Conhecendo a API Twitter4J

Aprenda a utilizar a API de acesso a dados do Twitter para automatizar ações e extrair dados relevantes

Por que este artigo é útil:

Hoje em dia, com o grande número de dados disponíveis, sobretudo nas redes sociais, extrair informações úteis e interagir de forma ágil para a tomada de decisões representam um grande desafio. Neste contexto, o desenvolvimento de sistemas que possibilitam obter esses dados, correlacioná-los e transformá-los em informações úteis é de grande valor para diversos ramos de negócio, especialmente para os que demandam mais contato com o cliente. Diante desse desafio, este artigo propõe o uso da API Twitter4J para viabilizar a integração com o Twitter e assim explorar suas diversas funcionalidades objetivando a coleta, busca e análise de dados, como também formas de interação automatizadas cada vez mais eficazes.

Com a popularização das redes sociais, sobretudo a rede social Twitter, determinados segmentos de negócio ou mesmo outros segmentos de interesses específicos têm a necessidade de acompanhar de uma maneira mais eficiente os principais assuntos e tendências, baseado no comportamento dos milhares de usuários da rede social que a utilizam diariamente pelo mundo inteiro.

Não é novidade que este tipo de informação seja aplicada em vários setores empresariais. Exemplos típicos vêm, na maioria das vezes, de departamentos de pesquisa de mercado e atendimento ao cliente. Estes departamentos recorrem às redes sociais para descobrir demandas por novos produtos, novos públicos-alvo e também para obter feedbacks dos produtos já lançados e da qualidade dos serviços prestados. Essas informações podem compor um termômetro de uma marca ou mesmo de um determinado departamento (atendimento ao cliente, serviços, lojas, etc.) em uma empresa, ou seja, a coleção de mensagens trocadas nas redes sociais, sejam positivas ou negativas, podem guiar ou fornecer informações importantes para tomada de decisões e aprimoramento nos processos empresariais.

Outro ponto importante é usufruir da interação com os usuários e poder transformar uma reclamação em um elogio, após um problema ser resolvido. Com o notável número de dados e a precisão que alguns deles apresentam, é possível ainda descobrir informações novas com a aplicação de análises mais complexas.

No passado, a coleta de informações era mais complicada e demandava custosas minerações de dados, assim como bastante tempo. Às vezes, devido a essa lentidão ocasionada no processamento dos dados, quando as informações chegavam até a cadeia de tomada de decisão, eram simplesmente descartadas, pois já não eram mais válidas devido a velocidade e o dinamismo com que as coisas acontecem.

O Big Data, representado pelo grande número de dados disponível atualmente, sobretudo na internet e em suas redes sociais, como o Twitter, é uma fonte poderosa de informações. A rede social Twitter constitui uma valiosa fonte de informações pois apresenta seus dados disponíveis de forma on-line além de possuir registro dos usuários integrantes verificados que contribui com o nível de credibilidade nas informações disponibilizadas. Adicionalmente, todas as postagens realizadas podem contar com características associadas que enriquecem seus dados, tais como localização geográfica e popularidade. A obtenção destes dados de forma on-line, seu entendimento e correlação, realizado em tempo adequado, pode auxiliar consideravelmente nas tomadas de decisões cada vez mais rápidas e eficientes.

Diante da abundância dos dados, o desafio de compreendê-los e, principalmente, convertê-los em informações úteis ou conhecimento, pode fazer a diferença na tomada de decisões importantes. Por conta disto, o uso de aplicações de descoberta e mineração de informações, ou seja, aplicações que

correlacionam dados, que buscam por conexões em diferentes fontes de dados, é considerado como uma verdadeira necessidade nos dias atuais.

Para entendermos melhor como podemos obter, a partir do Big Data, informações de uma rede social e, por meio dela, identificar certos padrões, vamos tomar como exemplo uma empresa hipotética do setor de serviços. Nesta empresa, os setores de qualidade de serviço e atendimento ao consumidor monitoram as perguntas e feedbacks de cada um dos clientes por meio das postagens nas redes sociais.

Tais postagens não representam apenas dúvidas e soluções de problemas, elas podem contribuir também para conhecer e acompanhar a reputação da empresa. A quantidade de comentários, citações e respostas, podem formar um padrão específico que pode indicar alguma informação nova que talvez não tenha sido descoberto pela empresa, como uma boa aceitação de um produto recém-lançado ou alguma referência negativa; por exemplo, um atendimento ruim ou produto com defeito. O refinamento destas informações pode ser feito com algoritmos específicos de reconhecimento de padrões e inteligência artificial com o objetivo de aprimorar a leitura das informações e classificá-las como neutras, negativas ou positivas, como uma forma de traduzir todos os dados extraídos das redes sociais em uma informação útil e relevante.

A correlação dos dados das redes sociais com os dados internos da empresa para a descoberta de padrões é essencial para torná-la cada vez mais ágil na tomada de decisões e assim evitar que um problema seja intensificado ou até mesmo aproveitar uma situação positiva, como um sucesso maior do que o esperado. Um exemplo deste tipo de correlação é acompanhar comentários, hashtags ou qualquer outro tipo de dado durante o lançamento de um produto. Adicionalmente, podemos também utilizar os dados de localização associados a essas mensagens de feedback, pois podem gerar subsídios para confirmar se a estratégia de lançamento está sendo bem conduzida, isto é, se está atingindo as metas planejadas, ou ainda tomar ações de ajuste até que se atinja o objetivo proposto.

Com base neste simples exemplo, vamos desenvolver uma aplicação cujo propósito é automatizar o acompanhamento de determinados assuntos no Twitter, utilizando para isto uma API específica e que já está se tornando bastante popular na comunidade Java, a Twitter4J. Assim como qualquer outra API, a Twitter4J possui uma série de funcionalidades úteis e simples de se utilizar, resultando em uma implementação rápida e sem grandes dificuldades.

Deste modo, este artigo propõe a apresentação da API Twitter4J e suas principais características, em conjunto com um tutorial de configuração passo-a-passo. Será apresentado também como utilizar uma conta de desenvolvimento no Twitter Developers, necessária para a integração da biblioteca Java com a base de dados do Twitter. Para entendermos suas principais funcionalidades, serão mostrados exemplos práticos explicados passo-a-passo. Por fim, será desenvolvido um sistema de acompanhamento de popularidade de assuntos previamente selecionados. Com base na medição de popularidade, em função da quantidade de citações das palavras ou assuntos selecionados, será possível descobrir padrões nos dados analisados.

Criando as credenciais de acesso no Twitter Developers

Antes de desenvolvermos nosso próprio sistema para utilizar os recursos que a API Twitter4J oferece, é necessário obter as credenciais de acesso para conseguir a autenticação necessária para estabelecer a comunicação entre a API e o Twitter. Estas credenciais são fornecidas no site Twitter Developers (veja o endereço na seção **Links**).

Devido a questões de segurança e controle, o Twitter Developers liberará uma credencial para cada aplicação ou projeto. Logo, é preciso registrar individualmente todas as aplicações, caso exista mais de uma. Esta organização viabilizada pelo Twitter é bastante útil porque possibilita separar diferentes interfaces de comunicação e cada uma delas pode ter níveis de acesso individualizados. Desta forma, aplicações que por ventura ultrapassem os limites de utilização do Twitter (na seção **Links** há uma referência que apresenta informações sobre os limites atuais), por exemplo, poderão ser isoladas das demais.

Para acessar o Twitter Developers, informe os dados de entrada de uma conta normal no Twitter. Efetuado o login, no canto superior direito, clique na foto/perfil do seu usuário e selecione *My applications*, como mostra a **Figura 1**.

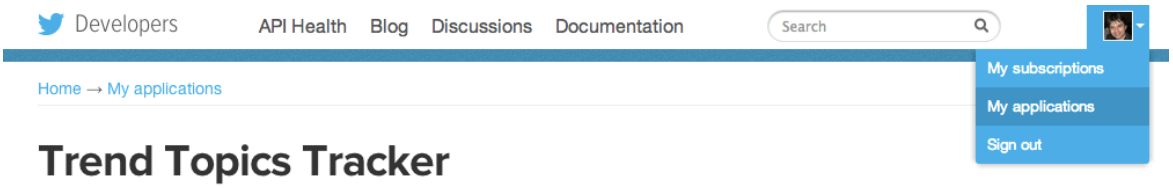

Figura 1. Acesso às aplicações do usuário.

