Sử dụng Node.js để tương tác với Graph API Facebook

25/03/2017 / Bởi Đinh Thiên Phúc / trong Node.js

Dưới đây là 1 số ngữ cảnh khi sử dụng Node.js để kết nối với API Graph của Facebook:

Nhược:

- Facebook có SDK Javascript riêng để tương tác với chính Graph API, tuy nhiên chỉ cho Javascript phía client. **Không có SDK Node.js**!
- Mặc dù có 1 số gói NPM cho phép làm việc với Graph API của Facebook, tuy nhiên **chẳng có package nào đủ hoàn thiện như các API cho các nền tảng mạng xã hội khác**, như Twit cho Twiter API. Cái ổn nhất là facebook-node-sdk, nhưng tôi lại không quá tin tưởng nó, nhất là cho các ứng dụng kinh doanh quan trọng.

- Tài liệu Graph API **không có ví dụ nào sử dụng vanilla Javascript** (mặc dù có 1 vài ví dụ sử dụng SDK phía client-side).
- Thời điểm tôi viết bài (12/3/2017), có **1 số blog và tutorial đã viết về topic này**.

Ưu:

- Nếu bạn có 1 vài ví dụ, **sử dụng Graph API Facebook với Node.js khá vui**.
- Bài viết này có rất nhiều **các ví dụ hữu ích sử dụng Node.js/Express**.

Nội dung

1. Xác thực & Access Token.
2. Truy vấn route Search.
3. Truy vấn 1 đối tượng Graph đơn.
4. Truy vấn trang người dùng quản lý.
5. Post nội dung.

Những nội dung trên dựa theo ứng dụng của tôi. Ví dụ, phần lớn mọi người gần như không cần lấy danh sách các trang người dùng quản lý. Mặc dù thế, các hành động liệt kê dưới đây có thể được sử dụng như 1 model cho các request.

Có 1 nhu cầu chung mà bài viết này không đề cập đến chính là làm việc với các response pagination. Tuy nhiên công việc pagination cũng không quá khó khăn. Đây là tài liệu cho việc đó.

Như thường lệ, nếu bạn có câu hỏi hay feedback, hãy liên lạc với tôi qua Twitter hay qua Contact Form.

Thư viện AJAX

Tất cả các ví dụ dưới đều sử dụng request-promise, do đó hãy chắc chắn rằng đã chạy dòng lệnh cài đặt:

```
npm install --save request request-promise
```

và import nó:

```
const request = require('request-promise');
```

Node framework

Như tôi đã đề cập, trong các ví dụ dưới tôi sử dụng framework Node.js Express.js.

1. Xác thực & Access Token

> *Để bắt đầu, bạn cần 1 access token thu được thông qua Facebook authentication.* ***Nếu bạn đã có sẵn 1 access token, bạn có thể bỏ qua phần này và tiếp tục ở phần kế tiếp.*** *Nếu bạn chưa có, hãy đọc tiếp*

Thông thường, 1 ứng dụng sẽ thu được 1 access token thông qua thủ tục xác thực. Passport.js là 1 package xác thực rất tuyệt dành cho ứng dụng Node.js.

Việc xác thực Facebook bằng Passport.js đi kèm với 1 ứng dụng demo rất hữu ích, sử dụng Express.js. Nếu bạn cần tìm hiểu về Passport.js, hãy bắt đầu với repository này.

Lấy token

Không may cho lắm, việc xác thực không phải là việc dễ dang, nhanh chóng. **Đầu tiên bạn phải có 1 ứng dụng đã đăng kí với Facebook**. Hướng dẫn bạn có thể đọc ở đây.

Một khi bạn đã có 1 ứng dụng đã đăng kí, bạn cần sử dụng `app_id` và `app_secret` ở file config trong ứng dụng xác thực của bạn. Chỉnh lại code trong ứng dụng demo để câu lệnh `console.log()` trả về access token. Bạn sẽ cần token cho các request bên dưới.

Lấy POST permission

Không phải tất cả các access token tạo ra đều như nhau. Nếu bạn định post đến feed của người dùng, hay đến 1 trang được quản lý bởi người dùng, bạn cần cụ thể hóa nó với `'publish_actions'` và `'manage_pages'` permission thông qua 1 đối tượng scope trong xác thực.

