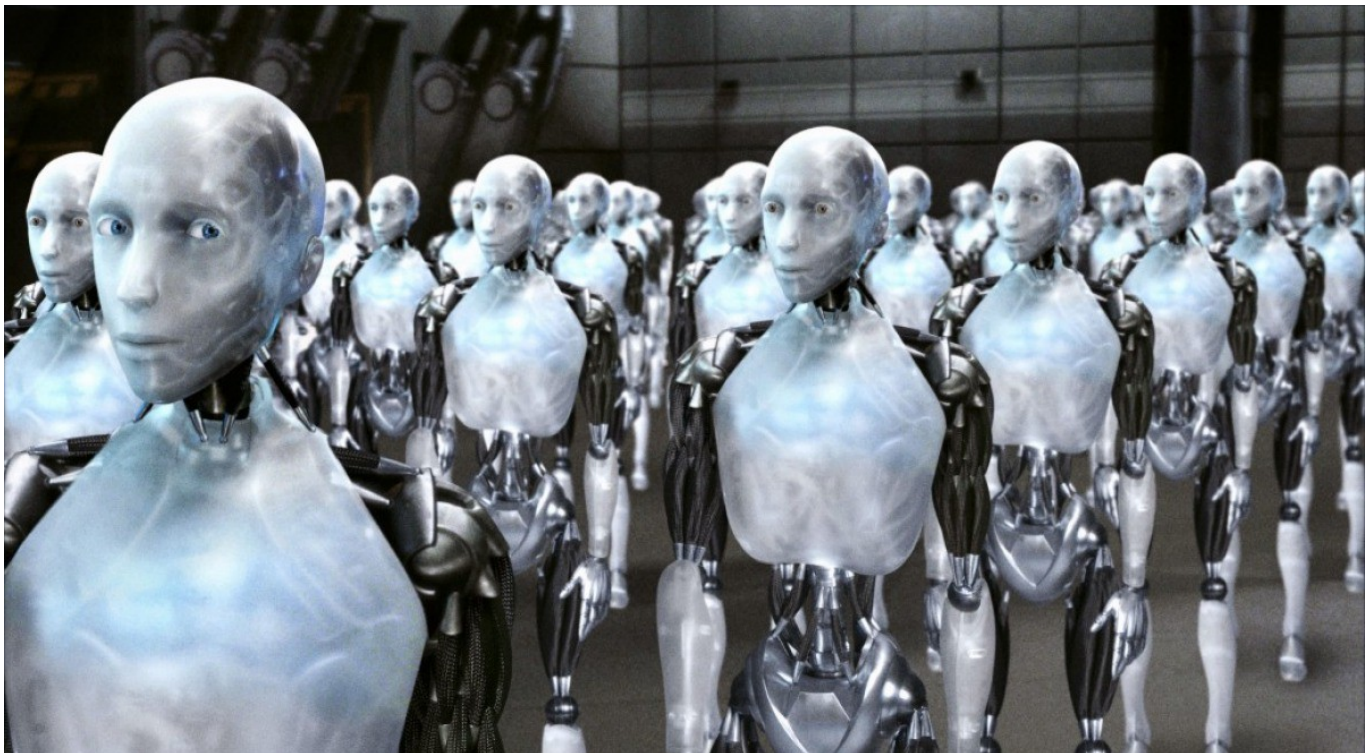


Criando bots do Telegram com API Java



Michel Fernandes [Follow](#)

Jun 29, 2017 · 8 min read



Atualmente estamos vivenciando um verdadeiro renascimento dos bots (versão abreviada de robots), que são processos automatizados e especializados em realizar tarefas específicas em prol dos usuários. Quando menciono o renascimento, é devido ao seu uso desde a época dos primeiros serviços on-line, predecessores da internet, onde eram utilizados essencialmente como Chat Bots, focados mais em utilizar uma inteligência artificial para simular uma conversa.

Os bots de hoje desempenham tarefas mais úteis e seguem para o ramo de atividade conhecida como Self-Care, auxiliando as empresas a substituírem cada vez mais o atendimento humano, seja presencial ou via telefone, por estes assistentes,

principalmente por sua grande facilidade tanto de implementar quanto ao usuário utilizá-los.

Das plataformas de conversação ou mensagens existentes no mercado, o Telegram é disparado o mais flexível e evoluído, principalmente por se tratar de uma iniciativa Open Source. O Facebook Messenger também inaugurou seu próprio sistema de bots há alguns meses atrás e a tendência é que todos os aplicativos tenham alguma forma de simular um assistente virtual e tirar mais proveito da base de usuários e criando mais pontes não somente entre os próprios usuários mas também com as empresas.