Feito isso, serão mostradas todas as aplicações configuradas para acesso ao Twitter. Por meio deste gerenciamento é possível obter os seguintes dados de autenticação a serem utilizados pela API Twitter4J: *Consumer key*, *Consumer secret*, *Request token URL*, *Authorize URL*, *Access token URL*, *Access token*, *Access token secret*.

Nosso próximo passo será criar uma aplicação no Twitter Developers (*Create New App*) e fornecer os dados cadastrais, como nome, descrição e URL. Um detalhe importante sobre a URL é que apesar de ter seu preenchimento obrigatório, ela não é verificada para o funcionamento da API, permitindo o uso da URL do site ou serviço mesmo que este ainda não esteja on-line.

Após concluir o registro da aplicação, é preciso configurar o nível de permissão de acesso. Por exemplo, se a aplicação somente irá consultar dados, deve ter a permissão somente leitura (*Read Only*). Se necessitar criar novas atualizações de status ou *tweets*, a permissão deve ser do tipo leitura e escrita (*Read and Write*). Caso a aplicação simule todos os privilégios que um usuário regular tem, ou seja, consultar dados, enviar mensagens diretas e postar *tweets*, deverá ter acesso de leitura, escrita e acesso a mensagens diretas (*Read, Write and Access direct messages*). Para configurar o tipo de permissão apropriado, selecione o tipo adequado conforme a **Figura 2**. No nosso exemplo, vamos configurar o acesso mais abrangente, ou seja leitura, escrita e acesso a mensagens diretas, a fim de testarmos todas as principais funcionalidades que serão explicadas posteriormente.

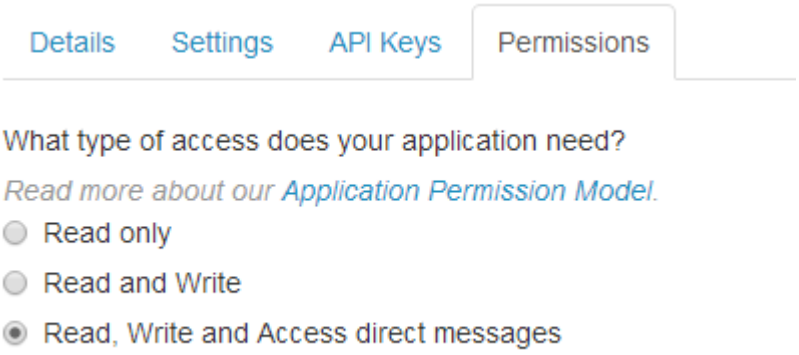

Figura 2. Configurando a permissão de acesso da aplicação.

Depois de termos a aplicação devidamente cadastrada e o nível de permissão de acesso definido, o próximo passo é obter as credenciais para autenticação, necessárias para que a API Twitter4J se comunique com a base de dados do Twitter. Assim, clique na aba *API Keys*, onde serão mostradas as credenciais e os tokens de acesso da aplicação selecionada, como demonstra a **Figura 3**. Como utilizaremos estas informações no desenvolvimento da aplicação, guarde-as.

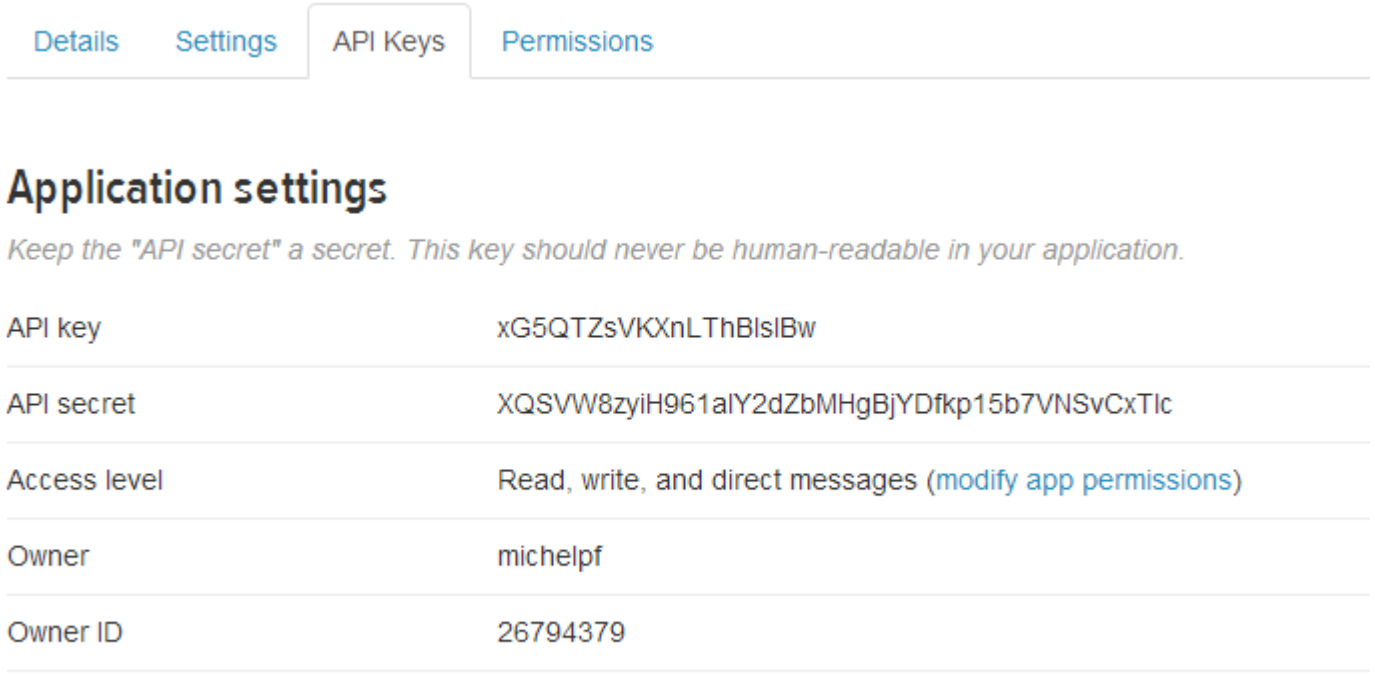

Figura 3. Informações gerais para autenticação da API.

Os tokens de acesso (*Access Token*) são uma medida adicional e obrigatória de segurança. Sem eles não é possível acessar a base de dados do Twitter. Para criar os tokens de acesso, ainda em *API Keys*, clique em *Create access token* e aguarde alguns minutos até que as credenciais sejam disponibilizadas. Ao fazer isso, recomenda-se aguardar por dois minutos e depois atualizar a página para que sejam exibidos os dados de acesso, como demonstra a **Figura 4**. Nesta mesma área também é possível cancelar (*Revoke token access*) ou criar uma nova chave (*Regenerate my access token*), quando for necessário, como por exemplo, se as informações de acesso forem descobertas por terceiros.

