MINIAPPS

Development Guide

Getting started

Table of Contents

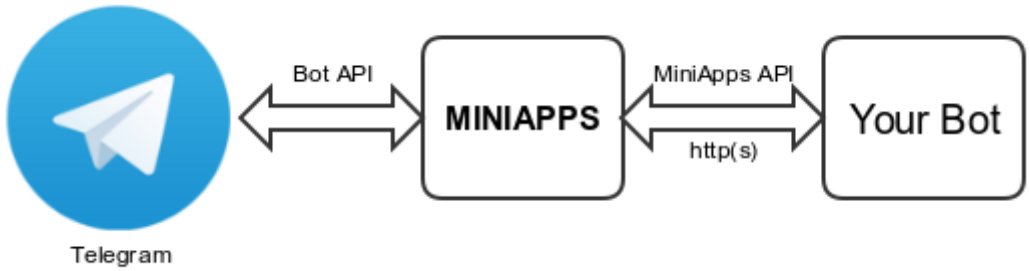
1 Introduction

This document introduces a step by step instruction of creating simple "Telegram" bots.

2 Preparation

First of all let's prepare our environment to be able to see results of our work.

For better understanding please pay attention on the picture below.

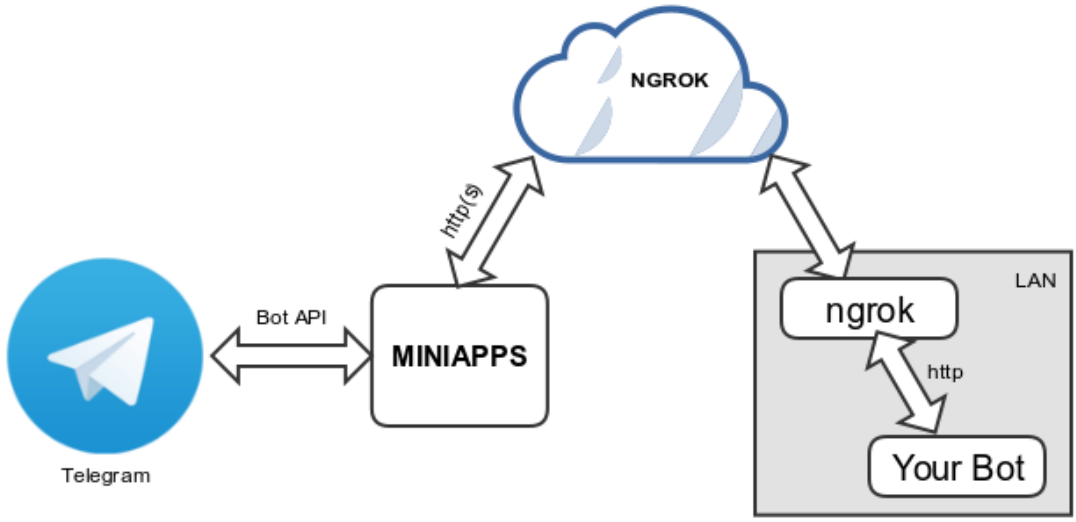


As you can see MiniApps communicates with your bot. It means the bot must be reachable by public URL. Thus you should have a public IP. Otherwise you can use NGROK service described in the next chapter.

To handle requests from MiniApps any http server is required. We will take Grizzly one.

2.1 NGROK service

As mentioned above this service is useful if your http server is located inside LAN. The picture in this case looks like this one:



The official detailed documentation is located here https://ngrok.com/docs. Following is a brief guide, just to force NGROK to be working for our purpose.