Grandes empresas já utilizam os bots de alguma forma, por exemplo a CEMIG já conta com este tipo de serviço para seus clientes para diversos tipos de solicitações, como pode ser verificado em mais detalhes nesta matéria. O Banco Original, por sua vez, também utiliza um bot, mas diferentemente, utiliza a plataforma do Facebook Messenger para disponibilizar extratos bancários e outros serviços corriqueiros de um cliente bancário, conforme esta notícia.

Em resumo, esta nova tendência de serviços está crescendo cada vez mais não apenas pela necessidade de redução de custos, mas também pelo motivo dos usuários quererem atendimento personalizado, direto e sem atravessadores. As plataformas na nuvem, sobretudo as que disponibilizam inteligência artificial como serviço, que é o caso da plataforma da IBM BlueMix com os serviços do Watson, podem potencializar ainda mais os bots, oferecendo capacidade de reconhecimento de linguagem natural e lógica de conversação, dentre outras ferramentas.

Primeiros passos

Antes de começarmos, é preciso que tenhamos uma conta válida no Telegram. A maneira mais simples é baixar o aplicativo das loja do seu respectivo sistema operacional, criar uma conta nova, pois é a partir do próprio aplicativo que solicitaremos a criação do bot.

O Telegram possui versões para todos os sistemas operacionais como também uma excelente versão Web neste link <https://web.telegram.org/#/im>.

Abaixo segue um exemplo de uma operação simples com o bot. Um detalhe importante é que um bot não tem um número de telefone associado, ele trabalha diretamente com um nome de usuário, deste modo, para acessar utilize o link

https://web.telegram.org/#/im?p=@botgram_bot ou contate diretamente o usuário @botgram_bot. Lembrando que este bot eu utilizo como testes e prova de conceito, não esperem utilizar com dados reais :).

domingo, 2 de outubro de 2016

O que esse bot pode fazer?

Bot modelo para entender as capacidades que ele pode fazer por você e sua empresa. Teste nossos comandos abaixo. Se tiver dúvidas, envie [/help](#) para mais informações. Obrigado!

COMEÇAR

Tela inicial do bot quando entramos no Telegram

Logo na primeira interação com o bot, é apresentado as informações básicas do que é possível fazer com ele e quais serviços ele pode dispor. Se concordar que ele pode lhe ajudar, clique em “Começar” para iniciar um diálogo. Uma coisa interessante é que até que a conversa seja encerrada pelo usuário, o bot pode interagir a qualquer momento, se valendo quando ativado, do Push Notification, para alertar por exemplo, promoções ou algum aviso importante.

Após iniciar, o Telegram automaticamente envia um comando /start que apresenta todos os comandos disponíveis, conforme abaixo. É como se fosse uma função de inicialização do bot, permitindo que o programador coloque suas próprias regras.



Michel
/start

21:54:17



Botgram
| Michel Fernandes

21:54:18

```
/start
```

Este bot contém os seguintes comandos disponíveis:

