# ROS JAVA

## 订阅者与发布者(Publishers and subscribers)

Publisher:

**final** Publisher<std\_msgs.String> publisher = connectedNode.newPublisher(**this**.topic\_name, **"std\_msgs/String"**);  
connectedNode.executeCancellableLoop(**new** CancellableLoop() {  
 **private int** sequenceNumber;  
  
 **protected void** setup() {  
 **this**.sequenceNumber = 0;  
 }  
  
 **protected void** loop() **throws** InterruptedException {  
 std\_msgs.String str = (std\_msgs.String)publisher.newMessage();  
 str.setData(**"Hello world! "** + **this**.sequenceNumber);  
 publisher.publish(str);  
 ++**this**.sequenceNumber;  
 Thread.sleep(1000L);  
 }  
});

示例demo在roscore中的android\_tutorial\_pubsub中，参考class Talker

Listener:

Subscriber<std\_msgs.String> subscriber = connectedNode.newSubscriber(**"chatter"**, std\_msgs.String.***\_TYPE***);  
subscriber.addMessageListener(**new** MessageListener<std\_msgs.String>() {  
 @Override  
 **public void** onNewMessage(std\_msgs.String message) {  
 Log.*d*(**"Listener"**, **"I heard \""** + message.getData() + **"\""**);  
 }  
});

实现示例：

Listener.java

**package** com.github.rosjava.android\_apps.teleop;  
  
**import** org.ros.message.MessageListener;  
**import** org.ros.namespace.GraphName;  
**import** org.ros.node.AbstractNodeMain;  
**import** org.ros.node.ConnectedNode;  
**import** org.ros.node.NodeMain;  
**import** org.ros.node.topic.Subscriber;  
**import** android.util.Log;  
  
**public class** Listener **extends** AbstractNodeMain {  
  
 @Override  
 **public** GraphName getDefaultNodeName() {  
 **return** GraphName.*of*(**"rosjava\_tutorial\_pubsub/listener"**);  
 }  
  
 @Override  
 **public void** onStart(ConnectedNode connectedNode) {  
 Subscriber<std\_msgs.String> subscriber = connectedNode.newSubscriber(**"chatter"**, std\_msgs.String.***\_TYPE***);  
 subscriber.addMessageListener(**new** MessageListener<std\_msgs.String>() {  
 @Override  
 **public void** onNewMessage(std\_msgs.String message) {  
 Log.*d*(**"Listener"**, **"I heard \""** + message.getData() + **"\""**);  
 }  
 });  
 }  
}

RootActivity.java

**package** com.github.rosjava.android\_apps.teleop;  
  
**import** android.os.Bundle;  
**import** android.widget.Button;  
**import** com.github.rosjava.android\_remocons.common\_tools.apps.RosAppActivity;  
**import** org.ros.node.NodeConfiguration;  
**import** org.ros.node.NodeMainExecutor;  
  
**public class** RootActivity **extends** RosAppActivity{  
 **private** Button **btn1**;  
 **private** NodeMainExecutor **nodeMainExecutor**;  
  
 **public** RootActivity() {  
 *// The RosActivity constructor configures the notification title and ticker messages.* **super**(**"android teleop"**, **"android teleop"**);  
 }  
  
 @Override  
 **public void** onCreate(Bundle savedInstanceState) {  
 setMainWindowResource(R.layout.***root***);  
 **super**.onCreate(savedInstanceState);  
 }  
 @Override  
 **protected void** init(NodeMainExecutor nodeMainExecutor) {  
 *//super.init(nodeMainExecutor);* **this**.**nodeMainExecutor** = nodeMainExecutor;  
  
 Listener listener = **new** Listener();  
 NodeConfiguration nodeConfiguration = NodeConfiguration.*newPublic*(getRosHostname());  
 nodeConfiguration.setMasterUri(getMasterUri());  
 nodeMainExecutor.execute(listener, nodeConfiguration);  
 }  
}

## 2.Server & Client

Server 和 Client实现和订阅者与发布者的实现很类似，只需要实现一个AbstractNodeMain

节点即可，同样是通过nodeMainExecutor.execute去执行

Server.java

**package** org.ros.android.android\_tutorial\_service;  
  
**import** org.ros.namespace.GraphName;  
**import** org.ros.node.AbstractNodeMain;  
**import** org.ros.node.ConnectedNode;  
**import** org.ros.node.NodeMain;  
**import** org.ros.node.service.ServiceResponseBuilder;  
**import** org.ros.node.service.ServiceServer;  
**import** android.util.Log;  
  