By means of this link you can download the latest version of the tool. Go to the catalog where you have downloaded the tool unpack and run it with the next command: **./ngrok http 8080**. In this case we will listen http port 8080. The output will be:

Code Block 1 ngrok output

```
ngrok by @inconshreveable                                       (Ctrl+C
to quit)

Tunnel Status                 online
```

```
Update                        update available (version 2.0.24, Ctrl-U to
update
Version                       2.0.19/2.0.25
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://d1d424ae.ngrok.io -> localhost:8080
Forwarding                    https://d1d424ae.ngrok.io -> localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.0
```

Tunnel Status must be "*online*" and external http(s) addresses available. From this output our local server is reachable by means of addresses:

```
http://d1d424ae.ngrok.io -> localhost:8080
https://d1d424ae.ngrok.io -> localhost:8080
```

2.2 Create MINIAPPS Account

The next step is to configure MiniApps.run to be able to connect "Telegram" to our bot. To do this we should configure a bot in MiniApps.run platform via "Personal account" page: https://dev.miniapps.run.

After login (you have to be registered) please push "*Connect bot via API*". Then:

1. Fill in "*Name*". You can select anyone. Let's say "Test".
2. In the line "*URL*" type the NGROK address with the suffix "/index.xml": https://d1d424ae.ngrok.io/index.xml (not secure "http://" can also be used). We will define the suffix in our tests later.
   Please remember to update this field by new URL if you restart "NGROK" service.
3. Push "Create" button.
4. Leave "*Request method*" by default: "*GET*".
5. The last thing you should do is to connect Telegram. Click the Telegram icon  and fill in "Telegram token". To have it please open a Telegram messenger and do the steps described in the next chapter.

Use 

link to go back to your bots list.

2.3 Register Telegram Bot

To do a further work you need have any kind of Telegram messenger: mobile application (for example for Android), web instance or desktop one and others.

Official description you can find here https://core.telegram.org/bots#botfather. Below is just a brief instruction.

1. Find the bot "*BotFather*" and click "*start*". The bot will answer with all possible commands it can treat. You need the first one:
   /newbot - create a new bot.
2. Click it or type and the bot will answer:
   Alright, a new bot. How are we going to call it? Please choose a name for your bot.

3. Type a name for you new bot. It can be whatever you want. Let's say "MyNewTest".
   You will see the answer:
   Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot.
4. Now fill in a username. It must be unique let's type MyNewTelegramTest_bot.

Finally you will see a message like this:

```
Done! Congratulations on your new bot. You will find it at
telegram.me/MyNewTelegramTest_bot. You can now add a
description, about section and profile picture for your bot, see
/help for a list of commands.

Use this token to access the HTTP API:
212117487:AAG-WRcJa...QtnGd8bUBu-L2LlQ

For a description of the Bot API, see this page:
https://core.telegram.org/bots/api
```

The main thing we need is the Telegram token which in our case is: **212117487:AAG-WRcJa...QtnGd8bUBu-L2LlQ**

(Fill in this value in the "*Telegram token*" field in the previous part "*Create MINIAPPS account*" and click "*Save*" to complete the service configuration.)

Now you can find you new bot by typing bot username: mynewtelegramtest_bot in "search" field like before or by clicking "telegram.me/MyNewTelegramTest_bot" link in the last BotFather's output.

Our preparation is done. Now let's make our first bot.

3 Hello World bot

This bot will reply "Hello World!" on the bot start in the "Telegram" messenger.

Let's write a Java code which starts up http server to handle GET requests. As a result we should have an XML page which MiniApps.run can treat.

We will divide the code into two classes. The first class: "MainClass" will be constant from test to test. The second one will differ for different bots.

3.1 Java code: MainClass class

Code Block 2 MainClass Java code

```
1    package test;
2
3    import org.glassfish.grizzly.http.server.*;
4    import java.io.IOException;
5
6    public class MainClass {
7        public static void main(String[] args) {
8            HttpServer httpServer = new HttpServer();
9            NetworkListener networkListener = new
NetworkListener("listener", "localhost", 8080);
10            httpServer.addListener(networkListener);
11            new HelloWorld(httpServer);
12
13            try {
14                httpServer.start();
15                System.out.println("Press any key to stop the
server...");
16                System.in.read();
17            } catch (IOException e) {
18                e.printStackTrace();
19            }
20        }
21    }
```

