如何实现对话聊天

了解Pepper 聊天流程

在AI领域内，如何将智能语音聊天成熟应用到机器人产品上是一个必须解决的课题。机器人智能聊天不仅能解答用户的任务型、决策型问题实时帮助到用户，更能在闲聊问题中将机器人与人的友好交互最大化展现出来。

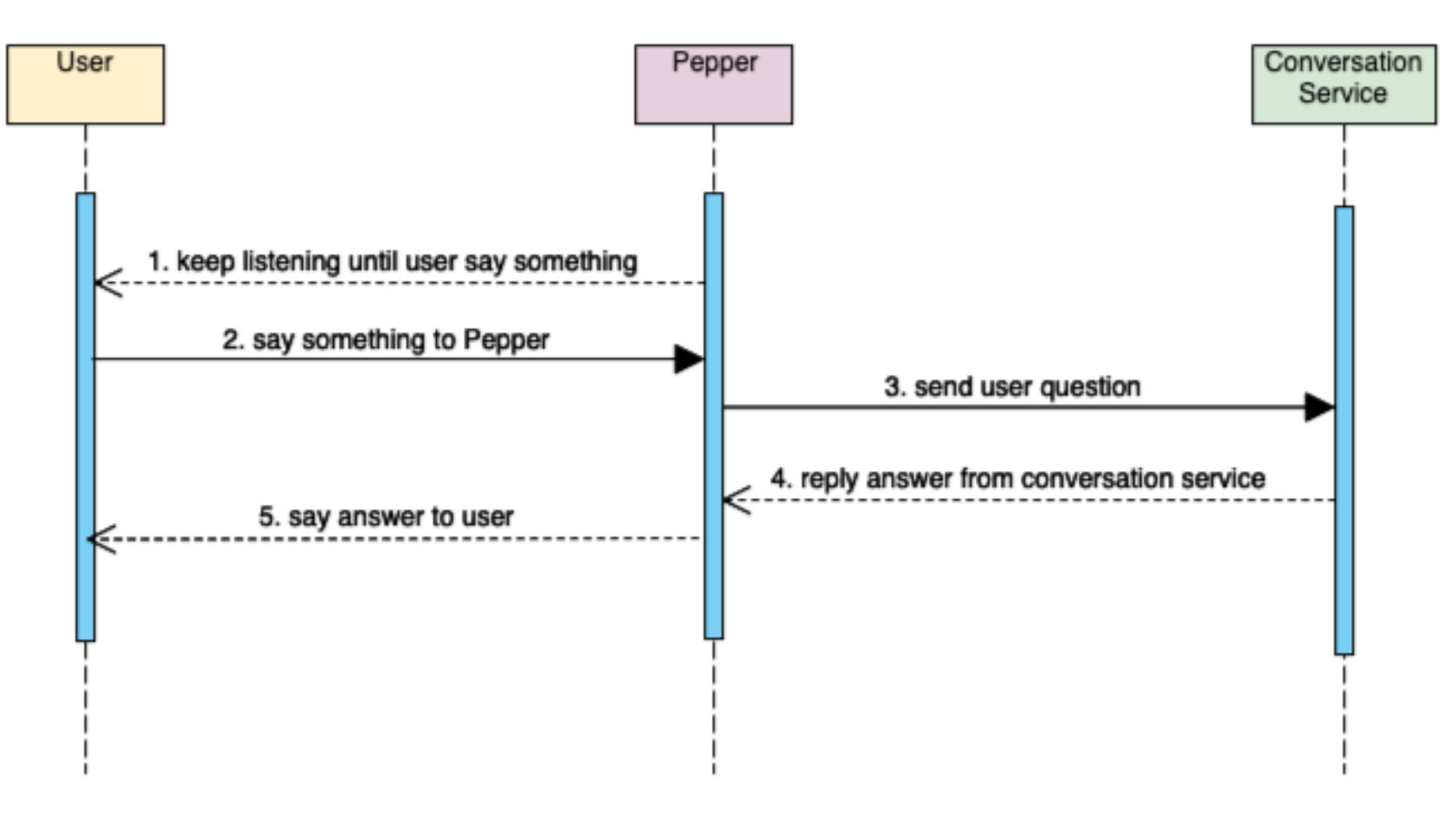
今天，我们将为大家讲解介绍Pepper的聊天交互流程，以及如何通过代码的方式来实现人机对话。