[/rastreo](#) para acompanhar suas encomendas nos Correios

[/megasena](#) para consultar os resultados da Megasena

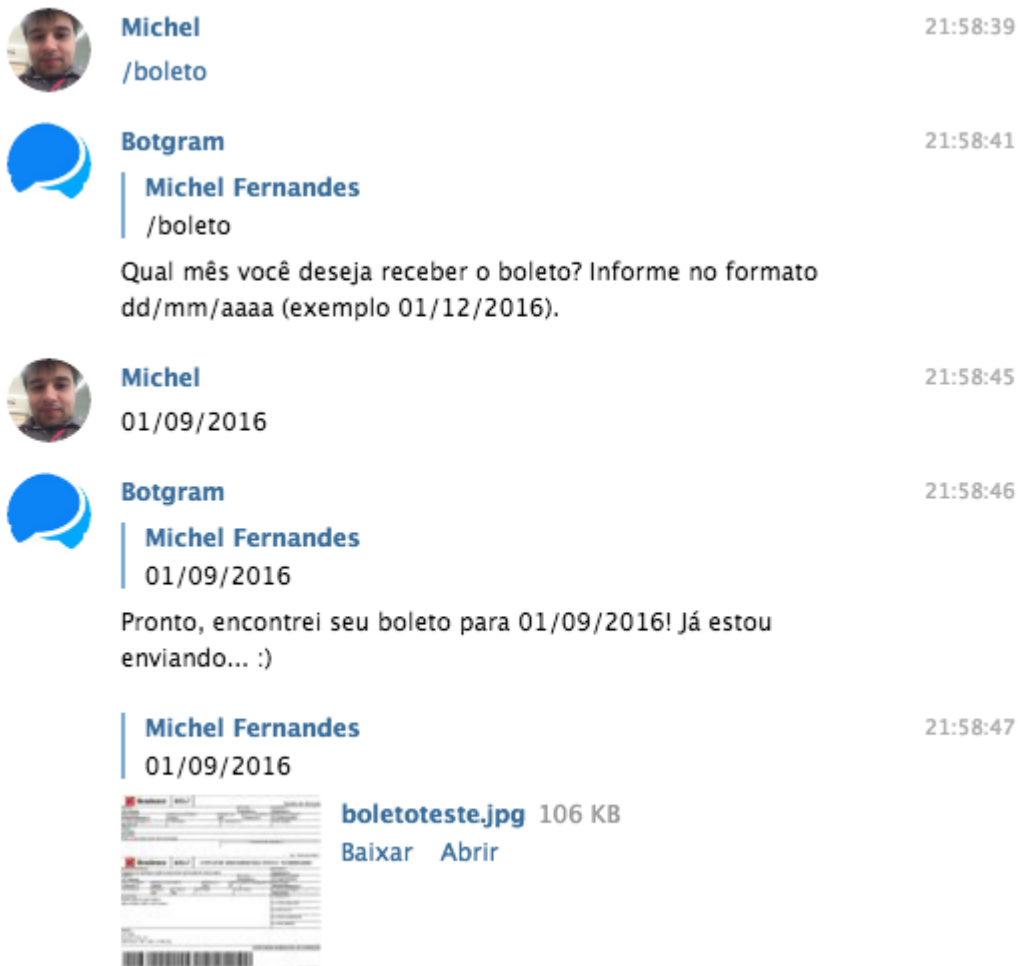
[/boleto](#) para receber boleto de cobrança

[/help](#) para listar os comandos disponíveis e ajuda.

Comandos disponíveis após entrar no bot

Os comandos representam uma forma mais simples de interagir pois não obriga o sistema a entender somente pelo diálogo o que o usuário deseja. De forma similar neste bot, o comando `/help`, também padrão, apresenta os comandos disponíveis.

Para testarmos uma simples interação, vamos utilizar o comando `/boleto`.



Execução de um comando no bot

Neste exemplo, o bot interage com o usuário em 3 etapas. A primeira explica o funcionamento do comando. Depois solicita uma data válida para obter o boleto respectivo. Finalmente envia o arquivo do boleto solicitado.

Cada interação é controlado pelo sistema gerenciador do bot, se encarregando por toda lógica e encadeamento necessário para tornar o diálogo fluente como também saber reconhecer o início e o fim dele.

Até agora espero que já temos entendido exatamente o que esperar de um bot do Telegram. Na próxima sessão iremos finalmente detalhar a criação de um deles.

Criando um novo bot

A criação de um novo bot é feita por um outro bot, o Bot Father. Acesse ele neste link <https://web.telegram.org/#/im?p=@BotFather> ou pelo usuário @BotFather.



BotFather

22:18:52








They call me the Botfather, I can help you create and set up Telegram bots. Please read this manual before we begin:
<https://core.telegram.org/bots>

You can control me by sending these commands:

- `/newbot` – create a new bot
- `/token` – generate authorization token
- `/revoke` – revoke bot access token
- `/setname` – change a bot's name
- `/setdescription` – change bot description
- `/setabouttext` – change bot about info
- `/setuserpic` – change bot profile photo
- `/setinline` – change inline settings
- `/setinlinegeo` – toggle inline location requests
- `/setinlinefeedback` – change inline feedback settings
- `/setcommands` – change bot commands list
- `/setjoingroups` – can your bot be added to groups?
- `/setprivacy` – what messages does your bot see in groups?
- `/deletebot` – delete a bot
- `/cancel` – cancel the current operation


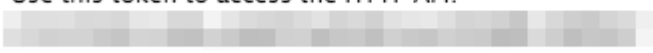
BotFather: o criador de bots no Telegram

Tal como os outros bots, ele possui uma série de comandos disponíveis e bem intuitivos. Para começarmos, vamos criar um novo bot com o comando `/newbot`.

-  **Michel** 22:20:30
/newbot
-  **BotFather** 22:20:30
Alright, a new bot. How are we going to call it? Please choose a name for your bot.
-  **Michel** 22:20:35
Java Bot
-  **BotFather** 22:20:35
Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot.
-  **Michel** 22:20:42
java_bot_blog
-  **BotFather** 22:20:42
Sorry, the username must end in 'bot'. E.g. 'Tetris_bot' or 'Tetrisbot'
-  **Michel** 22:20:52
javablogbot

Criando um novo bot com o BotFather

Durante o diálogo com o bot veja que há algumas regras na criação do nome do usuário, que precisa terminar com “bot”, com a finalidade clara para não confundir os usuários reais. Após isso é gerado o token de acesso para ser integrado na aplicação que ficará por trás de todas as regras como também é exibido o link do usuário, que neste caso ficou como [telegram.me/javablogbot](https://t.me/javablogbot).