Khi bạn yêu cầu các permission này, 1 người dùng đang đăng nhập sẽ thấy 1 prompt khởi động đang yêu cầu các permission đó. Nếu người dùng click vào nút ok, 1 acces token sẽ được sinh ra với permisssion.

Dưới đây là 1 request các permission publish chứa đối tượng với 1 thuộc tính scope trong route `'/login/facebook'`:

```
app.get('/login/facebook',
  passport.authenticate('facebook', {
    scope: ['publish_actions', 'manage_pages']
  }
));
```

Nếu bạn không thêm 1 đối tượng với thuộc tính `scope`, token trả về sẽ chỉ có permission `'public_profile'`.

Bạn có thể sẽ cần các permission khác, như `'user_about_me'`, `'user_photos'` hay `'user_feed'`, tuy nhiên đoạn code trên là đủ để bạn publish. Hãy xem trang này để xem thêm những nét chính về các permission khác nhau mà bạn có thể request trong quá trình xác thực.

Quá trình review ứng dụng

Với việc không submit ứng dụng tới 1 quá trình review ứng dụng chính thức của Facebook, đoạn mã phía trên sẽ chỉ cho bạn truy cập `'public_profile'`. Để request đến các permission khác, bạn cần chứng minh rằng bạn là lập trình viên có ý tốt, với 1 ứng dụng có ý tốt.

Để thu được access token với các permission, bạn phải điền vào phần "App Review" của trang cài đặt ứng dụng trên website Facebook Developer.

Quá trình review sẽ yêu cầu bạn miêu tả ứng dụng, submit 1 video về cách mà ứng dụng của bạn được sử dụng.

Hãy nhớ rằng, thủ tục để lấy 1 access token không liên quan đến việc tương tác với Graph API. Lý do duy nhất tôi đề cập đến nó chính là bạn sẽ thu được access token (có thể cùng với permission), như ví dụ code trên đã require 1 lần.

2. Graph API Search

Hãy bắt đầu phần này 1 khi bạn đã có access token. Chúng ta sẽ tìm cách truy vấn người dùng và các trang thông qua Facebook url: `https://graph.facebook.com/v2.8/search`. Với các đối tượng khác thì cũng tương tự. Dưới đây là 1 danh sách các đối tượng mà bạn có thể tìm:

- `user`
- `page`
- `event`
- `group`
- `place`
- `placetopic`

Route tìm kiếm phía dưới là 1 POST endpoint, yêu cầu 1 payload. Đối tượng payload có 2 thuộc tính: 1 thuộc tính search term (`queryTerm`), 1 thuộc tính search type (`searchType`).

Mặc dù route của Node/Express là 1 POST route, request ta gửi đến Graph API là GET.

Với ứng dụng mà chúng ta đang xây dựng, giá trị gán vào `searchType` nên là `'page'` hoặc `'user'`, và mỗi type sẽ trả về các thuộc tính khác nhau. Thực tế, `searchType` có thể là bất cứ đối tượng nào được liệt kê phía trên.

Hãy kiểm tra tài liệu ở phần đối tượng User và Page để xem các thuộc tính gì ứng với từng đối tượng, có thể request được hay không.

```
const payload = {
  queryTerm: 'Fiat',
  searchType: 'page'
}
```

Route:

```
const request = require('request-promise');

module.exports = (app) => {

  // you'll need to have requested 'user_about_me' permissions
  // in order to get 'quotes' and 'about' fields from search
  const userFieldSet = 'name, link, is_verified, picture';
  const pageFieldSet = 'name, category, link, picture, is_verified'


  app.post('/facebook-search', (req, res) => {
    const  { queryTerm, searchType } = req.body;

    const options = {
      method: 'GET',
      uri: 'https://graph.facebook.com/search',
      qs: {
        access_token: config.user_access_token,
        q: queryTerm,
        type: searchType,
        fields: searchType === 'page' ? pageFieldSet : userFieldSet
      }
    };

    request(options)
      .then(fbRes => {
// Search results are in the data property of the response.
// There is another property that allows for pagination of results.
// Pagination will not be covered in this post,
// so we only need the data property of the parsed response.
        const parsedRes = JSON.parse(fbRes).data;
        res.json(parsedRes);
      })
  });
}
```

Chú ý thuộc tính `fields` và các thể hiện `userFieldSet`, `pageFieldSet`. Khi tìm kiếm các trang, đoạn code trên request trường `'name, category, link, picture,`

`is_verified'` để trả về. Đây là tất cả các phần của trang public profile, nó không yêu cầu permission.