Pepper与用户的智能闲聊流程简要归纳如下：

Pepper通过头顶的麦克风聆听到用户的语音输入后，将输入语音识别转换为文本。
Pepper将文本跟本地或远程的对话服务进行查询，进而获取到回复文本。
Pepper将回复文本转换为回复语音后，再通过耳朵的扬声器播报给用户，从而实现一轮聊天交互。

综上，我们可以发现Pepper聊天功能其实是由Listen、Think、Say三部分组成。

交互流程可参考下图。



说明：

在上图的Pepper Listen环节里使用到了语音识别技术（Automatic Speech Recognition，简称 ASR）从而实现了自主接收用户语音，并将语音转换为文本的过程。

Pepper Think环节里Pepper接收用户的输入文本，并在内部通过算法完成对自然语言的理解，从而返回用户所期望的结果文本，此过程称之为语义理解。

Pepper Say环节是Pepper接收到结果文本后，将文本转换为语音，这里使用到文本转换语音技术（Text To Speech，简称TTS），最后再将语音结果播报给用户。

其中，当前Pepper 内置了Nuance的ASR（支持接入讯飞等第三方服务商），TTS。

使用QiSDK实现聊天功能

目前有两种方式可实现以下对应场景的聊天需求。

场景一：

用户在无网或弱网环境下与Pepper进行简单固定的对话交互。

示例场景：

用户可跟Pepper进行简单聊天互动，聊天内容主要包括打招呼、询问Pepper当前的时间等。

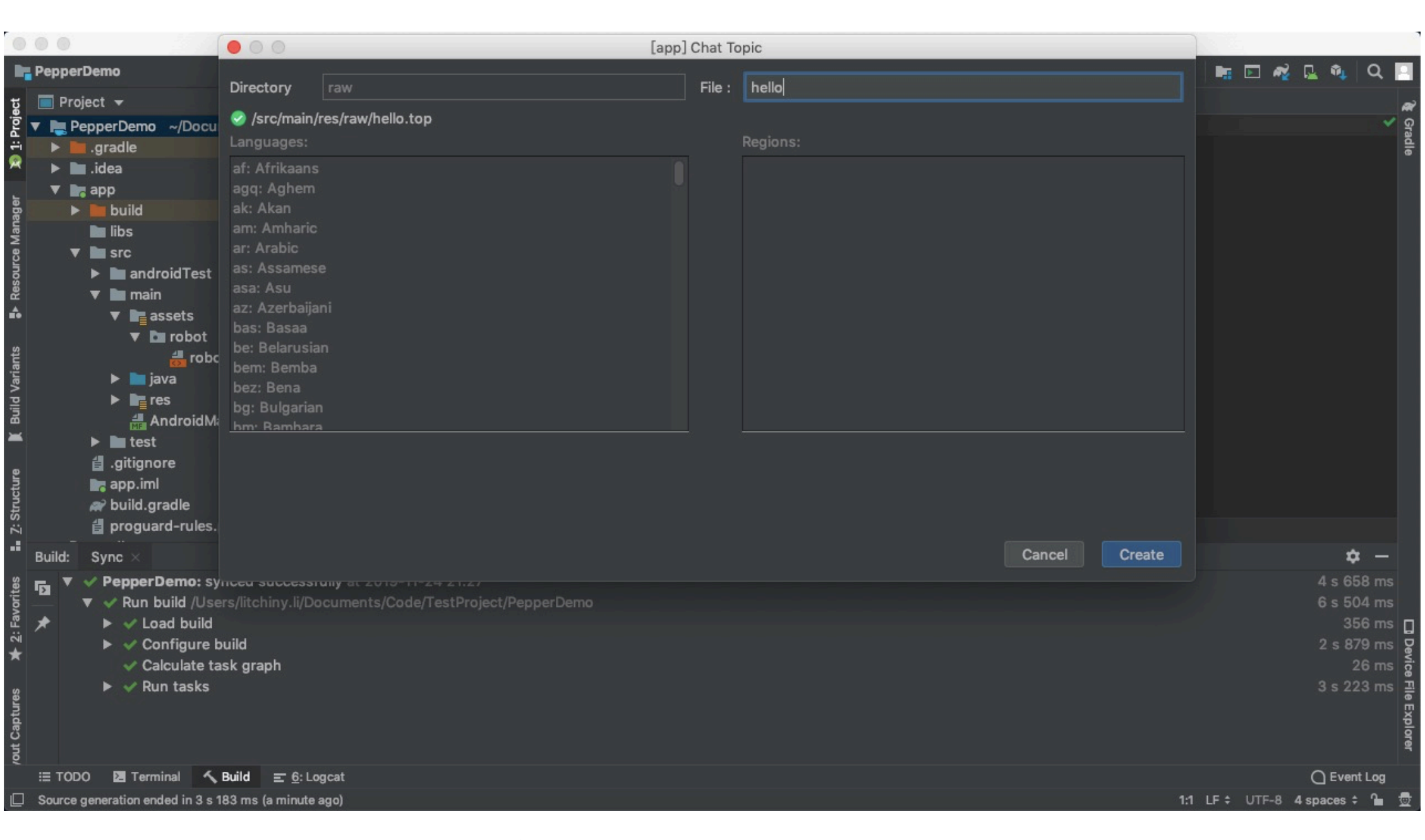
实现流程：

第一步：创建Robot应用。

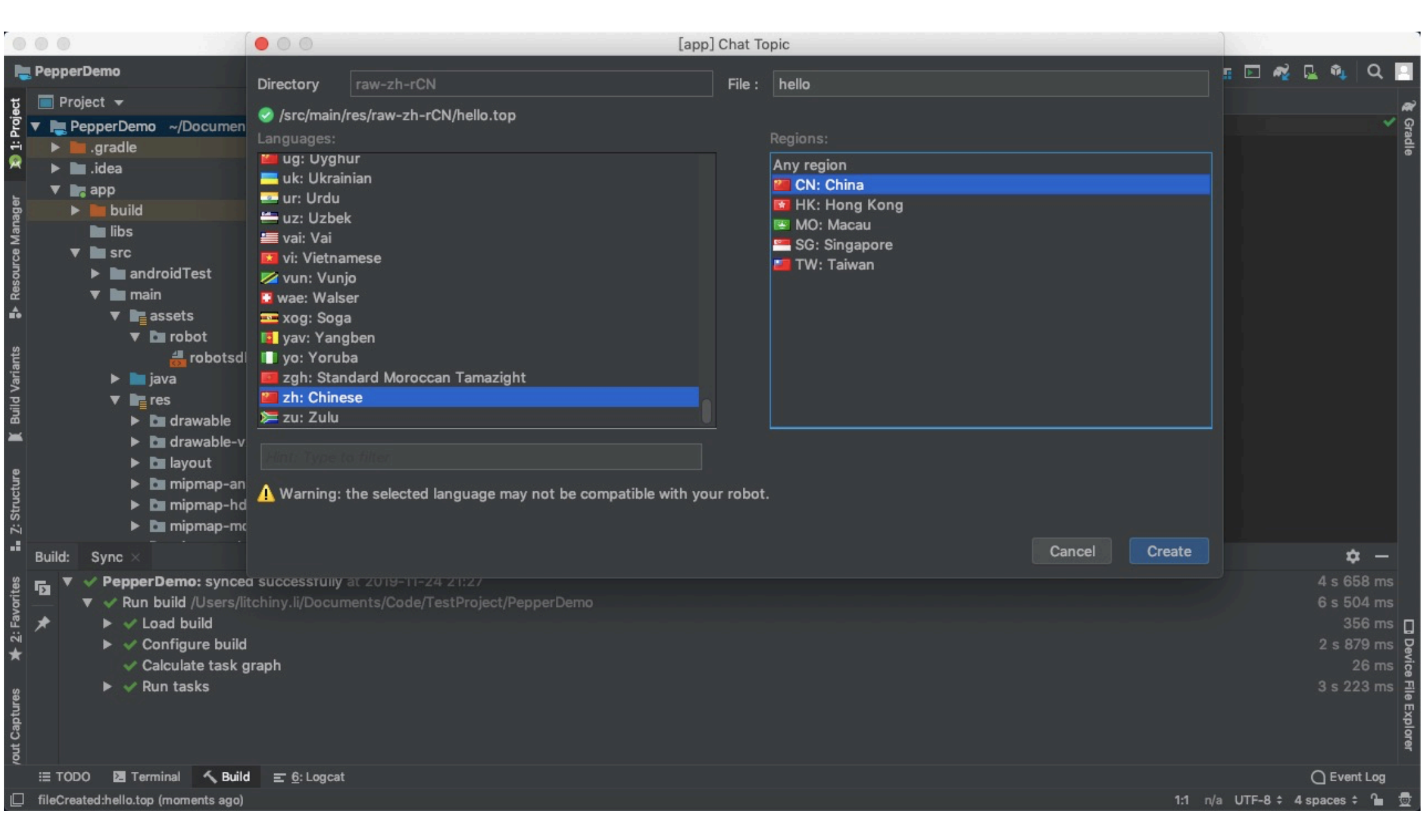
项目名为：PepperDemo；

第二步：创建Topic文件

右击选择项目名PepperDemo，找到New -> Chat Topic，创建hello.top（首次创建默认英文版本），该文件将添加到您的/res/raw/文件夹中。