-  **BotFather** 22:20:52
Done! Congratulations on your new bot. You will find it at [telegram.me/javablogbot](https://t.me/javablogbot). You can now add a description, about section and profile picture for your bot, see [/help](#) for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.
- Use this token to access the HTTP API:

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

Criando a aplicação Java

O Telegram possui uma vasta documentação de integração disponível, onde é possível criar o bot em qualquer linguagem de programação, pois todos seus métodos são disponibilizados em APIs REST. As comunidades de usuários por sua vez criam as APIs onde simplificam a integração e aceleram o desenvolvimento dos projetos.

No nosso caso, que iremos trabalhar com a linguagem Java, vamos utilizar a API Java API for Telegram Bots, disponível neste link do GitHub

<https://github.com/pengrad/java-telegram-bot-api>. Há opções de utilizar o Maven para a importação das bibliotecas necessárias. Para nosso exemplo ser mais simples, vamos optar por baixar os arquivos JAR e importá-los diretamente em nosso projeto. Os arquivos JAR podem ser baixados neste endereço

<https://oss.sonatype.org/content/repositories/releases/com/github/pengrad/java-telegram-bot-api/>. Após entrar neste link, opte pela versão mais recente que neste caso é a 2.1.2.

Index of /repositories/releases/com/github/pengrad/java-telegram-bot-api

| Name | Last Modified | Size | Description |
|----------------------------------|------------------------------|------|-------------|
| Parent Directory | | | |
| 1.0.0/ | Tue Aug 11 17:03:08 UTC 2015 | | |
| 1.0.1/ | Tue Aug 11 18:59:14 UTC 2015 | | |
| 1.0.2/ | Sun Sep 27 09:15:01 UTC 2015 | | |
| 1.0.3/ | Sun Sep 27 12:28:46 UTC 2015 | | |
| 1.1.0/ | Fri Oct 16 17:05:11 UTC 2015 | | |
| 1.2.0/ | Thu Oct 22 12:19:44 UTC 2015 | | |
| 1.2.1/ | Sat Oct 31 17:40:53 UTC 2015 | | |
| 1.2.2/ | Sun Nov 01 18:16:47 UTC 2015 | | |
| 1.2.3/ | Thu Nov 26 18:48:46 UTC 2015 | | |
| 1.2.4/ | Sun Dec 20 19:31:02 UTC 2015 | | |
| 1.3.0/ | Wed Jan 13 16:36:23 UTC 2016 | | |
| 1.3.1/ | Wed Jan 20 10:06:03 UTC 2016 | | |
| 1.3.2/ | Sat Jan 23 13:57:48 UTC 2016 | | |
| 1.3.3/ | Tue Apr 19 18:07:00 UTC 2016 | | |
| 2.0.0/ | Sat May 07 18:00:13 UTC 2016 | | |
| 2.0.1/ | Tue May 10 14:35:43 UTC 2016 | | |
| 2.1.0/ | Sat May 28 18:37:56 UTC 2016 | | |
| 2.1.1/ | Mon Jun 06 14:45:07 UTC 2016 | | |
| 2.1.2/ | Fri Sep 30 15:23:57 UTC 2016 | | |

| | | |
|---|------------------------------|-----|
| maven-metadata.xml | Fri Sep 30 15:23:56 UTC 2016 | 904 |
| maven-metadata.xml.md5 | Fri Sep 30 15:23:56 UTC 2016 | 32 |
| maven-metadata.xml.sha1 | Fri Sep 30 15:23:56 UTC 2016 | 40 |

Lista de versões da API

Após entrar na pasta com a versão mais recente, baixe o JAR sinalizado na figura abaixo.

