFactoryBean proxy = new RmiProxyFactoryBean( );

        proxy.setServiceUrl( SERVICE_NAME );

        proxy.setServiceInterface( ServicioOperaciones.class );

        return proxy;
    }
}
```

RMI

Cliente Spring

- Acceso al servicio como un component Spring

```
public static void main( String args[] ) throws IOException {
    AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext( RmiClientConfig.class );
    ServicioOperaciones servicio = context.getBean( ServicioOperaciones.class );
    System.out.println( "servicio: " + servicio.getClass() );
    int respuesta = servicio.sumar( 1, 3 );
    System.out.println( "respuesta: " + respuesta );
    context.close( );
}
```

RMI

Cliente Spring

- Ejecución

```
abr 28, 2018 9:13:15 AM org.springframework.context.support.AbstractApplicationContext prepareRefresh
[illegible]
abr 28, 2018 9:13:16 AM org.springframework.remoting.rmi.RmiServiceExporter getRegistry
INFORMACIÓN: Looking for RMI registry at port '1099'
abr 28, 2018 9:13:17 AM org.springframework.remoting.rmi.RmiServiceExporter getRegistry
INFORMACIÓN: Could not detect RMI registry - creating new one
abr 28, 2018 9:13:17 AM org.springframework.remoting.rmi.RmiServiceExporter prepare
[illegible]
Servidor RMI iniciado...

<terminated> RmiClientMain [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (28 abr. 2018 09:14:27)
servicio: class com.sun.proxy.$Proxy0
respuesta: 4
```

AGENDA

- Introducción
- RMI
- **HTTP Invoker**
- Hessian, Burlap
- Web Services
- AMQP/JMS

HTTP Invoker

- Proporciona acceso remote a servicios Spring a través de HTTP
- Protocolo propietario
- La comunicación se la hace a través de un Servlet *DispatcherServlet*

HTTP Invoker

web.xml

```
<servlet>
    <servlet-name>remoting</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load on startup>1</load on startup>
</servlet>

<servlet mapping>
    <servlet-name>remoting</servlet-name>
    <url-pattern>/remoting/*</url-pattern>
</servlet mapping>
```

- WEB-INF/lib/
  spring-webmvc-5.0.5.RELEASE.jar
- WEB-INF/remoting-serlvet.xml

HTTP Invoker

Configuración Java

- WEB-INF/lib/
  spring-webmvc-5.0.5.RELEASE.jar
- WEB-INF/remoting-serlvet.xml

HTTP Invoker

- Para exporter un servicio Spring a través de HTTP se debe registrar un nuevo servicio (`remoting-servlet.xml`)

```
<bean name="/OperacionesServiceHttp" class="org.springframework.remoting.httpinvoker.HttpInvokerServiceExporter">
    <property name="service" ref="servicioOperaciones"/>
    <property name="serviceInterface" value="spring03.servicios.Operaciones"/>
</bean>
```

- La ruta para accede al servicio será:

  http://localhost:8080/spring-03/remoting/OperacionesServiceHttp

HTTP Invoker

Cliente:

- El cliente es una aplicación que utiliza Spring

`spring03-client-conf.xml`

```
<bean id="httpInvokerProxy" class="org.springframework.remoting.httpinvoker.HttpInvokerProxyFactoryBean">
    <property name="serviceUrl" value="http://localhost:8080/spring-03/remoting/OperacionesServiceHttp"/>
    <property name="serviceInterface" value="spring03.interfaces.Operaciones"/>
</bean>
```

HTTP Invoker

Cliente:

- El cliente es una aplicación que utiliza Spring

```
public class ClienteHttp {

    @SuppressWarnings("resource")
    public static void main(String[] args) throws Exception {

        ApplicationContext context = new ClassPathXmlApplicationContext( "spring03-client-conf.xml" );

        Operaciones servicio = context.getBean( "httpInvokerProxy", Operaciones.class );

        int respuesta = servicio.sumar( 7, 5 );

        System.out.println( "Respuesta: " + respuesta );
    }
}
```

AGENDA

- Introducción
- RMI
- HTTP Invoker
- **Hessian, Burlap**
- Web Services
- AMQP/JMS

Hessian, Burlap

- Proporciona acceso remoto a servicios Spring a través de Hessian, Burlap

  http://hessian.caucho.com/

- Protocolo propietario
- La comunicación se la hace a través de un Servlet *DispatcherServlet*

Hessian, Burlap

`web.xml`

```
<servlet>
    <servlet-name>ranting</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load on startup>1</load on startup>
</servlet>