3.2 Java code: HelloWorld class

Code Block 3 OutputData Java code

```
1    package test;
2
3    import org.glassfish.grizzly.http.server.HttpHandler;
4    import org.glassfish.grizzly.http.server.HttpServer;
5    import org.glassfish.grizzly.http.server.Request;
6    import org.glassfish.grizzly.http.server.Response;
7
8    public class HelloWorld {
9
10       public HelloWorld(HttpServer httpServer){
11           httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
12               @Override
13               public void service(Request request, Response response)
throws Exception {
14                   response.setContentType("text/xml;charset=UTF-8");
```

```
15                      response.getWriter().write(
16                              "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n" +
17                                      "<page version=\"2.0\">\n" +
18                                      "<div>Hello World!</div>\n" +
19                                      "</page>"
20                      );
21                  }
22              }, "/index.xml");
23
24          }
25      }
```

Now we need to know if our server answers correct.

1. Start the server.
2. In a browser fill in an address: http://localhost:8080/index.xml. Why we have used this address? Please pay attention on the lines 9 of the "MainClass" class and 22 of the "HelloWorld" class.

We will see:

```
<page version="2.0">
 <div>Hello World!</div>
</page>
```

Then we need check if we can reach our server from outside like a request from MiniApps.run. As it was mentioned before we are using "NGROK" service. Replace the local address by new one: http://d1d424ae.ngrok.io/index.xml and check If you see the same output. If so our tunnel is working right.

According MiniApps XML format specification the output above looks ok. MiniApps.run should understand it.

Let's run our bot. Select the bot in the "Telegram" messenger and click "Start". If all done right you will see our "Hello World!" message.

4 Picture bot

Let's make our bot more capable. Let the bot to return a picture.

4.1 Java code: PictureBot class

Code Block 4 Picture Java code

```
1    package test.getpicture;
2
3    import org.glassfish.grizzly.http.server.HttpHandler;
4    import org.glassfish.grizzly.http.server.HttpServer;
5    import org.glassfish.grizzly.http.server.Request;
6    import org.glassfish.grizzly.http.server.Response;
7    import java.nio.file.Files;
8    import java.nio.file.Path;
9    import java.nio.file.Paths;
10
11    public class PictureBot {
12
13        public PictureBot(HttpServer httpServer){
14            httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
15                @Override
16                public void service(Request request, Response response)
throws Exception {
17                    response.setContentType("text/xml;charset=UTF-8");
18                    response.getWriter().write(
19                            "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n" +
20                                    "<page version=\"2.0\">\n" +
21                                    "<title>Picture</title>\n" +
22                                    "<attachment type=\"photo\"
src=\"resources/picture.jpg\"/>\n" +
23                                    "</page>"
24                    );
25                }
26            }, "/index.xml");
27
28            httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
29                @Override
30                public void service(Request request, Response response)
throws Exception {
31                    response.setContentType("image/jpeg");
32                    Path path = Paths.get("resources/picture.jpg");
33                    byte[] data = Files.readAllBytes(path);
34                    response.getOutputStream().write(data, 0,
data.length);
35                    response.getOutputStream().flush();
36                }
37            }, "/resources/picture.jpg");
38
39        }
40    }
```

In the line 22 you should set type as "photo" for correct displaying of the picture in the messenger. In the same line you should define a picture location. Any attachment you want to provide is set as "attachment" tag (the same Java code line). In this case Telegram will ask the content additionally.

To clarify where we should place the picture on our project let's have a look at the directory structure:

```
GetPicture/
 resources/
    picture.jpg
 src/
    test/
       getpicture/
          MainClass.java
          PictureBot.java
```

Continue. If you test your code via a browser you will see:

```
<page version="2.0">
 <title>Picture</title>
 <attachment type="photo" src="resources/picture.jpg"/>
</page>
```

Telegram understands the next types of attachment:

- video
- photo
- voice (OPUS format)
- document (arbitrary file)
- audio
- sticker (WebP format)
- location (`<attachment type="location" latitude="55.008353" longitude="82.935733"/>`)

Now you can test your bot. As a result you will see your picture.