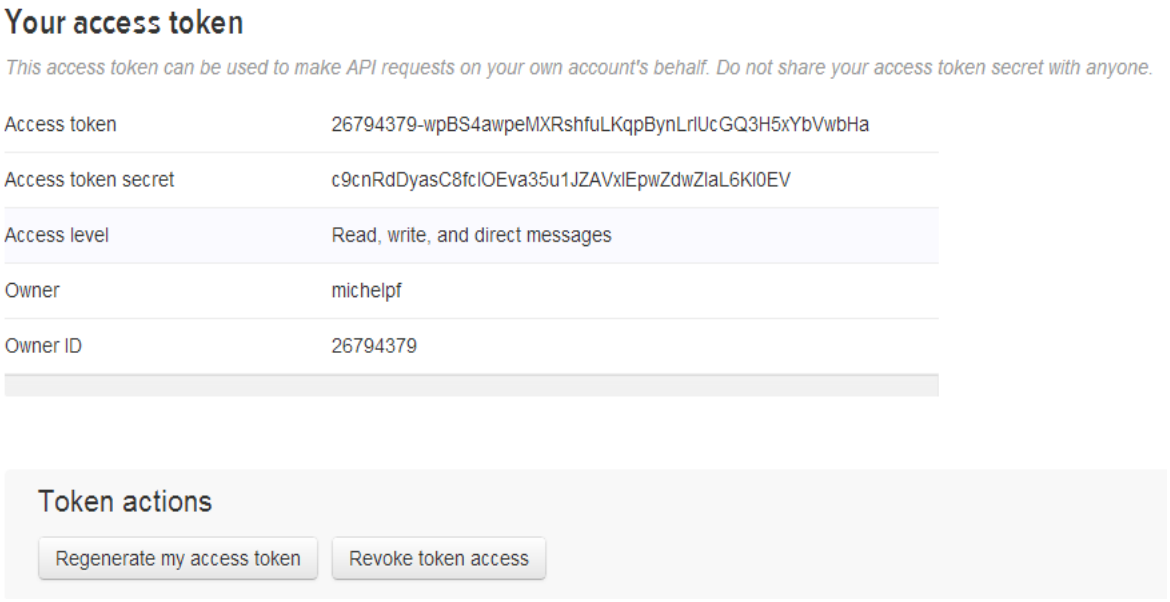

Figura 4. Token de acesso para autenticação da API.

Com os dados de acesso prontos, podemos configurá-los na API Twitter4J e finalmente começar o desenvolvimento de uma aplicação que acessa os dados do Twitter.

Instalando as bibliotecas do Twitter4J

A instalação da API Twitter4J, como qualquer outra instalação de API, é bastante simples. Primeiramente, devemos acessar o site do Twitter4J (veja o endereço na seção **Links**) e baixar a última versão disponível na seção *Downloads*.

Após descompactar o arquivo baixado, você encontrará uma série de arquivos. Deste conjunto, vamos utilizar apenas os seguintes JARs presentes na pasta *lib*: *twitter4j-async-3.0.5.jar*, *twitter4j-core-3.0.5.jar*, *twitter4j-media-support-3.0.5.jar* e *twitter4j-stream-3.0.5.jar*.

Neste ponto já temos todos os requisitos básicos para iniciar o desenvolvimento da aplicação. Então, abra a IDE Eclipse e crie um novo projeto Java (*New Java Project*). Em seguida, precisamos importar os arquivos binários citados anteriormente. Para isso, acesse o menu *Project* > *Properties*. Nas propriedades do projeto, temos que configurar as bibliotecas no Build Path. Assim, acesse a opção *Java Build Path*, clique em *Add External JARs...* (ver **Figura 5**) e depois selecione os arquivos da API.

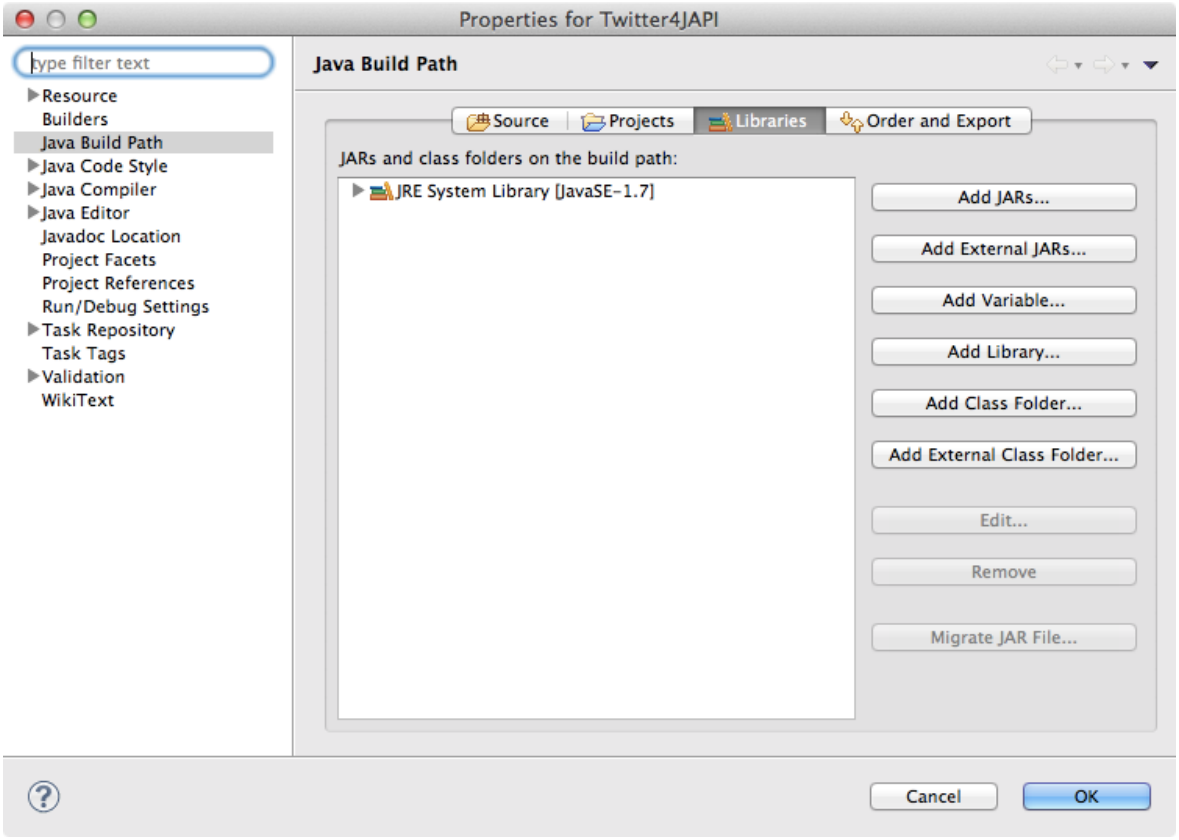

Figura 5. Adicionando as bibliotecas ao projeto.

Apesar não haver regras sobre em qual pasta manter as bibliotecas da API, recomendamos que estas sejam adicionadas na mesma pasta do projeto no Workspace. Assim, caso o projeto seja copiado para outro lugar, as bibliotecas irão junto, não sendo necessário baixá-las novamente.

Com as bibliotecas importadas, podemos iniciar nossa primeira aplicação. Neste exemplo, vamos obter os *tweets* (e seus autores) da *timeline* de um usuário do Twitter.

Dito isso, vamos efetivamente iniciar o desenvolvimento da aplicação. A primeira parte a ser implementada, antes de iniciar qualquer tipo de interação com o Twitter, é a da autenticação de acesso. Para isso, devemos utilizar as credenciais que configuramos anteriormente, que são: *Consumer Key*, *Consumer Secret*, *Access Token* e *Token Secret*. Essas credenciais são definidas, conforme a **Listagem 1**, no objeto **twitter**, por meio dos métodos

setOAuthAccessToken(AccessToken), para os tokens de acesso, e **setOAuthConsumer()**, para *Consumer Key* e *Secret Key*. Com isso teremos as informações de acesso devidamente ajustadas, de acordo com o nível de privilégios concedido previamente.

A próxima etapa será carregar todas as atualizações de status ou *tweets* da *timeline* em uma coleção de valores do tipo **List**. Para isso, chamamos o método **twitter.getHomeTimeline()**, conforme a linha 18, e armazenamos o resultado na coleção **statuses**. Em seguida, para percorrer esta coleção e ler todos os tweets, foi utilizada a estrutura **foreach**. Deste modo foram obtidos os dados de cada *tweet*, como o nome do usuário (**status.getUser().getName()**) e a mensagem associada (**status.getText()**).

Após terminar o desenvolvimento da aplicação, revise as importações utilizadas no código fonte para evitar importações de classes indevidas. Verifique que na **Listagem 1** são apresentadas as importações de todas as classes utilizadas com o intuito de evitar qualquer associação indevida com outras classes de mesmo nome.

