介绍

从本质上讲，您可以将 Telegram Bot API视为为您的查询提供JSON 编码响应的软件。

另一方面，机器人本质上是一种例程、软件或脚本，它通过HTTPS 请求查询 API并等待响应。您可以发出多种类型的请求，也可以使用和接收许多不同的对象作为响应。

由于**您的浏览器**能够发送 HTTPS 请求，因此您可以使用它快速试用该 API。获取令牌后，尝试将此字符串粘贴到您的浏览器中：

```
https://api.telegram.org/bot<YOUR_BOT_TOKEN>/getMe
```

理论上，您可以通过浏览器或其他定制工具（如cURL ）使用此类**基本请求**与 API 进行交互。虽然这可以用于像上面示例这样的简单请求，但对于较大的应用程序来说并不实用，而且扩展性也不好。因此，本指南将向您展示如何使用库和框架以及一些**基本的编程技能**来构建更强大、更可扩展的项目。

如果您知道如何编码，您将很快完成每个步骤 – 如果您刚刚开始，本指南将向您展示您需要学习的一切。

> 在本指南中，我们将使用Java，因为它是最流行的编程语言之一。不过，您可以使用任何语言进行操作，因为所有步骤基本相同。
> 由于 Java 是完全跨平台的，因此每个代码示例都适用于任何操作系统。如果您选择其他语言， C#、Python、Go和TypeScript
> 中都有等效示例。

准备工作

首先，我们将简要介绍如何**创建您的第一个项目**、获取您的**API 令牌**以及下载所有必要的**依赖项和库**。

出于本指南的目的，您将要创建的机器人的副本也在@TutorialBot上线– 您可以随时查看它，以了解您自己的实现在每个步骤之后应该是什么样子。

获取你的机器人令牌

在此上下文中，**令牌**是用于在机器人 API 上验证您的机器人（而非您的帐户）的字符串。每个机器人都有一个唯一的令牌，也可以随时通过@BotFather撤销该令牌。

获取令牌非常简单，只需联系@BotFather、发出命令并按照步骤操作，直到获得新令牌即可。您可以在此处 `/newbot` 找到分步指南。

你的令牌看起来会像这样：

```
4839574812:AAFD39kkdpWt3ywyRZergyOLMaJhac60qc
```


创建你的项目

在 IntelliJ 中，转到 `File > New > Project` 。

填写相应字段：

- **名称**– 项目的名称。例如，*BotTutorial*。
- **位置**– 存储项目的位置。您可以使用默认值。
- **语言**–Java
- **构建系统**– 处理依赖项的框架。选择*Maven*。
- **JDK** – 选择您下载的版本。我们将使用版本*17*。
- **添加示例代码**– 保持**选中**状态，它将为您生成一些所需的文件。
- **高级设置>GroupId**–我们建议*教程*。
- **高级设置 > ArtifactId** – 您可以使用默认值。

点击*“创建”*后，如果一切操作正确，左上角的**项目**视图将显示如下**项目结构：**

```
BotTutorial
├─ .idea
├─ src
│  └─ main
│     └─ java
│        └─ tutorial
│           └─ Main
└─ pom.xml
```

> 其他 IDE 也遵循类似的模式。依赖管理系统将根据您选择的语言而具有不同的名称（如果是内置的，则根本没有名称）。

如果这看起来很吓人，别担心。我们将只使用 `Main` 文件和 `pom.xml` 文件。
事实上，要检查到目前为止一切是否正常，请双击*Main并单击public class Main*左侧的小绿色箭头，然后选择第一个选项。
如果您正确遵循了这些步骤，*Hello world!*应该会出现在下面的控制台中。

添加框架依赖

我们现在将指示 IDE 下载并配置使用 API 所需的一切。
这非常简单，并且在后台自动进行。

`pom.xml` 首先，在屏幕左侧找到您的文件。
双击打开它并添加：

```
<dependencies>
```

介绍 准备工作 开始编码 首次运行 接收消息 发送消息 Echo 机器人

获取你的机器人令牌 创建机器人类 复制文本

下载 IDE 可用方法 复制一切

选择一个框架或库 注册机器人 执行命令 按钮和键盘 导航 数据库 托管

创建你的项目 创建你的命令 按钮类型 进一步阅读

添加框架依赖 命令逻辑 创建按钮

创建键盘

发送键盘

菜单触发器

extends TelegramLongPollingBot。将出现一条红线 – 这仅仅意味着我们缺少一些重要的方法。

要解决此问题，请将鼠标悬停在红线上，单击“*实现方法*”，然后单击“确定”。
根据 IDE 的不同，此选项可能称为“*实现缺失方法*”或类似名称。

你应该得到这个 – 如果出现问题，请随意从这里复制并粘贴到你的课程中：

```
package tutorial;
import org.telegram.telegrambots.bots.TelegramLongPollingBot;
import org.telegram.telegrambots.meta.api.objects.Update;

public class Bot extends TelegramLongPollingBot {

  @Override
  public String getBotUsername() {
      return null;
  }

  @Override
  public String getBotToken() {
      return null;
  }

  @Override
  public void onUpdateReceived(Update update) {}

}
```