**public class** Server **extends** AbstractNodeMain {  
  
 @Override  
 **public** GraphName getDefaultNodeName() {  
 **return** GraphName.*of*(**"rosjava\_tutorial\_services/server"**);  
 }  
  
 @Override  
 **public void** onStart(ConnectedNode connectedNode) {  
 Log.*d*(**"###########"**, **"create service server"**);  
 connectedNode.newServiceServer(**"add\_two\_ints"**, rosjava\_test\_msgs.AddTwoInts.***\_TYPE***,  
 **new** ServiceResponseBuilder<rosjava\_test\_msgs.AddTwoIntsRequest, rosjava\_test\_msgs.AddTwoIntsResponse>() {  
 @Override  
 **public void** build(rosjava\_test\_msgs.AddTwoIntsRequest request, rosjava\_test\_msgs.AddTwoIntsResponse response) {  
 response.setSum(request.getA() + request.getB());  
 Log.*d*(**"###########"**, **"add two ints"**);  
 }  
 });  
 }  
}

Client.java

**package** com.github.rosjava.android\_apps.listener;  
  
**import** org.ros.exception.RemoteException;  
**import** org.ros.exception.RosRuntimeException;  
**import** org.ros.exception.ServiceNotFoundException;  
**import** org.ros.namespace.GraphName;  
**import** org.ros.node.AbstractNodeMain;  
**import** org.ros.node.ConnectedNode;  
**import** org.ros.node.NodeMain;  
**import** org.ros.node.service.ServiceClient;  
**import** org.ros.node.service.ServiceResponseListener;  
**import** android.util.Log;  
  
**public class** Client **extends** AbstractNodeMain {  
 @Override  
 **public** GraphName getDefaultNodeName() {  
 **return** GraphName.*of*(**"rosjava\_tutorial\_services/client"**);  
 }  
  
 @Override  
 **public void** onStart(**final** ConnectedNode connectedNode) {  
 ServiceClient<rosjava\_test\_msgs.AddTwoIntsRequest, rosjava\_test\_msgs.AddTwoIntsResponse> serviceClient;  
 **try** {  
 serviceClient = connectedNode.newServiceClient(**"add\_two\_ints"**, rosjava\_test\_msgs.AddTwoInts.***\_TYPE***);  
 } **catch** (ServiceNotFoundException e) {  
 **throw new** RosRuntimeException(e);  
 }  
 **final** rosjava\_test\_msgs.AddTwoIntsRequest request = serviceClient.newMessage();  
 request.setA(2);  
 request.setB(2);  
 serviceClient.call(request, **new** ServiceResponseListener<rosjava\_test\_msgs.AddTwoIntsResponse>() {  
 @Override  
 **public void** onSuccess(rosjava\_test\_msgs.AddTwoIntsResponse response) {  
 connectedNode.getLog().info(  
 String.*format*(**"%d + %d = %d"**, request.getA(), request.getB(), response.getSum()));  
 Log.*d*(**"client"**, **"get request info, a = "** + request.getA() + **"b = "** + request.getB());  
 }  
  
 @Override  
 **public void** onFailure(RemoteException e) {  
 **throw new** RosRuntimeException(e);  
 }  
 });  
 }  
}

## 3.使用自定义消息实现Server & Client之间通信

实现接口：

在src/main/java目录下创建文件夹rosjava\_test\_msgs, 创建三个文件AddTest.java AddTestRequest.java AddTestResponse.java

AddTest.java

**package** rosjava\_test\_msgs;  
  
**import** org.ros.internal.message.Message;  
  
**public interface** AddTest **extends** Message{  
 String ***\_TYPE*** = **"rosjava\_test\_msgs/AddTest"**;  
 String ***\_DEFINITION*** = **"int64 a\nint64 b\n---\nint64 sum\n"**;  
}

AddTestRequest.java

**package** rosjava\_test\_msgs;  
  
**import** org.ros.internal.message.Message;  
  
**public interface** AddTestRequest **extends** Message{  
 String ***\_TYPE*** = **"rosjava\_test\_msgs/AddTestRequest"**;  
 String ***\_DEFINITION*** = **"int64 a\nint64 b\n"**;  
  