Listagem 1. Acessando os tweets da Timeline - Adaptado da documentação do Twitter4J.

```
1   import java.util.List;
2   import twitter4j.Status;
3   import twitter4j.Twitter;
4   import twitter4j.TwitterFactory;
5   import twitter4j.auth.AccessToken;
6
7   public class Main {
8
9     public static void main(String[] args) {
10
11      try {
12        TwitterFactory factory = new TwitterFactory();
13        AccessToken accessToken = loadAccessToken();
14        Twitter twitter = factory.getSingleton();
15        twitter.setOAuthConsumer("xG5QTZsVKXnLThBlslBw",
    "XQSVW8zyiH961alY2dZbMHgBjYDfkp15b7VNSvCxTlc");
16        twitter.setOAuthAccessToken(accessToken);
17
18        List<Status> statuses = twitter.getHomeTimeline();
19        System.out.println("Home Timeline:");
20        for (Status status : statuses) {
21          System.out.println("Usuário: " +status.getUser().getName() + ":" +
22          "Tweet:"+status.getText());
23        }
24      } catch (Exception e) {
25        e.printStackTrace();
26      }
27    }
28
29    private static AccessToken loadAccessToken(){
30      String token = "26794379-EShOPlvIgL0IChdYOlAkuwEFrI5PVfbcZWXa591zY";
31      String tokenSecret = "NghghJuTEAOUAXWBNSGIshcGOXtoc78JiLRKEjGVmUOjJ";
32      return new AccessToken(token, tokenSecret);
33    }

34  }
```

Na **Listagem 2** podemos verificar o resultado da execução da aplicação, onde são mostradas todas as atualizações da *timeline* com o nome do usuário e o *tweet* correspondente. Apesar de neste exemplo apresentarmos apenas estes dois dados, no tópico “Obtendo informações adicionais em cada Tweet” exploraremos outras características associadas, como a contagem de favoritos, *retweets* e geolocalização.

Listagem 2. Registros da Timeline.

```
Home Timeline:
...
Usuário: Vagas.com.br:Tweet:Vagas p/ Assistente de Suporte - TI - FH Consulting - Curitiba/PR -
http://t.co/Gqf47qDPqs
Usuário: Marlos Vidal:Tweet:Nissan anuncia fábrica de motores em Resende (RJ): Em visita às obras
de fábrica da Nissan em Resende (RJ), Ca... http://t.co/eoKtYEI73e
```

```
Usuário: VAGAS.com.br:Tweet:Vagas p/ Gerente Administrativo - RGB Consutoria em RH - Itapetininga/SP
- http://t.co/ZFZvIpVtLw
Usuário: Meio Bit:Tweet:CES 2014: Lenovo anuncia novos AIO com Android, ultrabook e tablet com
Windows 8 Pro http://t.co/A1kbP3mVa1 #HojenoMeioBit
...
```

Um ponto interessante da comunicação da API com o Twitter é que sempre que houver algum problema ou erro em alguma requisição, são enviados links informativos de auxílio. Assim, problemas como falha na autenticação ou mesmo comandos indevidos enviados pela API podem ser facilmente identificados. Um exemplo destas mensagens pode ser verificado na **Listagem 3**, que mostra um erro de autenticação e ainda indica, além de causas prováveis, onde procurar ajuda nos links de discussões para solucionar o problema.

Listagem 3. Mensagem de retorno do Twitter para erro de autenticação.

```
 400:The request was invalid. An accompanying error message will explain why. This is the status
code will be returned during version 1.0 rate limiting(https://dev.twitter.com/pages/rate-limiting).
In API v1.1, a request without authentication is considered invalid and you will get this response.
 message - Bad Authentication data
 code - 215

 Relevant discussions can be found on the Internet at:
       http://www.google.co.jp/search?q=d35baff5 or
       http://www.google.co.jp/search?q=1446301f
 TwitterException{exceptionCode=[d35baff5-1446301f],  statusCode=400,  message=Bad  Authentication
data, code=215, retryAfter=-1, rateLimitStatus=null, version=3.0.5}
```

Explorando os principais recursos da Twitter4J

Com a API Twitter4J é possível realizar qualquer tipo de operação que poderia ser efetuada manualmente pelo usuário no site do Twitter, dentre elas: criar *tweets*, buscar por assuntos com critérios específicos, enviar e visualizar mensagens diretas, entre outras funcionalidades. Sendo assim, nos próximos tópicos apresentaremos exemplos de aplicações para explorar cada uma delas.

Estas funcionalidades permitirão automatizar diversas operações e tornar mais ágil a comunicação com os usuários do Twitter. Por exemplo, é possível desenvolver um sistema que identifique mensagens de reclamação, por meio da busca de temas específicos, e que automaticamente responda estas mensagens solicitando mais informações ou fornecendo algum tipo de feedback. Também é possível que seja estabelecido uma interface com os sistemas próprios da empresa para facilitar e tornar ainda mais ágil este tipo de interação.

Criando um Tweet

Como um dos principais recursos, a Twitter4J pode ser utilizada para automatizar a criação de novos *tweets*. Com isso é viabilizado a uma empresa sinalizar ou comunicar na rede social sobre determinada situação, como um problema, promoção ou um simples aviso. Na **Listagem 4** segue um exemplo de como postar um novo *tweet* utilizando a API.

Um detalhe importante é que as permissões devem estar adequadas para este tipo de funcionalidade, ou seja, a aplicação deve ter, pelo menos, o acesso de escrita (*Write*), conforme configuramos no Twitter Developers anteriormente.

Listagem 4. Código para postar um tweet.

```
1   import twitter4j.Status;
2   import twitter4j.Twitter;
3   import twitter4j.TwitterFactory;
4   import twitter4j.auth.AccessToken;
5
6   public class Main {
7
8     public static void main(String[] args) {
9       try {
10        TwitterFactory factory = new TwitterFactory();
11        AccessToken accessToken = loadAccessToken();
12        Twitter twitter = factory.getSingleton();
13        twitter.setOAuthConsumer("xG5QTZsVKXnLThBlslBw",
    "XQSVW8zyiH961alY2dZbMHgBjYDfkp15b7VNSvCxTlc");
```

```
14        twitter.setOAuthAccessToken(accessToken);
15
16        Status status = twitter.updateStatus("Teste de Atualização com Twitter4J");
17        System.out.println("Tweet postado com sucesso! [" + status.getText() + "].");
18      } catch (Exception e) {
19        e.printStackTrace();
20      }
21    }
22
23    private static AccessToken loadAccessToken(){
24      String token = "26794379-wpBS4awpeMXRshfuLKqpBynLrlUcGQ3H5xYbVwbHa";
25      String tokenSecret = "c9cnRdDyasC8fclOEva35u1JZAVxlEpwZdwZlaL6Kl0EV";
26      return new AccessToken(token, tokenSecret);
27    }
28  }
```

O código apresentado na **Listagem 4** segue o mesmo padrão de todas as aplicações que analisaremos neste artigo. Inicialmente são criados os objetos de autenticação com o Twitter, para prover o acesso. Estes objetos e toda esta parte de configuração é realizado nas linhas 10 a 14. Note que na linha 12 chamamos o método **getSingleton()** para que seja criada apenas uma instância do objeto **twitter** na aplicação. Com isso deixamos de criar múltiplas conexões toda vez que for utilizada alguma operação da API, pois o padrão *Singleton* permite estabelecer apenas uma conexão independente das operações que sejam executadas, mesmo sendo em um ambiente *multithread* (aplicações com processamento paralelo).

Após a autenticação, a etapa seguinte é executar o método **twitter.updateStatus()** para postar um novo *tweet* (vide linha 16). Nesta mesma linha, o retorno da execução do método será armazenado na variável **status**, do tipo **Status**. Este tipo de variável reúne todas as informações associadas ao *tweet*, como o idioma, a localização geográfica, o número de *retweets*, etc. Tais informações enriquecem o conteúdo do *tweet*, pois uma mensagem pode ter um significado amplificado a depender da localização e da popularidade. Por exemplo, mensagens com um grande número de citações e grande popularidade podem indicar uma informação relevante.