<servlet mapping>
    <servlet-name>ranting</servlet-name>
    <url-pattern>/ranting/*</url-pattern>
</servlet mapping>
```

- `WEB-INF/lib/`
  - `spring-webmvc-4.1.6.RELEASE.jar`
  - `hessian-4.0.37.jar`
- `WEB-INF/remoting-serlvet.xml`

Hessian, Burlap

- Para exporter un servicio Spring a través de Hessian se debe registrar un nuevo servicio (`remoting-servlet.xml`)

```
<bean name="/OperacionesServiceHessian" class="org.springframework.remoting.caucho.HessianServiceExporter">
    <property name="service" ref="servicioOperaciones"/>
    <property name="serviceInterface" value="spring03.servicios.Operaciones"/>
</bean>
```

- La ruta para accede al servicio será:

  http://localhost:8080/spring-03/remoting/OperacionesServiceHessian

Hessian, Burlap

Cliente:

- `hessian-4.0.37.jar`

```
import com.caucho.hessian.client.HessianProxyFactory;

public class ClienteHessian {

    public static void main(String[] args) throws Exception {
        String url = "http://localhost:8080//spring-03/remoting/OperacionesServiceHessian";

        HessianProxyFactory factory = new HessianProxyFactory();

        Operaciones servicio = (Operaciones ) factory.create( Operaciones.class, url );

        int respuesta = servicio.sumar( 5, 8 );

        System.out.println( "Resultado: " + respuesta );

    }
}
```

Hessian, Burlap

Cliente Spring:

- El cliente es una aplicación que utiliza Spring

```
<bean id="hessianInvokerProxy" class="org.springframework.remoting.caucho.HessianProxyFactoryBean">
    <property name="serviceUrl" value="http://localhost:8080//spring-03/remoting/OperacionesServiceHessian"/>
    <property name="serviceInterface" value="spring03.interfaces.Operaciones"/>
</bean>
```

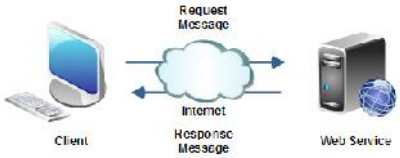
```
public class ClienteHessian2 {

    @SuppressWarnings("resource")
    public static void main(String[] args) throws Exception {

        ApplicationContext context = new ClassPathXmlApplicationContext( "spring03-client-conf.xml" );

        Operaciones servicio = context.getBean( "hessianInvokerProxy", Operaciones.class );

        int respuesta = servicio.sumar( 7, 5 );

        System.out.println( "Respuesta: " + respuesta );
    }
}
```

AGENDA

- Introducción
- RMI
- HTTP Invoker
- Hessian, Burlap
- **Web Services**
- AMQP/JMS

Web Services

TEMA 03



AGENDA

- Introducción
- RMI
- HTTP Invoker
- Hessian, Burlap
- Web Services
- **AMQP/JMS**

AMQP/JMS

TEMAS 2.7 y 2.8

RESUMEN

EXPORTAR

- RMI

  `org.springframework.remoting.rmi.RmiServiceExporter`

- HTTP

  `org.springframework.remoting.httpinvoker.`
  `HttpInvokerServiceExporter`

- Hessian

  `org.springframework.remoting.caucho.HessianServiceExporter`

EXPORTAR

CONSUMIR

- RMI

  `org.springframework.remoting.rmi.RmiProxyFactoryBean`

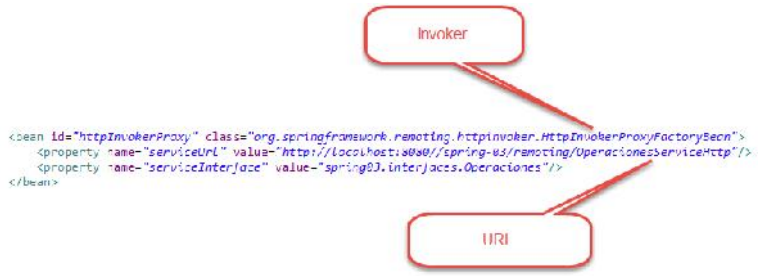
- HTTP

  `org.springframework.remoting.httpinvoker.`
  `HttpInvokerProxyFactoryBean`

- Hessian

  `org.springframework.remoting.caucho.HessianProxyFactoryBean`

CONSUMIR



PREGUNTAS?

Bibliografía

- Craig Walls. Spring in Action, Fourth Edition. Manning, 2015.
- Chros Schaefer, Clarence Ho, and Rob Harrop. Pro Spring. Apress, 2014.
- http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/, consultada el 2015-05-17,