> 如果 TelegramLongPollingBot 下出现红线，则表示您未正确设置 pom.xml。如果是这种情况，请从此处重新启动。

可用方法

让我们逐一研究这 3 种方法。

- **getBotUsername** – 必须编辑此方法才能始终返回您的机器人的用户名。您应该用它替换*空返回值*。
- **getBotToken** – 框架将使用此方法来检索您的机器人令牌。您应该用令牌替换*空返回值*。

- **onUpdateReceived** – 这是最重要的方法。每当有新的更新可用时，它都会被自动调用。让我们 `System.out.println(update);` 在其中添加一个调用来快速显示我们得到的内容。

替换完所有字符串后，你应该得到这样的结果：

介绍 准备工作 开始编码 首次运行 接收消息 发送消息 Echo 机器人

获取你的机器人令牌 创建机器人类 复制文本

下载 IDE 可用方法 复制一切

选择一个框架或库 注册机器人 执行命令 按钮和键盘 导航 数据库 托管

创建你的项目 创建你的命令 按钮类型 进一步阅读

添加框架依赖 命令逻辑 创建按钮

创建键盘

发送键盘

菜单触发器

注册机器人

要在 API 上**注册机器人**，只需在将启动应用程序**的 main 方法中**添加几行代码即可。如果您将类命名为 `Bot` ，则您的 main 方法应如下所示：

```
public static void main(String[] args) throws TelegramApiException {
  TelegramBotsApi botsApi = new TelegramBotsApi(DefaultBotSession.class);
  botsApi.registerBot(new Bot());
}
```

> 您可以将该方法放在任何类中。由于 Main 类中有一个自动生成的 `main` 方法，因此我们将在本教程中使用该方法。

首次运行

现在是第一次**运行你的机器人的**
时候了。点击左侧的绿色箭头 `public static void main` 并选择第一个选项。

然后就什么都没有了。是的，有点虎头蛇尾。
这是因为您的机器人**没有任何内容可打印**– 没有**新的更新，**因为还没有人给它发消息。

如果您尝试在 Telegram 上向机器人发送消息，您将看到控制台中弹出**新更新**。此时，您就拥有了自己的 Telegram 机器人 – 相当了不起。现在，让我们让它更智能一些。

> 如果没有弹出任何内容，请确保您向正确的机器人发送了消息，并且您在代码中粘贴的令牌是正确的。

接收消息

每次有人向您的机器人发送**私人消息** `onUpdateReceived` 时，您的方法将被自动调用，并且您将能够处理 `update` 包含**消息的**参数以及大量其他信息，您可以在此处查看详细信息。

现在我们重点关注两个价值观：

- **用户**– 发送消息的人。通过 访问 `update.getMessage().getFrom()` 。
- **消息**– 已发送的内容。通过 访问 `update.getMessage()` 。

了解了这一点，我们可以在控制台输出中使其更清晰一些。

```
@Override
public void onUpdateReceived(Update update) {
    var msg = update.getMessage();
    var user = msg.getFrom();
```

介绍 准备工作 开始编码 首次运行 接收消息 发送消息 Echo 机器人

获取你的机器人令牌
下载 IDE
选择一个框架或库
创建你的项目
添加框架依赖

创建机器人类
可用方法
注册机器人

执行命令
创建你的命令
命令逻辑

按钮和键盘
按钮类型
创建按钮
创建键盘
发送键盘
菜单触发器

导航 数据库 托管

进一步阅读

复制文本
复制一切

```
    SendMessage sm = SendMessage.builder()
                     .chatId(who.toString()) //Who are we sending a message to
                     .text(what).build();    //Message content
    try {
         execute(sm);                        //Actually sending the message
    } catch (TelegramApiException e) {
         throw new RuntimeException(e);      //Any error will be printed here
    }
}
```

`main` 在注册机器人后，继续在方法中运行此方法。
在本例中，我们假设您的用户 ID 是 `1234` 。

```
public static void main(String[] args) throws TelegramApiException {
    TelegramBotsApi botsApi = new TelegramBotsApi(DefaultBotSession.class);
    Bot bot = new Bot();                 //We moved this line out of the register method, to access it la
ter
    botsApi.registerBot(bot);
    bot.sendText(1234L, "Hello World!");  //The L just turns the Integer into a Long
}
```