Index of /repositories/releases/com/github/pengrad/java-telegram-bot-api/2.1.2

| Name | Last Modified | Size | Description |
|--|------------------------------|--------|-------------|
| Parent Directory | | | |
| java-telegram-bot-api-2.1.2-javadoc.jar | Fri Sep 30 15:22:12 UTC 2016 | 351282 | |
| java-telegram-bot-api-2.1.2-javadoc.jar.asc | Fri Sep 30 15:22:06 UTC 2016 | 475 | |
| java-telegram-bot-api-2.1.2-javadoc.jar.asc.md5 | Fri Sep 30 15:22:08 UTC 2016 | 32 | |
| java-telegram-bot-api-2.1.2-javadoc.jar.asc.sha1 | Fri Sep 30 15:22:07 UTC 2016 | 40 | |
| java-telegram-bot-api-2.1.2-javadoc.jar.md5 | Fri Sep 30 15:22:15 UTC 2016 | 32 | |
| java-telegram-bot-api-2.1.2-javadoc.jar.sha1 | Fri Sep 30 15:22:13 UTC 2016 | 40 | |
| java-telegram-bot-api-2.1.2-sources.jar | Fri Sep 30 15:22:03 UTC 2016 | 59081 | |
| java-telegram-bot-api-2.1.2-sources.jar.asc | Fri Sep 30 15:22:21 UTC 2016 | 475 | |
| java-telegram-bot-api-2.1.2-sources.jar.asc.md5 | Fri Sep 30 15:22:24 UTC 2016 | 32 | |
| java-telegram-bot-api-2.1.2-sources.jar.asc.sha1 | Fri Sep 30 15:22:23 UTC 2016 | 40 | |
| java-telegram-bot-api-2.1.2-sources.jar.md5 | Fri Sep 30 15:22:05 UTC 2016 | 32 | |
| java-telegram-bot-api-2.1.2-sources.jar.sha1 | Fri Sep 30 15:22:04 UTC 2016 | 40 | |
| java-telegram-bot-api-2.1.2.jar | Fri Sep 30 15:21:54 UTC 2016 | 99275 | |
| java-telegram-bot-api-2.1.2.jar.asc | Fri Sep 30 15:22:17 UTC 2016 | 475 | |
| java-telegram-bot-api-2.1.2.jar.asc.md5 | Fri Sep 30 15:22:20 UTC 2016 | 32 | |
| java-telegram-bot-api-2.1.2.jar.asc.sha1 | Fri Sep 30 15:22:19 UTC 2016 | 40 | |
| java-telegram-bot-api-2.1.2.jar.md5 | Fri Sep 30 15:21:57 UTC 2016 | 32 | |
| java-telegram-bot-api-2.1.2.jar.sha1 | Fri Sep 30 15:21:56 UTC 2016 | 40 | |
| java-telegram-bot-api-2.1.2.pom | Fri Sep 30 15:21:58 UTC 2016 | 1923 | |
| java-telegram-bot-api-2.1.2.pom.asc | Fri Sep 30 15:22:26 UTC 2016 | 475 | |
| java-telegram-bot-api-2.1.2.pom.asc.md5 | Fri Sep 30 15:22:31 UTC 2016 | 32 | |
| java-telegram-bot-api-2.1.2.pom.asc.sha1 | Fri Sep 30 15:22:29 UTC 2016 | 40 | |
| java-telegram-bot-api-2.1.2.pom.md5 | Fri Sep 30 15:22:00 UTC 2016 | 32 | |

Lista de arquivos contidos na API

Antes de prosseguirmos para desenvolvermos nossa aplicação Java, um conceito simples em relação a integração dos bots deve ficar clara. Há duas maneiras de prover a comunicação entre o serviço do Telegram e o nosso sistema:

Webhook: é a comunicação mais utilizada, por ser on-line, requer que uma página https seja acionada toda a vez que o bot for interagido. Requer obrigatoriamente um sistema web para processar as solicitações.

Pooling: é a forma mais simples, pois não requer acesso contínuo na internet como também não precisa de um certificado válido (ex. https). Basicamente as mensagens pendentes serão processadas periodicamente e terão um delay mínimo de processamento. No nosso exemplo iremos trabalhar com este método por ser mais aderente ao JavaSE.

Construindo a aplicação Java

Crie um novo projeto, depois crie uma nova pasta chamada “lib” para copiarmos o JAR da biblioteca da API do Telegram e outras bibliotecas que são necessárias também. Para facilitar, abaixo segue a lista delas. Se for utilizar o Maven, não se preocupe pois todas elas constam no POM, que é o arquivo que gerencia todas as dependências.

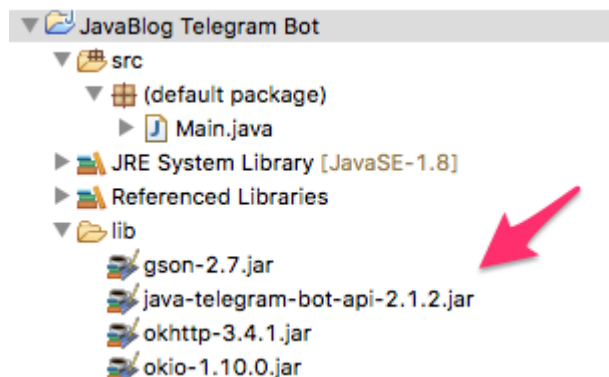
1. API Telegram

(<https://oss.sonatype.org/content/repositories/releases/com/github/pengrad/java-telegram-bot-api/2.1.2/>)