5 Keyboard bot

What if we want to provide some different media content? We need give user a possibility to make a choice. Let's add our own keyboard to the bot. As a reaction on user's choice the bot will send a media file with selected type of attachment. For instance our bot will show three buttons to download "Video" or "Voice" files or show a Geo location. First two buttons will be located in one line and the "Geo" one will take a complete line of buttons.

5.1 Java code: KeyboardBot class

Code Block 5 Keyboard bot Java code

```
1    package test.keyboard;
2
3    import org.glassfish.grizzly.http.server.HttpHandler;
4    import org.glassfish.grizzly.http.server.HttpServer;
5    import org.glassfish.grizzly.http.server.Request;
6    import org.glassfish.grizzly.http.server.Response;
7    import java.io.IOException;
8    import java.nio.file.Files;
9    import java.nio.file.Path;
10    import java.nio.file.Paths;
11
12    public class KeyboardBot {
13
14        static byte[] readFile(String filename) throws IOException {
15            Path path = Paths.get(filename);
16            byte[] data = Files.readAllBytes(path);
17            return data;
18        }
19
20        public KeyboardBot(HttpServer httpServer){
21            httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
22                @Override
23                public void service(Request request, Response response)
throws Exception {
24                    response.setContentType("text/xml;charset=UTF-8");
25                    response.getWriter().write(
26                            "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n" +
27                                    "<page version=\"2.0\">\n" +
28                                    "<title>Please select</title>\n" +
29                                    "<navigation>\n" +
30                                    "<link
pageId=\"a01.xml\">Video</link>\n" +
31                                    "<link
pageId=\"a02.xml\">Voice</link>\n" +
32                                    "</navigation>\n" +
33                                    "<navigation>\n" +
34                                    "<link
pageId=\"a03.xml\">Geo</link>\n" +
35                                    "</navigation>\n" +
36                                    "</page>"
37                    );
38                }
39            }, "/index.xml");
40
41            httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
42                @Override
43                public void service(Request request, Response response)
```

```
throws Exception {
44                      response.setContentType("text/xml;charset=UTF-8");
45                      response.getWriter().write(
46                              "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n" +
47                                      "<page version=\"2.0\">\n" +
48                                      "<title>Video</title>\n" +
49                                      "<attachment type=\"video\"
src=\"resources/koala.mp4\"/>\n" +
50                                      "</page>"
51                      );
52                  }
53              }, "/a01.xml");
54
55              httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
56                  @Override
57                  public void service(Request request, Response response)
throws Exception {
58                      response.setContentType("video/mp4");
59                      byte[] data = readFile("resources/koala.mp4");
60                      response.getOutputStream().write(data, 0,
data.length);
61                      response.getOutputStream().flush();
62                  }
63              }, "/resources/koala.mp4");
64
65              httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
66                  @Override
67                  public void service(Request request, Response response)
throws Exception {
68                      response.setContentType("text/xml;charset=UTF-8");
69                      response.getWriter().write(
70                              "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n" +
71                                      "<page version=\"2.0\">\n" +
72                                      "<title>Voice</title>\n" +
73                                      "<attachment type=\"voice\"
src=\"resources/example.opus\"/>\n" +
74                                      "</page>"
75                      );
76                  }
77              }, "/a02.xml");
78
79              httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
80                  @Override
81                  public void service(Request request, Response response)
throws Exception {
82                      response.setContentType("audio/ogg");
83                      byte[] data = readFile("resources/example.opus");
84                      response.getOutputStream().write(data, 0,
data.length);
85                      response.getOutputStream().flush();
86                  }
87              }, "/resources/example.opus");
88
89              httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
90                  @Override
91                  public void service(Request request, Response response)
throws Exception {
92                      response.setContentType("text/xml;charset=UTF-8");
93                      response.getWriter().write(
94                              "<?xml version=\"1.0\" encoding=\"UTF-
```

```
8\"?>\n" +
95                                      "<page version=\"2.0\">\n" +
96                                      "<title>Geo location</title>\n" +
97                                      "<attachment type=\"location\"
latitude=\"54.861811\" longitude=\"83.091641\"/>\n" +
98                                      "</page>"
99                      );
100                 }
101             }, "/a03.xml");
102
103         }
104     }
```

When you start your server and run the bot it will show three buttons as expected. How to place buttons on right positions? Pay attention on the code lines 29-35. You can see "navigation" tags. So, one tag block includes one line of buttons. As mentioned before our code contains two lines of buttons.