 **long** getA();  
  
 **void** setA(**long** var1);  
  
 **long** getB();  
  
 **void** setB(**long** var1);  
}

AddTestResponse.java

**package** rosjava\_test\_msgs;  
  
**import** org.ros.internal.message.Message;  
  
**public interface** AddTestResponse **extends** Message{  
 String ***\_TYPE*** = **"rosjava\_test\_msgs/AddTestResponse"**;  
 String ***\_DEFINITION*** = **"int64 sum"**;  
  
 **long** getSum();  
  
 **void** setSum(**long** var1);  
}

上述的文件在android\_core和android\_apps中都需要创建，两边需要完全对应起来，下面是Server和Client的实现方式

Server.java

**package** org.ros.android.android\_tutorial\_service;  
  
**import** org.ros.namespace.GraphName;  
**import** org.ros.node.AbstractNodeMain;  
**import** org.ros.node.ConnectedNode;  
**import** org.ros.node.NodeMain;  
**import** org.ros.node.service.ServiceResponseBuilder;  
**import** org.ros.node.service.ServiceServer;  
**import** android.util.Log;  
  
**public class** Server **extends** AbstractNodeMain {  
  
 @Override  
 **public** GraphName getDefaultNodeName() {  
 **return** GraphName.*of*(**"rosjava\_tutorial\_services/server"**);  
 }  
  
 @Override  
 **public void** onStart(ConnectedNode connectedNode) {  
 Log.*d*(**"###########"**, **"create service server"**);  
 connectedNode.newServiceServer(**"AddTest"**, rosjava\_test\_msgs.AddTest.***\_TYPE***,  
 **new** ServiceResponseBuilder<rosjava\_test\_msgs.AddTestRequest, rosjava\_test\_msgs.AddTestResponse>() {  
 @Override  
 **public void** build(rosjava\_test\_msgs.AddTestRequest request, rosjava\_test\_msgs.AddTestResponse response) {  
 response.setSum(request.getA() + request.getB());  
 Log.*d*(**"###########"**, **"add test"** + response.getSum());  
 }  
 });  
 }  
}

Client.java

**package** com.github.rosjava.android\_apps.listener;  
  
**import** org.ros.exception.RemoteException;  
**import** org.ros.exception.RosRuntimeException;  
**import** org.ros.exception.ServiceNotFoundException;  
**import** org.ros.namespace.GraphName;  
**import** org.ros.node.AbstractNodeMain;  
**import** org.ros.node.ConnectedNode;  
**import** org.ros.node.NodeMain;  
**import** org.ros.node.service.ServiceClient;  
**import** org.ros.node.service.ServiceResponseListener;  
**import** android.util.Log;  
  
  
**public class** Client **extends** AbstractNodeMain {  
 @Override  
 **public** GraphName getDefaultNodeName() {  
 **return** GraphName.*of*(**"rosjava\_tutorial\_services/client"**);  
 }  
  
 @Override  
 **public void** onStart(**final** ConnectedNode connectedNode) {  
 ServiceClient<rosjava\_test\_msgs.AddTestRequest, rosjava\_test\_msgs.AddTestResponse> serviceClient;  
 **try** {  
 serviceClient = connectedNode.newServiceClient(**"AddTest"**, rosjava\_test\_msgs.AddTest.***\_TYPE***);  
 } **catch** (ServiceNotFoundException e) {  
 **throw new** RosRuntimeException(e);  
 }  
 **final** rosjava\_test\_msgs.AddTestRequest request = serviceClient.newMessage();  
 request.setA(2);  
 request.setB(10);  
 serviceClient.call(request, **new** ServiceResponseListener<rosjava\_test\_msgs.AddTestResponse>() {  
 @Override  
 **public void** onSuccess(rosjava\_test\_msgs.AddTestResponse response) {  
 connectedNode.getLog().info(  
 String.*format*(**"%d + %d = %d"**, request.getA(), request.getB(), response.getSum()));  
 Log.*d*(**"client"**, **"get request info, a = "** + request.getA() + **"b = "** + request.getB());  
 }  
  
 @Override  
 **public void** onFailure(RemoteException e) {  
 **throw new** RosRuntimeException(e);  
 }  
 });  
 }  
}