如果您正确执行了所有操作，每次启动代码时，您的机器人都会向您发送“*Hello World!*”。向群组或频道发送消息（假设您拥有相关权限）非常简单，只需将其替换 `1234` 为相应聊天的 ID 即可。

> 尝试使用其他类型的消息，例如 SendPhoto、SendSticker、SendDice...
> 完整列表可从这里开始查看。

Echo 机器人

让我们通过编写**Echo Bot**来练习到目前为止尝试过的一切。
它的功能相当简单：它收到的每条短信都会直接发回给用户。

复制文本

编码的最直观的方法是保存用户 ID 并 `sendText` 在每次更新后立即调用。

换句话说：

```
@Override
public void onUpdateReceived(Update update) {
    var msg = update.getMessage();
    var user = msg.getFrom();
```



```
}
```

替换方法调用后 `onUpdateReceived` ，运行代码将产生一个功能齐全的**Echo Bot**。

> 本教程假设更新始终包含消息，以简化操作。但事实可能并非总是如此 – 请务必在代码中实施所有适当的检查，以使用适当的方法处理每种类型的更新。

执行命令

要了解命令是什么以及它如何工作，我们建议阅读此专门摘要。
在本指南中，我们将重点介绍事物的技术方面。

创建你的命令

首先打开@BotFather。
类型 `/mybots` > *Your_Bot_Name* >编辑机器人>编辑命令。

现在发送一个新命令，然后发送简短的描述。
为了本教程的目的，我们将实现两个简单的命令：

```
scream - Speak, I'll scream right back
whisper - Shhhhhhh
```

命令逻辑

我们希望**Echo Bot**在**尖叫模式**时以大写字母回复，否则以正常字母回复。

首先，让我们**创建一个变量**来存储当前模式。

```
public class Bot extends TelegramLongPollingBot {

   private boolean screaming = false;

   [...]
}
```

然后，让我们改变一些逻辑来**解释这种模式**。

```
public void onUpdateReceived(Update update) {
    [...]                                   //Same variables as the previous versions
   if(screaming)                            //If we are screaming
       scream(id, update.getMessage());     //Call a custom method
```


如您所见，它会检查消息是否**为命令**。如果是，机器人将进入**尖叫模式**。
在更新方法中，我们检查**处于哪种模式**，然后复制消息或将其转换为大写，然后再将**其发回**。

就这样。现在机器人可以**执行命令**并相应地改变其行为。

当然，这种简化的逻辑将改变机器人对**每个人**的行为，而不仅仅是发送命令的人。这对于本教程来说可能很有趣，但**在生产环境中不起作用**——考虑使用 Map、字典或等效数据结构为各个用户分配设置。

> 记住始终执行一些基本的全局命令。
> 您可以通过对命令执行简单的反馈来练习 `/start` ，我们故意省略了这一点。

按钮和键盘

为了简化用户与机器人的互动，您可以用方便的按钮替换许多基于文本的交流。这些按钮可以执行各种操作，并且可以为每个用户进行自定义。

按钮类型

按钮主要有两种类型：

- **回复按钮- 用于提供预定义文本**回复选项的列表。
- **内联按钮**- 用于提供快速导航、快捷方式、URL、游戏等。

使用这些按钮就像将一个 `ReplyKeyboardMarkup` 或一个附加 `InlineKeyboardMarkup` 到您的 `SendMessage` 对象一样简单。

本指南将重点介绍**内联按钮，**因为它们只需要几行额外的代码。

创建按钮

首先，让我们创建一些按钮。

```
var next = InlineKeyboardButton.builder()
           .text("Next").callbackData("next")
           .build();

var back = InlineKeyboardButton.builder()
           .text("Back").callbackData("back")
           .build();

var url = InlineKeyboardButton.builder()
           .text("Tutorial")
```

```
            .url("https://core.telegram.org/bots/api")
            .build();
```

让我们回顾一下我们指定的字段：

介绍 准备工作 开始编码 首次运行 接收消息 发送消息 Echo 机器人

获取你的机器人令牌 创建机器人类 复制文本

下载 IDE 可用方法 复制一切

选择一个框架或库 注册机器人 执行命令 按钮和键盘 导航 数据库 托管

创建你的项目 创建你的命令 按钮类型 进一步阅读

添加框架依赖 命令逻辑 创建按钮

创建键盘

发送键盘

菜单触发器

```
            .keyboardRow(List.of(back))
            .keyboardRow(List.of(url))
            .build();
```

> 您可以将此代码放在您喜欢的任何位置，重要的是确保键盘变量可以从发送新菜单的方法调用中访问。如果您对这个概念感到困惑并且不知道将它们放在哪里，只需将它们粘贴到命令处理流程上方即可。