6 Echo bot

Finally we are going to prepare a bot which receives some attachment from a Telegram user and send it back.

Let's send some mp3 file to the bot.

6.1 Java code: EchoBot class

Code Block 6 Echo Java code

```
1    package test.echo;
2
3    import org.glassfish.grizzly.http.server.HttpHandler;
4    import org.glassfish.grizzly.http.server.HttpServer;
5    import org.glassfish.grizzly.http.server.Request;
6    import org.glassfish.grizzly.http.server.Response;
7    import org.json.simple.JSONArray;
8    import org.json.simple.JSONObject;
9    import org.json.simple.parser.JSONParser;
10    import java.io.BufferedInputStream;
11    import java.net.MalformedURLException;
12    import java.net.URL;
13
14    public class EchoBot {
15
16        static String url = "";
17
18        public EchoBot(HttpServer httpServer){
19
20            httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
21                @Override
22                public void service(Request request, Response response)
throws Exception {
23                    response.setContentType("text/xml;charset=UTF-8");
24                    response.getWriter().write(
25                            "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n" +
26                                    "<page version=\"2.0\">\n" +
27                                    "<div>\n" +
28                                    "<input navigationId=\"submit\"
name=\"my_input\" title=\"Send me mp3 file or text message\"/>\n" +
29                                    "</div>\n" +
30                                    "<navigation id=\"submit\">" +
31                                    "<link
pageId=\"answer.xml\">Done</link>\n" +
32                                    "</navigation>\n" +
33                                    "</page>"
34                    );
35                }
36            }, "/index.xml");
37
38            httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
39                @Override
40                public void service(Request request, Response response)
throws Exception {
41                    String inputType =
request.getParameter("input_type");
42                    String input = request.getParameter("my_input");
43
44                    if(inputType != null) {
```

```
45                          JSONParser parser = new JSONParser();
46                          JSONArray array = (JSONArray)
parser.parse(input);
47                          for (int i = 0; i < array.size(); i++);
48                          JSONObject object = (JSONObject) array.get(0);
49                          url = (String) object.get("url");
50                          response.setContentType("text/xml;charset=UTF-
8");
51                          response.getWriter().write(
52                                  "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n" +
53                                          "<page version=\"2.0\">\n" +
54                                          "<title>Your mp3
file</title>\n" +
55                                          "<attachment type=\"audio\"
src=\"test.mp3\"/>\n" +
56                                          "</page>"
57                          );
58                      }
59                      else {
60                          response.setContentType("text/xml;charset=UTF-
8");
61                          response.getWriter().write(
62                                  "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n" +
63                                          "<page version=\"2.0\">\n" +
64                                          "<div>" + input + "</div>\n" +
65                                          "</page>"
66                          );
67
68                      }
69                  }
70              }, "/answer.xml");
71
72              httpServer.getServerConfiguration().addHttpHandler(new
HttpHandler() {
73                  @Override
74                  public void service(Request request, Response response)
throws MalformedURLException {
75                      URL url1 = new URL(url);
76                      byte[] buffer = new byte[1024];
77                      int count = 0;
78                      response.setContentType("audio/mpeg");
79                      try (BufferedInputStream bis = new
BufferedInputStream(url1.openStream())){
80                          while ((count = bis.read(buffer, 0, 1024)) != -
1){
81                              response.getOutputStream().write(buffer, 0,
count);
82                          }
83                          response.getOutputStream().flush();
84                      } catch (Exception e){
85                          e.printStackTrace();
86                      }
87                  }
88              }, "/test.mp3");
89          }
90      }
```

For reasons of test Java code simplicity, we are not checking if a Telegram user has uploaded a file with other content type.

To test it you should start the bot and attach some mp3 file. The bot will send it back.

Now you know how to create a simple "Telegram" bot and provide your service this way.