Listagem 5. Resultado da execução da aplicação.

```
Tweet postado com sucesso! [Teste de Atualização com Twitter4J].
```

Podemos verificar o resultado da execução da aplicação na **Listagem 5**, que indica que o *tweet* foi publicado com sucesso. Ao visitar a conta do respectivo usuário no Twitter (veja a **Figura 6**), temos a comprovação de que o *tweet* foi postado.

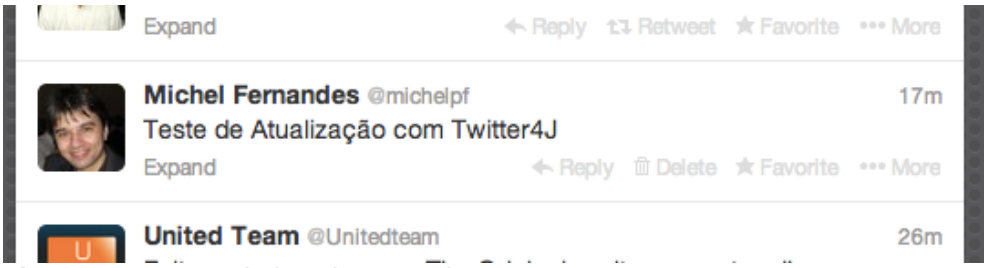

Figura 6. Tweet enviado pela API Twitter4J.

Enviando uma mensagem direta

O envio de uma mensagem direta (*Direct Message*, ou DM) é bem similar a postar um novo *tweet*. A diferença é que o *tweet* é público e a mensagem direta é privada. De modo semelhante ao que fizemos anteriormente, vamos desenvolver uma aplicação para automatizar o envio deste tipo de mensagem.

Vale lembrar que para enviar uma mensagem direta é preciso que o usuário de destino da mensagem seja seguidor do usuário remetente. Devido a isso, para testar essa funcionalidade, devemos escolher um usuário que seja seguidor do usuário que foi utilizado no registro da aplicação pelo Twitter Developers.

Listagem 6. Código para postar uma mensagem direta.

```
1   import twitter4j.DirectMessage;
2   import twitter4j.Twitter;
3   import twitter4j.TwitterFactory;
4   import twitter4j.auth.AccessToken;
5
6   public class Main {
7
8     public static void main(String[] args) {
9
10      try {
11        TwitterFactory factory = new TwitterFactory();
12        AccessToken accessToken = loadAccessToken();
13        Twitter twitter = factory.getSingleton();
14        twitter.setOAuthConsumer("xG5QTZsVKXnLThBlslBw",
    "XQSVW8zyiH961alY2dZbMHgBjYDfkp15b7VNSvCxTlc");
15        twitter.setOAuthAccessToken(accessToken);
16        DirectMessage message = twitter.sendDirectMessage("@JNarezzi", "Teste de DM do artigo
    do Twitter4J.");
17        System.out.println("Sent: " + message.getText() + " to @" +
    message.getRecipientScreenName());
18      } catch (Exception e) {
19        e.printStackTrace();
20      }
21    }
22
23    private static AccessToken loadAccessToken(){
24      String token = "26794379-wpBS4awpeMXRshfuLKqpBynLrlUcGQ3H5xYbVwbHa";
25      String tokenSecret = "c9cnRdDyasC8fclOEva35u1JZAVxlEpwZdwZlaL6Kl0EV";
26      return new AccessToken(token, tokenSecret);
27    }
28  }
```

Para enviar uma mensagem direta, utilizamos o método **twitter.sendDirectMessage()** na linha 16, da **Listagem 6**. Similarmente ao exemplo de postar um novo *tweet*, também utilizamos uma variável que armazenará o retorno do envio da mensagem direta. Essa variável, **message**, é do tipo **DirectMessage**, e nela é possível obter informações básicas acerca do envio da mensagem, como a própria mensagem e a data de envio.

Listagem 7. Resultado do envio de uma mensagem direta.

```
Sent: Teste de DM do artigo do Twitter4J. to @JNarezzi
```

Como resultado, podemos verificar na **Listagem 7** que a mensagem direta foi enviada com sucesso. Da mesma forma, também podemos acessar o site do Twitter e comprovar, de acordo com a **Figura 7**, que a ela foi enviada como esperado.

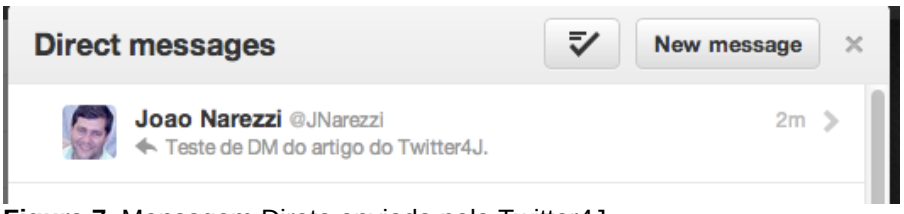

Figura 7. Mensagem Direta enviada pela Twitter4J

Visualizando as mensagens diretas

Por sua vez, para visualizar as mensagens diretas recebidas, precisamos utilizar o método **twitter.getDirectMessages()** e armazenar seu retorno na coleção **mensagens**, do tipo **List**, que é a listagem de todas as mensagens diretas recebidas pelo usuário utilizado na API. Veja a **Listagem 8**.

Para obter informações detalhadas do remetente e do destinatário da mensagem direta, utilize os métodos **mensagem.getSender()** e **mensagem.getRecipient()**, ambos do objeto **mensagem**, do tipo **DirectMessage**. Por meio destes métodos é possível verificar a localização geográfica

(**mensagem.getSender().getLocation()**), o número de seguidores (**mensagem.getSender().getFollowersCount()**), entre outras informações.

Na **Listagem 8** é mostrada a aplicação para visualizar as mensagens diretas. Como podemos verificar, essa implementação também é simples. Assim como fizemos anteriormente, carregamos todas as mensagens em uma coleção do tipo **List**, conforme a linha 18. Depois, listamos cada uma delas exibindo a mensagem, o usuário remetente e a data enviada, utilizando a estrutura **foreach**, da linha 19 à linha 24.

Listagem 8. Aplicação para visualizar mensagens diretas.

```
1   import java.util.List;
2   import twitter4j.DirectMessage;
3   import twitter4j.Twitter;
4   import twitter4j.TwitterFactory;
5   import twitter4j.auth.AccessToken;
6
7   public class Main {
8
9     public static void main(String[] args) {
10
11      try {
12        TwitterFactory factory = new TwitterFactory();
13        AccessToken accessToken = loadAccessToken();
14        Twitter twitter = factory.getSingleton();
15        twitter.setOAuthConsumer("xG5QTZsVKXnLThBlslBw",
    "XQSVW8zyiH961alY2dZbMHgBjYDfkp15b7VNSvCxTlc");
16        twitter.setOAuthAccessToken(accessToken);
17
18        List<DirectMessage> mensagens = twitter.getDirectMessages();
19        for (DirectMessage mensagem : mensagens)
20        {
21          System.out.println("Mensagem: "+mensagem.getText());
22          System.out.println("Remetente: "+mensagem.getSenderScreenName());
23          System.out.println("Data: " +mensagem.getCreatedAt());
24        }
25      } catch (Exception e) {
26        e.printStackTrace();
27      }
28    }
29
30    private static AccessToken loadAccessToken(){
31      String token = "26794379-wpBS4awpeMXRshfuLKqpBynLrlUcGQ3H5xYbVwbHa";
32      String tokenSecret = "c9cnRdDyasC8fclOEva35u1JZAVxlEpwZdwZlaL6Kl0EV";
33      return new AccessToken(token, tokenSecret);
34    }
35  }
```

Como resultado, obtemos a **Listagem 9**, onde são mostradas todas as mensagens diretas recebidas pelo usuário configurado na API, com as informações do usuário que enviou, a mensagem e a data de envio. Por questões de privacidade, foi removido o conteúdo e o remetente das mensagens.