2. Okio (<http://repo1.maven.org/maven2/com/squareup/okio/okio/1.10.0/>)

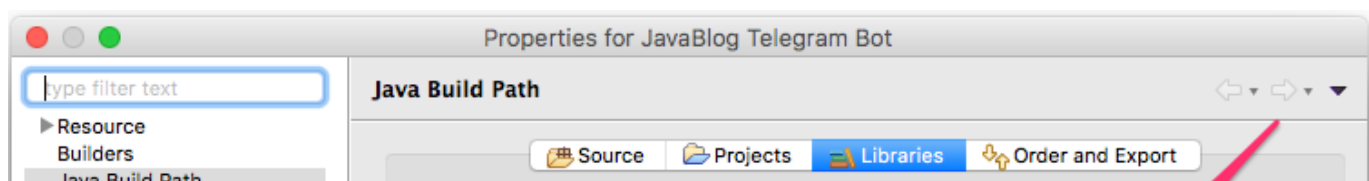
3. OkHttp (<http://repo1.maven.org/maven2/com/squareup/okhttp3/okhttp/3.4.1/>)

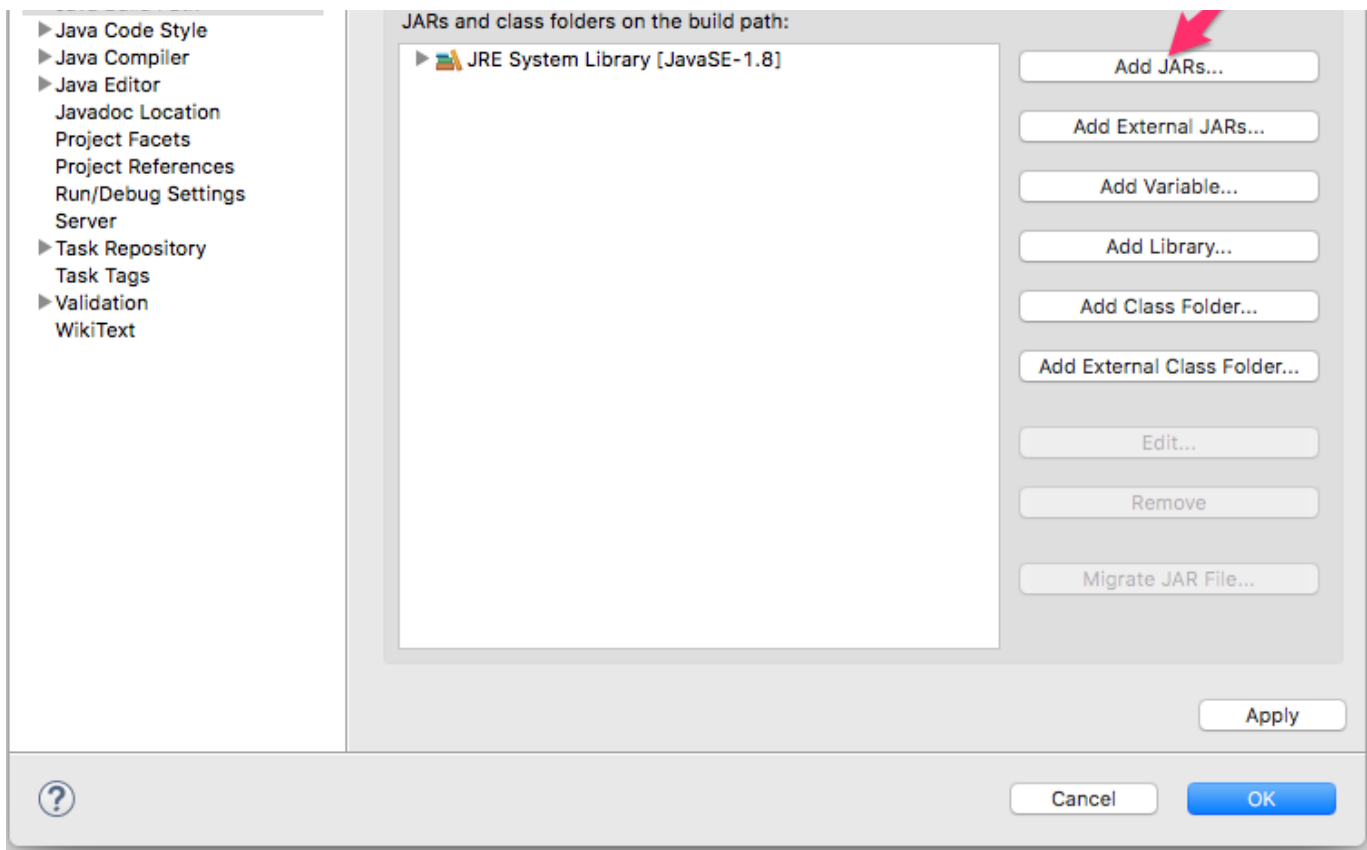
4. Gson (<http://repo1.maven.org/maven2/com/google/code/gson/gson/2.7/>)