[app] Chat Topic
Directory: raw
File: hello
/src/main/res/raw/hello.top

因当前案例中使用中文进行聊天对话，所以需重复步骤1，并在语言栏里选择中文，创建hello.top，该文件将添加到您的/res/raw-zh-rCN/文件夹中。


[app] Chat Topic
Directory: raw-zh-rCN
File: hello
/src/main/res/raw-zh-rCN/hello.top
Languages: zh: Chinese
Regions: CN: China
Warning: the selected language may not be compatible with your robot.

找到/res/raw-zh-rCN/hello.top，填写如下内容。

```
topic: ~hello()
concept:(hello) [你好 嗨]
concept:(age_question) [你多大了 你几岁了]
concept:(age_answer) [我今年5岁了呢, 别看我年纪小, 其实我懂得可多啦]
u:(~hello) ~hello
u:(~age_question) ~age_answer
concept:(time_question) [几点了 现在时间是多少]
u:(~time_question) 当前时间是：$time_answer
```

语法解释：

> u符号代表用户规则，其格式如下：

u:(human input) robot answer

当Pepper听到(human input)时，他会说出robot answer。

concept代表一组短语赋值给一个变量，在用户规则中使用~符号调用。

注意：

要发现如何设计更复杂的Topic，请参阅：QiChat-语法。

第三步：创建Topic实例

首先使用TopicBuilder类来构建一个Topic，然后传入第二步创建的top资源。

将以下代码放入onRobotFocusGained方法中：

```
Topic topic = TopicBuilder.with(qiContext)//使用QiContext创建Builder
                .withResource(R.raw.hello)  //设置 topic 资源
                .build();// 建立 topic
```

第四步：创建QiChatbot实例

首先使用QiChatbotBuilder类来构建一个QiChatbot，然后传入第三步创建的topic。

将以下代码放入onRobotFocusGained方法中：

```
QiChatbot qiChatbot = QiChatbotBuilder.with(qiContext)
                                       .withTopic(topic)
                                       .build();
```

第五步：实现简单时间查询服务

当前时间的返回值通过调用Calendar的相关API查询获取，以下为示例代码。

```
private String returnTimeAnswer(){
  Calendar calendar = Calendar.getInstance();
  int hour = calendar.get(Calendar.HOUR_OF_DAY);
  int minute = calendar.get(Calendar.MINUTE);
  return hour +"点"+ minute + "分";
}
```

在onRobotFocusGained方法里，qiChatbot设置获取到的时间返回值。

```
//设置top文件内当前时间问答的回复
qiChatbot.variable("time_answer").setValue(returnTimeAnswer());
```

注意：在hello.top文件的问答文本值的设定必须优先Chat的创建（请参考本案例中的时间设定），否则将无法在对话中获得回复。

第六步：创建Chat实例

现在增加Chat和Future的全局变量在你的MainActivity里。

```
private Chat chat;
private Future<Void> chatFuture;
```

首先使用ChatBuilder类来构建Chat，然后传入第四步创建的qiChatbot。

将以下代码放入onRobotFocusGained方法中：

```
//创建一个Chat
chat = ChatBuilder.with(qiContext)
                .withChatbot(qiChatbot)
                .build();
```

第七步：运行Chat，增设监听

Chat这个类有addOnStartedListener方法，该方法可让我们在Chat开始时得到通知，使用log在控制台将通知打印出来。

```
//设置开始Chat的监听
chat.addOnStartedListener(() -> Log.i(TAG, "Discussion started."));
```

我们现在可以在onRobotFocusGained方法里运行Chat了，并增加错误日志跟踪。

```
// 异步运行Chat
chatFuture = chat.async().run();
//在Chat运行过程中，如果遇到错误需要记录错误信息。
chatFuture.thenConsume(future -> {
    if (future.hasError()) {
        Log.e(TAG, "Discussion finished with error.", future.getError());
    }
});
```

注意：

需要在onRobotFocusLost方法里移除掉Chat的启动监听，并申请取消Chat的运行。

```
if (chat != null) {
    chat.removeAllOnStartedListeners();
}
if (chatFuture != null) {
    chatFuture.requestCancellation();
}
```