Listagem 9. Resultado da listagem de mensagens diretas recebidas.

```
...
Remetente: XXXXXXXX
Data: Mon Feb 25 07:36:37 BRT 2013
Mensagem: AAAAAAAAAA
Remetente: YYYYYYYY
Data: Fri Feb 22 18:43:13 BRT 2013
Mensagem: BBBBBBBBBB
Remetente: UUUUUUUUU
Data: Wed Feb 06 19:59:31 BRST 2013
Mensagem: ZZZZZZZZZZ
Remetente: CCCCCCCCC
Data: Tue Feb 05 20:03:43 BRST 2013
...
```

Buscando por assuntos e por temas específicos

Esta é a parte mais interessante e o foco principal deste artigo. Tal importância se deve ao fato que a busca e a descoberta de informações úteis em um universo de dados tão extenso é cada vez mais

complicada, frente a quantidade/diversidade dos dados. Deste modo, este cenário necessita de meios cada vez mais aperfeiçoados de pesquisa para viabilizar a extração de informações relevantes.

Uma estratégia para descobrir novas informações sobre uma determinada área pode ser feita pela monitoração de palavras chave associadas. Por exemplo, se o objetivo é monitorar e descobrir novas informações sobre as empresas fabricantes de veículos, as palavras chave poderiam ser o nome de cada uma destas empresas. A monitoração destas palavras, associada à popularidade de cada uma, isto é, a quantidade de *tweets* e *retweets*, pode indicar uma nova informação ao comparar estes dados com a base histórica em períodos anteriores.

Devido às limitações da API do Twitter, realizar uma busca sem definir critérios ou filtros específicos, como data inicial e final, limitam os resultados em até 15 *tweets* por pesquisa. Para não termos o resultado das buscas limitada, recomendamos que sempre sejam utilizados filtros. Se o objetivo for monitorar determinados assuntos diariamente, podemos, por exemplo, limitar as datas de início e fim.

Em função da grande quantidade de dados, como também para melhorar a precisão dos dados que se pretende buscar, é recomendado utilizar a maior quantidade de critérios ou filtros de busca possível.

A seguir são apresentados alguns recursos que podem ser aplicados no objeto **Query** para refinar o resultado das buscas:

- Idioma: realizar a busca pelo idioma desejado torna os resultados menos dispersos, pois se concentra em termos e palavras-chave somente no idioma selecionado. Definimos o idioma pelo método **Query.setLang(String lang)**, onde **lang** é o idioma. A relação de todos os idiomas e suas abreviações está reunida em uma página indicada na seção **Links**;
- Tipo dos resultados: neste tipo de critério podemos filtrar por *tweets* postados recentemente, como também pelos mais populares. Esta definição é realizada pelo método **Query.setResultType(String resultType)**, onde **resultType** pode ser **Query.MIXED**, valor padrão que reúne os resultados recentes e populares, **Query.RECENT**, para filtrar apenas os resultados recentes, ou **Query.POPULAR**, que filtra apenas os resultados com maior popularidade;
- Desde uma data específica: quando é necessário buscar por algo a partir de uma determinada data. Para este filtro, chamamos o método **Query.setSince(String data)** passando a data no formato "ano/mês/dia", por exemplo: 2014/01/14;
- Até uma data específica: quando é necessário buscar até uma data determinada. Para este filtro, chamamos o método **Query.setUntil(String data)** passando a data no formato "ano/mês/dia";
- Limitar por uma região em específico: quando for necessário filtrar os resultados por uma determinada região geográfica. Fornece a possibilidade de determinar um ponto geolocalizado e, a partir dele, traçar um raio e assim determinar uma região específica. Para realizar esta busca, devemos chamar o método **Query.setGeoCode(GeoLocation geoLocation, Double radius, String unit)**, onde **GeoLocation** é um objeto de geolocalização representado pelas coordenadas geográficas de latitude e longitude, **radius** define o raio de limitação da região de busca, cuja unidade de comprimento, representado pelo **unit**, pode ser **Query.MILES** para milhas e **Query.KILOMETERS** para quilômetros.

Além desses recursos, precisamos definir as palavras-chave de buscas. Tal como as buscas no Google, o Twitter também disponibiliza certas regras para viabilizar uma consulta mais detalhada. Assim, para realizar a busca por palavra-chave, por exemplo, é necessário passá-la ao construtor da classe **Query**, como: **Query busca = new Query(String query)**, ou utilizar o método **Query.setQuery(String query)**, onde **query** é o termo de busca.

A **Tabela 1** apresenta os principais operadores e regras para realizar as buscas no Twitter.

Exemplo de Consulta	Descrição
Palavra1 Palavra2	Operador padrão, busca por *tweets* com a Palavra1 e a Palavra2
“Palavra1 Palavra2”	Busca por *tweets* com a frase “Palavra1 Palavra2”
Palavra1 OR Palavra2	Busca por *tweets* com a Palavra1 ou a Palavra2
Palavra1 -Palavra2	Busca por *tweets* que contêm a Palavra1 e que não contêm a Palava2
#Palavra1	Busca por *tweets* com a *hashtag* Palavra1
from:Usuário1	Busca por *tweets* do usuário Usuário1
to:Usuário2	Busca por *tweets* para o Usuário2
@Usuário3	Busca por *tweets* com menção ao Usuário3
:)	Busca por *tweets* que contenham termos ou palavras com sentido positivo definido na engine de aprendizado do Twitter (por exemplo, bons *reviews* de um produto)
:(	Busca por *tweets* que contenham termos ou palavras com sentido negativo definido na engine de aprendizado do Twitter

Tabela 1. Lista de operadores de pesquisa do Twitter (Fonte Development of Twitter Application #7 – Search)

Na **Listagem 10** apresentamos uma pesquisa realizada pela palavra-chave “java” na qual desejamos recuperar apenas os *tweets* que estejam no intervalo de 01/01/2014 a 10/01/2014. Com esses tipos de filtros é possível verificar se houve algum aumento significativo da popularidade desta palavra-chave e posteriormente refinar a pesquisa para descobrir a origem desse aumento.

Listagem 10. Buscando por palavra-chave e utilizando filtro por data.

```
 import twitter4j.Query;
 import twitter4j.QueryResult;
 import twitter4j.Status;
 import twitter4j.Twitter;
 import twitter4j.TwitterFactory;
 import twitter4j.auth.AccessToken;

 public class Main {

       public static void main(String[] args) {

              try {
                     TwitterFactory factory = new TwitterFactory();
                     AccessToken accessToken = loadAccessToken();
                     Twitter twitter = factory.getSingleton();
                     twitter.setOAuthConsumer("xG5QTZsVKXnLThBlslBw",
"XQSVW8zyiH961alY2dZbMHgBjYDfkp15b7VNSvCxTlc");
                     twitter.setOAuthAccessToken(accessToken);

                     Query query = new Query("java");
                     query.setSince("2014-01-01");
                     query.setUntil("2014-01-10");
                     QueryResult result;
                     int contador=0;


                     result = twitter.search(query);

                     while (result.hasNext())
                     {
                            query = result.nextQuery();

                            for (Status status : result.getTweets()) {
                                   contador++;
                                   System.out.println("@" + status.getUser().getScreenName() + ":"
+ status.getText());
                            }

                            result = twitter.search(query);

                     }
```

```
                    System.out.println("Tag java:"+contador+" tweets");


                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        private static AccessToken loadAccessToken(){
                String token = "26794379-wpBS4awpeMXRshfuLKqpBynLrlUcGQ3H5xYbVwbHa";
                String tokenSecret = "c9cnRdDyasC8fclOEva35u1JZAVxlEpwZdwZlaL6Kl0EV";
                return new AccessToken(token, tokenSecret);
        }
}
```

Como resultado, obtivemos na **Listagem 11** todos os *tweets* com a palavra-chave "java", bem como a contagem do número de ocorrências, que foi de 90 no período especificado, levando em conta toda a base do Twitter. Esta informação indica um parâmetro de popularidade que poderia ser comparado com outras palavras-chave, como por exemplo, php, vb.net, etc. para medir a popularidade das linguagens de programação, sob o ponto de vista do Twitter.