Estrutura do projeto Java

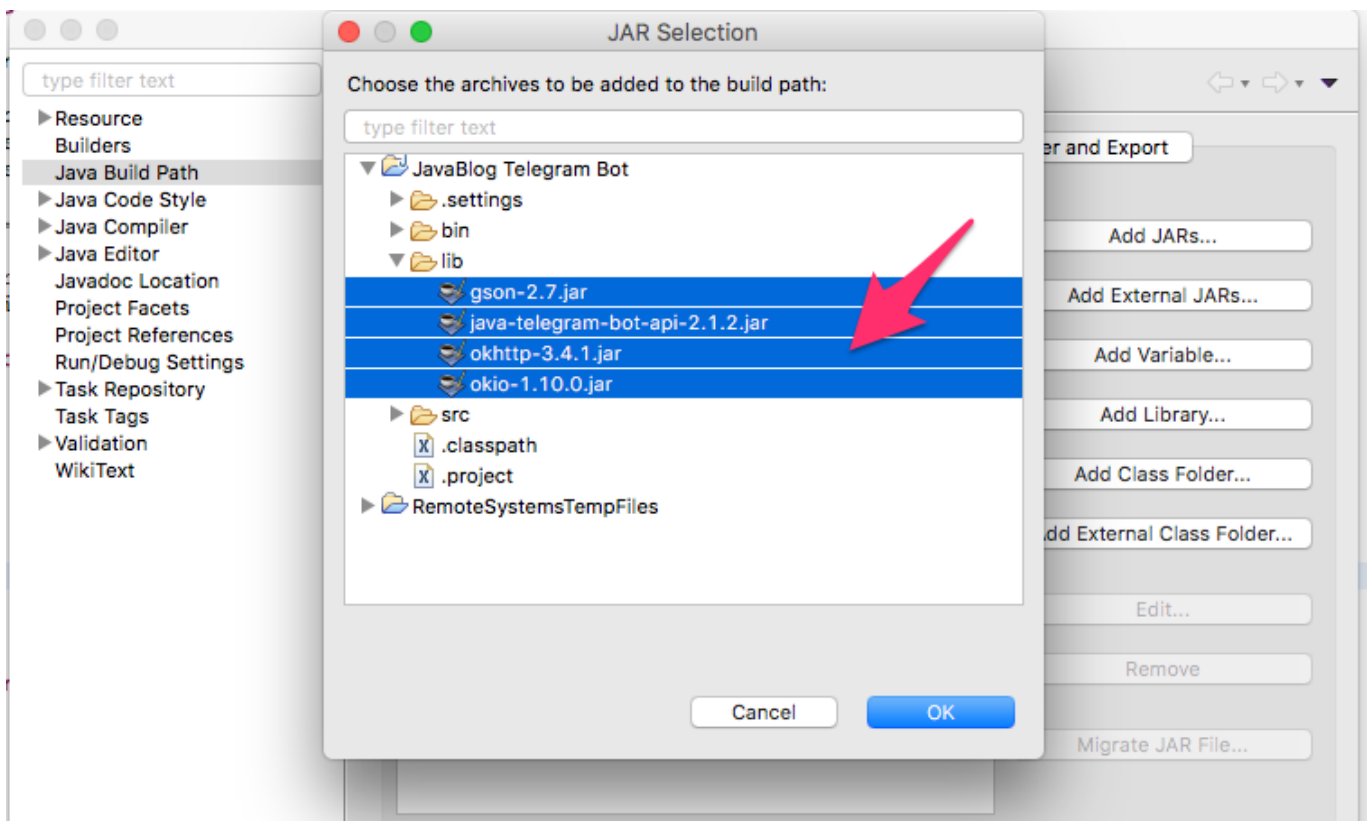
A seguir, vamos incluir esta biblioteca no Build Path, para que fique acessível ao projeto. Vá no menu “Project>Properties”, depois clique em “Java Build Path”.