发送键盘

发送键盘只需要为消息指定回复标记。

```
public void sendMenu(Long who, String txt, InlineKeyboardMarkup kb){
    SendMessage sm = SendMessage.builder().chatId(who.toString())
            .parseMode("HTML").text(txt)
            .replyMarkup(kb).build();

    try {
        execute(sm);
    } catch (TelegramApiException e) {
        throw new RuntimeException(e);
    }
}
```

> 您可能已经注意到，我们还添加了一个新参数。 HTML 这称为格式化选项，它允许我们使用 HTML 标签并在稍后为文本添加格式。

菜单触发器

我们可以为每个新用户发送一个新菜单，但为了简单起见，我们添加一个将生成菜单的新命令。我们可以通过在之前的命令流中添加一个新的**else 子句来实现这一点。**

```
 var txt = msg.getText();
 if(msg.isCommand()) {
        if (txt.equals("/scream"))
            screaming = true;
        else if (txt.equals("/whisper"))
```

```
            screaming = false;
        else if (txt.equals("/menu"))
            sendMenu(id, "<b>Menu 1</b>", keyboardM1);
        return;
```


在这种情况下，处理仅意味着执行由按钮唯一标识的**操作，然后关闭查询**。

一个非常基本的按钮处理程序可能看起来像这样：

```
private void buttonTap(Long id, String queryId, String data, int msgId) {

    EditMessageText newTxt = EditMessageText.builder()
            .chatId(id.toString())
            .messageId(msgId).text("").build();

    EditMessageReplyMarkup newKb = EditMessageReplyMarkup.builder()
            .chatId(id.toString()).messageId(msgId).build();

    if(data.equals("next")) {
        newTxt.setText("MENU 2");
        newKb.setReplyMarkup(keyboardM2);
    } else if(data.equals("back")) {
        newTxt.setText("MENU 1");
        newKb.setReplyMarkup(keyboardM1);
    }

    AnswerCallbackQuery close = AnswerCallbackQuery.builder()
            .callbackQueryId(queryId).build();

    execute(close);
    execute(newTxt);
    execute(newKb);
}
```

使用此处理程序，只要点击按钮，您的机器人就会自动在内联菜单之间导航。
扩展此概念可实现可导航子菜单、设置和动态页面的无限组合。

数据库

Telegram**不会**为您托管更新数据库 - 一旦您处理并使用更新，它将不再可用。这意味着用户列表、消息列表、当前用户内联菜单、设置等功能**必须**由机器人开发人员实现和维护。

如果您的机器人需要这些功能之一，并且您想要开始进行**数据持久性**，我们建议您研究您选择的语言的序列化实践和库以及可用的数据库。

实现数据库超出了本指南的范围，但是，网上有几份关于简单嵌入式**开源**软件解决方案的指南，如SQLite、HyperSQL、Derby等等。

> 您选择的语言还将影响可用和支持的数据库 - 上面的列表假设您遵循此 Java 教程。

- **上传您的可执行文件/包**

一旦您的计算机与新服务器之间建立了有效的ssh
连接，您就应该上传可执行文件和所有相关文件。我们假设可运行的 jar `TutorialBot.jar` 及其数据库 `dbase.db` 当前位于 `/TBot` 文件夹中。

```
$ scp -r /TBot/ username@server_ip:/bots/TBotRemote/
```

- **运行应用程序**

根据您选择的语言，您可能需要以不同的方式配置服务器环境。如果您选择 Java，则只需安装兼容的 JDK。

```
$ apt install openjdk-17-jre
$ java -version
```

如果一切操作正确，您应该会看到 Java 版本作为输出，以及一些其他值。这意味着您已准备好运行应用程序。

现在，运行可执行文件：

```
$ cd /bots/TBotRemote/
$ java -jar TutorialBot.jar
```

您的机器人现在在线，用户可以随时与其互动。

> 为了简化和模块化此过程，您可以使用专门的docker 容器或等效服务。
> 如果您按照其中一个等效示例（C#、Python、Go和TypeScript ）进行操作，您可以在此处找到一组详细的说明来导出和运行代码。

进一步阅读

如果您已经读到这里，您可能会对这些额外的指南和文档感兴趣：

- 通用机器人平台概述
- 机器人功能详细列表
- 完整 API 参考

如果您在遵循本指南时遇到任何问题，您可以通过 Telegram 的@BotSupport联系我们。

电报

Telegram 是一款基于云的移动和桌面消息应用程序，注重安全性和速度。

关于

介绍 准备工作 开始编码 首次运行 接收消息 发送消息 Echo 机器人

获取你的机器人令牌 创建机器人类 复制文本

下载 IDE 可用方法 复制一切

选择一个框架或库 注册机器人 执行命令 按钮和键盘 导航 数据库 托管

创建你的项目 创建你的命令 按钮类型 进一步阅读

添加框架依赖 命令逻辑 创建按钮

创建键盘

发送键盘

菜单触发器

应用程序

平台

按
上