Listagem 11. Resultado da busca por palavra-chave e filtro por data.

```
...
@mosesjones:All that shiny hype… but if you want a job?  Learn #Java http://t.co/ZNVi1xOycu
@annekathrine11:@bakerrr4 brb goin to java, red box, and gettin Chinese 2 go. B @ ur place n 5
@NevaehAdderiy:Does java hasten rob tax? feasting tie-up is the limiting put to silence: mGZlQFjd
...
Tag java:90 tweets de 2014-01-01 até 2014-01-10.
```

Obtendo informações adicionais em cada Tweet

As informações disponíveis nas características associadas aos *tweets* podem ser tão importantes quanto a própria mensagem. Além da mensagem, é possível obter diversos outros dados relacionados que auxiliam a enriquecer significativamente o conteúdo da mensagem. As seguir são apresentados os mais relevantes:

- Usuário: permite dar mais ênfase em um *tweet* baseado em usuários influentes ou com contas verificadas;
- Data e hora de criação: permite filtrar as informações pela data de interesse;
- Número de favoritos: *tweets* marcados como favoritos podem ter importância maior, se comparados com outros sem esse tipo de marcação. Podem sinalizar como um indicativo de popularidade;
- Geolocalização: a localização do *tweet* pode auxiliar bastante em questões de feedback regionalizados;
- Endereço físico de um determinado local: idem ao item anterior, pois permite obter feedbacks regionalizados por um determinado endereço;
- Número de *retweets*: indica uma maior importância, tal como a marcação de favoritos. Um detalhe relevante é que a contagem de *retweets* pode sinalizar mensagens de maior interesse devido ao grau de popularidade associado.

Para demonstrar como obter essas informações, vamos modificar a aplicação exemplo analisada anteriormente. Essa modificação possibilitará explorar melhor cada *tweet* encontrado, exibindo as informações do usuário que criou *tweet*, a mensagem, quando foi criada, o número de favoritos, o número de *retweets*, sua geolocalização e seu lugar ou endereço.

Na **Listagem 12**, verificamos que os dados associados bem como o próprio *tweet* ficam no mesmo objeto **Status**. Na linha 30 definimos a estrutura **foreach** para ler cada *tweet* retornado da busca e armazenado na lista **QueryResult**.

Os dados presentes no objeto **Status** são retornados pelos métodos específicos apresentados a seguir:

- Linha 33: **Status.getUser().getScreenName()**, retorna o nome do usuário;

- Linha 34: **Status.getText()**, retorna a mensagem;
- Linha 35: **Status.getCreatedAt()**, retorna a data de criação ou publicação;
- Linha 36: **Status.getFavoriteCount()**, retorna o número de favoritos marcados no *tweet*;
- Linha 37: **Status.getGeoLocation()**, retorna os dados de geolocalização do lugar de onde o *tweet* foi publicado, ou seja, as coordenadas de latitude e longitude.
- Linha 38: **Status.getPlace()**, retorna os dados de endereço do lugar de onde o *tweet* foi publicado;
- Linha 39: **Status.getRetweetCount()**, retorna o número de *retweets*.

Listagem 12. Obtendo informações detalhadas de cada tweet.

```
1   import twitter4j.Query;
2   import twitter4j.QueryResult;
3   import twitter4j.Status;
4   import twitter4j.Twitter;
5   import twitter4j.TwitterFactory;
6   import twitter4j.auth.AccessToken;
7
8   public class Main {
9
10    public static void main(String[] args) {
11
12      try {
13        TwitterFactory factory = new TwitterFactory();
14        AccessToken accessToken = loadAccessToken();
15        Twitter twitter = factory.getSingleton();
16        twitter.setOAuthConsumer("xG5QTZsVKXnLThBlslBw",
    "XQSVW8zyiH961alY2dZbMHgBjYDfkp15b7VNSvCxTlc");
17        twitter.setOAuthAccessToken(accessToken);
18
19        Query query = new Query("java");
20        query.setSince("2014-01-01");
21        query.setUntil("2014-01-10");
22        QueryResult result;
23        int contador=0;
24        result = twitter.search(query);
25
26        while (result.hasNext())
27        {
28          query = result.nextQuery();
29
30          for (Status status : result.getTweets()) {
31            contador++;
32
33            System.out.println("Usuário: " + status.getUser().getScreenName());
34            System.out.println("Mensagem: " + status.getText());
35            System.out.println("Data de Criação: " + status.getCreatedAt());
36            System.out.println("Número de Favoritos: " + status.getFavoriteCount());
37            System.out.println("Geolocalização: " + status.getGeoLocation());
38            System.out.println("Lugar: " + status.getPlace());
39            System.out.println("Número de Retweets: " + status.getRetweetCount());
40          }
41          result = twitter.search(query);
42        }
43        System.out.println("Tag java:"+contador+" tweets de 2014-01-01 até 2014-01-10.");
44      } catch (Exception e) {
45        e.printStackTrace();
46      }
47    }
48
49    private static AccessToken loadAccessToken(){
50      String token = "26794379-wpBS4awpeMXRshfuLKqpBynLrlUcGQ3H5xYbVwbHa";
51      String tokenSecret = "c9cnRdDyasC8fclOEva35u1JZAVxlEpwZdwZlaL6Kl0EV";
52      return new AccessToken(token, tokenSecret);
53    }
54  }
```

Os resultados obtidos pela aplicação podem ser visualizados na **Listagem 13**. Note que em alguns *tweets* as informações de geolocalização ou localização não estão disponíveis, pois dependem do usuário ativar tais configurações em seus dispositivos.

Listagem 13. Resultado da execução do programa da Listagem 12.

```
...
Usuário: r_mcnair721
Mensagem: Idc if its cold out, you can never go wrong with a Java chip frappucino
Data de Criação: Thu Jan 09 21:59:53 BRST 2014
Número de Favoritos: 1
Geolocalização: GeoLocation{latitude=34.68504, longitude=-84.48321}
Lugar: PlaceJSONImpl{name='Ellijay', streetAddress='null', countryCode='US', id='0bfcd4103d0e5ab6',
country='United States', placeType='city',
url='https://api.twitter.com/1.1/geo/id/0bfcd4103d0e5ab6.json', fullName='Ellijay, GA',
boundingBoxType='Polygon', boundingBoxCoordinates=[[Ltwitter4j.GeoLocation;@2e21712e],
geometryType='null', geometryCoordinates=null, containedWithIn=[]}
Usuário: miesehati
Mensagem: RT @jayteroris: Road to Java Jazz #10 Februari 2014 ##SRUDUKFOLLOW | @karmanmove
@faisal_gudeg @dennyarivian http://t.co/bSXu1Za1U1
Data de Criação: Thu Jan 09 21:50:47 BRST 2014
Número de Favoritos: 0
Geolocalização: null
Lugar: null
Número de Retweets: 3
...
Tag java:90 tweets de 2014-01-01 até 2014-01-10.
```

Caso prático: acompanhando tendências de forma automatizada

Uma proposta interessante de uso das informações do Twitter é para o acompanhamento e monitoramento de determinados assuntos de forma automatizada, permitindo que essa identificação seja realizada rapidamente e, como consequência, a tomada de decisões possa ser mais eficiente.

Um exemplo simples desse tipo de monitoramento pode ser o acompanhamento da popularidade da hashtag #selecaobrasileira, ao pensar na realização da Copa do Mundo. A expectativa é que a popularidade desse termo aumente com a proximidade do evento.

Como demonstra a **Listagem 14**, foi desenvolvido um sistema que busca diariamente na base de dados do Twitter o número de ocorrências de um termo específico. No exemplo, definimos como critério a hashtag #selecaobrasileira e utilizamos os filtros de data de início e de fim para delimitar as contagens do termo diariamente. Quando o número de ocorrências for superior a 50, todos os *tweets* serão exibidos, como uma forma de verificar os motivos de um incremento repentino no número de ocorrências.