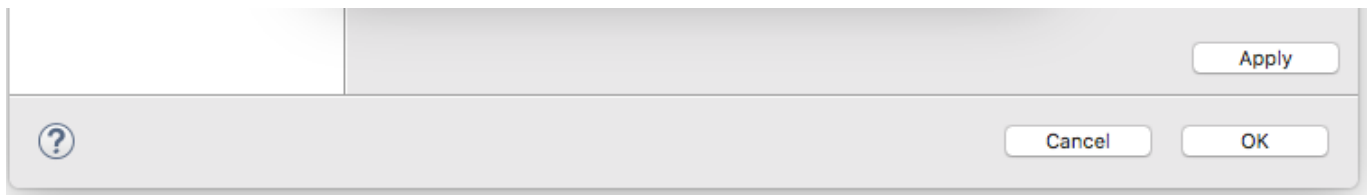




Propriedades do projeto Java

Clique em “Add JARs...” e escolha o arquivo que copiamos na pasta lib, dentro do projeto.





Seleção de JARs para uso no projeto

Pronto, agora estamos preparados para iniciar a programação deste novo bot!

Desenvolvendo a aplicação

Para fins didáticos, iremos desenvolver esta aplicação da forma mais simples. Posteriormente você poderá ampliar o uso e utilizar outros recursos interessantes da API. Para este primeiro exemplo vamos receber as mensagens enviadas ao bot, depois enviaremos uma ação de “Escrevendo” para simular que o bot está de fato atuando na resposta e, finalmente, iremos responder de volta.

Qualquer mensagem que recebida será respondida como “Não entendi...”.

```
1  public class Main {
2
3      public static void main(String[] args) {
4
5          //Criação do objeto bot com as informações de acesso
6          TelegramBot bot = TelegramBotAdapter.build("SEU_TOKEN_DE_ACESSO");
7
8          //objeto responsável por receber as mensagens
9          GetUpdatesResponse updatesResponse;
10         //objeto responsável por gerenciar o envio de respostas
11         SendResponse sendResponse;
12         //objeto responsável por gerenciar o envio de ações do chat
13         BaseResponse baseResponse;
14
15         //controle de off-set, isto é, a partir deste ID será lido as mensagens
16         int m=0;
17
18         //loop infinito pode ser alterado por algum timer de intervalo curto
19         while (true){
20
21             //executa comando no Telegram para obter as mensagens pendentes
22             updatesResponse = bot.execute(new GetUpdates().limit(100).offse
23
24             //...
25         }
```

```
24 //lista de mensagens
25 List<Update> updates = updatesResponse.updates();
26
27 //análise de cada ação da mensagem
28 for (Update update : updates) {
29
30     //atualização do off-set
31     m = update.updateId()+1;
32
33     System.out.println("Recebendo mensagem:" + update.message
34
35     //envio de "Escrevendo" antes de enviar a resposta
36     baseResponse = bot.execute(new SendChatAction(update.mes
37     //verificação de ação de chat foi enviada com sucesso
38     System.out.println("Resposta de Chat Action Enviada?" +
39
40     //envio da mensagem de resposta
41     sendResponse = bot.execute(new SendMessage(update.messag
42     //verificação de mensagem enviada com sucesso
43     System.out.println("Mensagem Enviada?" +sendResponse.isO
44
45     }
46
47 }
48
49 }
50
51 }
```

Main.java hosted with ♥ by GitHub

[view raw](#)

O código é bem simples.

Inicialmente inicializamos o objeto “bot” com o token de acesso. Logo em seguida, criamos um pooling num loop infinito que poderia ser substituído por um processo periódico que funcionasse em um intervalo curto de tempo, para que a experiência de conversar com o bot não fique prejudicada. Dentro do pooling, é enviado um comando ao Telegram para recuperar as mensagens pendentes para o bot a partir de um certo off-set que é o último ID, ou seja, para evitar trazer mensagens repetidas pois as mesmas ficam por um curto período de tempo no buffer do sistema, é possível enviar esse comando sempre com o ID da última mensagem incrementado para trazer novas

mensagens a partir desta. Prosseguindo, o próximo passo é enviar um feedback ao usuário que o bot está trabalhando, que são os Chat Actions. Estas ações possuem as possíveis atualizações de status, como “Escrevendo...”, “Enviando Arquivo...”, etc. É uma forma de entender que o bot está de fato processando a requisição. O objeto `baseResponse` é responsável por obter o resultado do envio do Chat Action, para conferências e validações de envio. Após isso é enviado a mensagem de resposta e seu resultado é armazenado no objeto `sendResponse`.

A partir deste simples exemplo é possível evoluir para reconhecimento de comandos e até mesmo utilizar o Java para consumir APIs de terceiros, na mesma ideia do que foi apresentado no bot de exemplo na introdução deste artigo.

O projeto completo pode ser obtido no GitHub.

[Telegram](#) [Java](#) [Chatbots](#)

[About](#) [Help](#) [Legal](#)