场景二：

用户和Pepper在较好网络环境中完成丰富语料的聊天互动。

目标场景：

用户可跟Pepper进行丰富语料的聊天互动，语料库的来源为Server端数据库。

实现流程：

第一步：创建Robot应用。

项目名为：PepperBaseChat；

第二步：创建你的ChatbotReaction

ChatbotReaction这个类可理解为Pepper对于用户语音输入的反应， 例如通常使Pepper说话（但也可以播放动画，在平板电脑上显示内容等）。

在您项目的app模块中，创建一个名为MyChatbotReaction的新类继承自BaseChatbotReaction，并根据下方例子实现对应抽象类：

```
public class MyChatbotReaction extends BaseChatbotReaction {
    private String text;
    private Future<Void> sayFuture;

    protected MyChatbotReaction(QiContext qiContext, String text) {
        super(qiContext);
        this.text = text;
    }

    @Override
    public void runWith(SpeechEngine speechEngine) {
        Say say = SayBuilder.with(speechEngine).withText(text).build();
        sayFuture = say.async().run();
    }

    @Override
    public void stop() {
        if (sayFuture != null) {
            sayFuture.requestCancellation();
            sayFuture = null;
        }
    }
}
```

第三步：创建你的Chatbot

现在我们有了自定义的MyChatbotReaction类，我们还需要再创建一个Chatbot类，该类将

调用Server端的对话接口。
绑定之前创建的MyChatbotReaction。

创建一个MyChatBot类，并为其提供以下代码：

```
public class MyChatBot extends BaseChatbot {
    private QiContext qiContext;

    protected MyChatBot(QiContext qiContext) {
        super(qiContext);
        this.qiContext = qiContext;
    }

    @Override
    public StandardReplyReaction replyTo(Phrase phrase, Locale locale) {
          //获得用户的问题文本
        String phraseText = phrase.getText();
          //获取对话回复内容
        String text = replyFromServer(phraseText);
          //创建 MyChatBotReaction
        MyChatbotReaction myChatBotReaction = new MyChatbotReaction(qiContext, text);
        return new StandardReplyReaction(myChatBotReaction,
                TextUtils.isEmpty(phraseText) ? ReplyPriority.FALLBACK : ReplyPriority.NORMAL);
    }

    private String replyFromServer(String text) {
        //调用Server端获取对话请求的回复内容, 请根据需求自行实现
        ...
        return "这是一个来自Pepper的问候";
    }
}
```

第四步：创建运行Chat

现在增加Future的全局变量在你的MainActivity里。

```
private Future<Void> chatFuture;
```

在MainActivity中的onRobotFocusGained方法中创建并运行Chat：

```
//创建 MyChatBot
MyChatBot myChatBot = new MyChatBot(qiContext);
//创建 Chat
Chat chat = ChatBuilder.with(qiContext)
                .withChatbot(myChatBot)
//      仅在使用第三方ASR需要调用此方法, 详细教程请查看常见问题Q1
//             .withAsrDriverParameters(myAsrParams)
                .build();
//运行chat
chatFuture = chat.async().run();
```

注意：

需要在onRobotFocusLost方法里申请取消Chat的允许。

```
if (chatFuture != null) {
    chatFuture.requestCancellation();
}
```

现在，您可以在Pepper上运行此应用程序，并在与Pepper语音交互时，聆听Pepper播报从Server端获取的对话回复。

常见问题

Q1：如何将讯飞语音识别应用在Pepper上？

A1：请参阅：讯飞语音识别。

Q2：如何修改Pepper 说话时的语速语调？

A2：因Pepper是通过Say来实现语音播报，所以可在说话文本中增添固定关键字来达到修改语速语调的目的。

关键字如下：

语调语速内容	Say效果	说明
\\rspd=50\\	语速调整（值越大，语速越快，反之则反）。	在需要更改语速的内容前插入 \\rspd=value\\，这个值在50至400之间. 默认值是 100。
\\vct=50\\	语调调整（值越大，语调越高，反之则反）。	在需要更改语调的内容前插入 \\vct=value\\，这个值在50至200之间. 默认值是 100。
\\pau=1000\\	停顿间隔调整。	在需要更改说下一句话停顿时间前插入\\pau=value\\，对应值的单位是秒。

示例代码：

```
 private void say(QiContext qiContext) {
        if (qiContext == null) return;
        String text = "我现在是正常说话的\\rspd=50\\我的语速变慢了。" +
                "\\vct=50\\我的语调变低了。" +
                "\\pau=1000\\我说话停顿的时间变长了";
        try {
            Say say = SayBuilder.with(qiContext)
              .withText(text)
              .buildAsync()
              .get();
            Future sayFuture = say.async().run();
        } catch (ExecutionException e) {
            e.printStackTrace();
        }
    }
```