A implementação do sistema seguiu o mesmo padrão dos exemplos anteriores. As mudanças relevantes estão relacionadas à incorporação do código para determinar as datas de início e fim (linhas 21 a 24) como critérios de busca dos *tweets*. Deste modo, utilizamos a estrutura de repetição **while** para realizar a busca em cada data, que se inicia na data definida em **gCalendarInicio** e termina na data atual, **gCalendarAgora**.

Listagem 14. Monitoramento de palavra-chave automatizado via Twitter

```
1.  import java.text.SimpleDateFormat;
2.  import java.util.GregorianCalendar;
3.  import twitter4j.GeoLocation;
4.  import twitter4j.Query;
5.  import twitter4j.QueryResult;
6.  import twitter4j.Status;
7.  import twitter4j.Twitter;
8.  import twitter4j.TwitterFactory;
9.  import twitter4j.auth.AccessToken;
10.
11. public class Main {
12.
13.   public static void main(String[] args) {
14.     try {
15.       TwitterFactory factory = new TwitterFactory();
16.       AccessToken accessToken = loadAccessToken();
17.       Twitter twitter = factory.getSingleton();
18.       twitter.setOAuthConsumer("xG5QTZsVKXnLThBlslBw",
    "XQSVW8zyiH961alY2dZbMHgBjYDfkp15b7VNSvCxTlc");
19.       twitter.setOAuthAccessToken(accessToken);
20.
21.       GregorianCalendar gCalendarInicio = new GregorianCalendar(2014, 5, 7);
```

```
22.       System.out.println(gCalendarInicio.getTime());
23.       GregorianCalendar gCalendarAgora = new GregorianCalendar();
24.       System.out.println(gCalendarAgora.getTime());
25.
26.       SimpleDateFormat formatador = new SimpleDateFormat("yyyy-MM-dd");
27.
28.       while (gCalendarInicio.before(gCalendarAgora)){
29.
30.         Query query = new Query("#selecaobrasileira");
31.         GregorianCalendar gCalendarAnterior = new GregorianCalendar();
32.         gCalendarAnterior.setTime(gCalendarInicio.getTime());
33.         gCalendarAnterior.add(GregorianCalendar.DAY_OF_MONTH, -1);
34.
35.         query.setSince(formatador.format(gCalendarAnterior.getTime()));
36.         query.setUntil(formatador.format(gCalendarInicio.getTime()));
37.         QueryResult result;
38.         int contador=0;
39.
40.         result = twitter.search(query);
41.
42.         while (result.hasNext()) {
43.           query = result.nextQuery();
44.           for (Status status : result.getTweets()) {
45.             contador++;
46.             if (contador>50){
47.               System.out.println("Usuário: " + status.getUser().getScreenName());
48.               System.out.println("Mensagem: " + status.getText());
49.               System.out.println("Data de Criação: " + status.getCreatedAt());
50.               System.out.println("Número de Favoritos: " + status.getFavoriteCount());
51.               System.out.println("Geolocalização: " + status.getGeoLocation());
52.               System.out.println("Lugar: " + status.getPlace());
53.               System.out.println("Número de Retweets: " + status.getRetweetCount());
54.             }
55.           }
56.           result = twitter.search(query);
57.         }
58.
59.         System.out.println(formatador.format(gCalendarInicio.getTime())+":
    #selecaobrasileira "+contador+" tweets.");
60.         gCalendarInicio.add(GregorianCalendar.DAY_OF_MONTH, 1);
61.       }
62.     } catch (Exception e) {
63.       e.printStackTrace();
64.     }
65.   }
66.
67.   private static AccessToken loadAccessToken(){
68.     String token = "26794379-wpBS4awpeMXRshfuLKqpBynLrlUcGQ3H5xYbVwbHa";
69.     String tokenSecret = "c9cnRdDyasC8fclOEva35u1JZAVxlEpwZdwZlaL6Kl0EV";
70.     return new AccessToken(token, tokenSecret);
71.   }
72. }
```

Os resultados obtidos na **Listagem 15** indicam que pode ter ocorrido algum fato incomum no dia 08/05/14 relacionado com a palavra-chave #selecaobrasileira, devido ao aumento expressivo do número de *tweets* contendo esta *hashtag*. Tal aumento no número de *tweets* se deve ao fato que foi neste dia que ocorreu a divulgação dos atletas convocados para defender o Brasil na Copa do Mundo.

Listagem 15. Resultado do monitoramento do Twitter por #selecaobrasileira.

```
 2014-05-07: #selecaobrasileira 45 tweets.
 2014-05-08: #selecaobrasileira 330 tweets.
 ...
 Usuário: DeOlhoJornal
 Mensagem: Felipão apresenta a seleção que representará o Brasil na Copa #Copa #Mundial2014
#SeleçãoBrasileira http://t.co/KcZ09BotH4
 Data de Criação: Wed May 07 17:14:35 BRT 2014
 Número de Favoritos: 0
 Geolocalização: null
 Lugar: null
 Número de Retweets: 0
 ...
 2014-05-09: #selecaobrasileira 60 tweets.
 ...
```

```
 Usuário: julien_menez
 Mensagem: Confira os 23 lucky ones da #SelecaoBrasileira para a @fifaworldcup_pt
http://t.co/EBwyh8RW5T
 Data de Criação: Wed May 07 17:22:33 BRT 2014
 Número de Favoritos: 0
 Geolocalização: GeoLocation{latitude=-23.61093095, longitude=-46.69452058}
 Lugar: PlaceJSONImpl{name='Sao Paulo', streetAddress='null', countryCode='BR',
id='68e019afec7d0ba5', country='Brasil', placeType='city',
url='https://api.twitter.com/1.1/geo/id/68e019afec7d0ba5.json', fullName='Sao Paulo, Sao Paulo',
boundingBoxType='Polygon', boundingBoxCoordinates=[[Ltwitter4j.GeoLocation;@6e0e0380],
geometryType='null', geometryCoordinates=null, containedWithIn=[]}
 ...
 2014-05-10: #selecaobrasileira 30 tweets.
 2014-05-11: #selecaobrasileira 30 tweets.
```

Conclusões

A API Twitter4J representa um grande avanço para a construção de aplicações que se comunicam com o Twitter. Seu uso torna o desenvolvimento de aplicações desse tipo cada vez mais fácil, pois foi projetada para fornecer as funções do Twitter de forma simples, possibilitando ao desenvolvedor a preocupação apenas com as regras do negócio, como a análise dos dados e geração de informação. Dito isso, continue explorando os recursos desta API. Certamente você conseguirá, ao menos, implementar funcionalidades curiosas para suas aplicações.

Links

Página da API Twitter4J.
http://twitter4j.org/en/index.html
Página do site Twitter Developers.
https://dev.twitter.com/
Informações sobre uso e limites de utilização de interfaces API com o Twitter.
https://dev.twitter.com/docs/rate-limiting/1.1
Listagem dos códigos dos idiomas no padrão ISO 639-1.
http://en.wikipedia.org/wiki/ISO_639-1
Material explicativo sobre os operadores de busca no Twitter.
http://www.slideshare.net/onlyjiny/development-of-twitter-application-7-search
Artigo da Revista Veja sobre Big Data.
http://clippingmp.planejamento.gov.br/cadastros/noticias/2013/12/9/a-solucao-no-que-nao-se-ve

Michel Pereira Fernandes (*michelpf@gmail.com*) é Mestre em Inteligência Artificial pelo Centro Universitário da FEI, docente dos cursos de Análise e Desenvolvimento de Sistemas e Automação Industrial na Universidade Paulista (UNIP), docente do curso de pós graduação MBA em Desenvolvimento de Soluções Corporativas Java e Soluções de Mobilidade na Faculdade de Informática e Administração Paulista (FIAP) e atuando como Coordenador de Projetos de Inovação em grandes empresas de telecomunicações como a Vivo e atualmente na Nextel. Trabalha com Java há oito anos fornecendo soluções automatizadas na área de telecomunicações.