`fields` request khi tìm kiếm người dùng có đặc điểm là 1 phần của user public profile. Mặc dù các trường này tường minh, tuy nhiên dữ liệu sẽ không được trả về chừng nào chưa định rõ trong thuộc tính `fields` của request.

Search endpoint của Graph API chỉ trả về các thông tin đã công khai (tôi nghĩ thế). Bạn ắt hẳn muốn có thêm các thông tin khác mà access token cho phép.

Trong tình huống của tôi, sếp tôi có client quản lý 1 vài trang Facebook. Một trong những client này muốn tìm kiếm trang, chọn 1, sau đó lấy thông tin. Điều này yêu cầu 1 lời gọi API thứ 2, 1 request cho 1 đối tượng cụ thể (trong trường hợp này, là đối tượng `page`) từ Graph API.

3. Truy vấn 1 đối tượng Graph đơn

Phần này sẽ hướng dẫn bạn lấy thông tin từ 1 đối tượng đơn. Nó sẽ bắt đầu bằng việc truy vấn 1 người dùng đơn lẻ, kết thúc bằng việc truy vấn 1 bức ảnh.

Khi truy vấn route search (phần trên), 1 id sẽ được trả về, không phụ thuộc vào kiểu đối tượng hay các trường. Bạn cần id này để tạo 1 request đối tượng đơn (ví dụ `'page'`, `'user'`, `'event'`, `'photo'`, `'post'`,...).

Dưới đây là ví dụ truy vấn 1 người dùng đơn bằng id:

Permission: nếu bạn có id của 1 đối tượng Graph, bạn có thể request các dữ liệu chi tiết chừng nào mà access token của bạn vẫn có permission cho các trường bạn request.

Access token sử dụng trong request dưới có các permission sau:

- `'user_relationships'`
- `'user_about_me'`
- `'user_location'`
- `'user_website'`
- `'user_photos'`
- `'user_posts`
- `'email'`

Với các permission này, đoạn code dưới sẽ trả về **thông tin chi tiết của người dùng**.

Các đối tượng Graph khác có thể được request tương tự, chỉ khác ở các trường trong request.

Nếu bạn không có access token người dùng từ trình xác thực, bạn có thể sử dụng access token của ứng dụng. Với trường hợp dưới đây, bạn chỉ có thể request các dữ liệu profile có sẵn đã public.

```
app.get('/facebook-search/:id', (req, res) => {

  // you need permission for most of these fields
  const userFieldSet = 'id, name, about, email, accounts, link, is_

  const options = {
    method: 'GET',
    uri: `https://graph.facebook.com/v2.8/${req.params.id}`,
    qs: {
      access_token: user_access_token,
      fields: userFieldSet
    }
  };
  request(options)
    .then(fbRes => {
      res.json(fbRes);
    })
})
```

Dưới đây là response dạng JSON nhận được sau khi request dữ liệu người dùng của chính tôi (sử dụng 1 access token với tất cả các permission trên). Tôi sẽ thay thế các thông tin nhạy cảm với cụm `'youwish!'`.

```
{
  "id": "1458722400807259",
  "name": "Loren Stewart",
  "email": "youwish!",
  "location": {
    "id": "109434625742337",
```

```
    "name": "West Hollywood, California"
  },
  "accounts": {
    "data": [
      {
        "access_token": "youwish!",
        "category": "Internet Company",
        "name": "Codemore",
        "id": "1775471846051866",
        "perms": [
          "ADMINISTER",
          "EDIT_PROFILE",
          "CREATE_CONTENT",
          "MODERATE_CONTENT",
          "CREATE_ADS",
          "BASIC_ADMIN"
        ]
      }
    ],
    "paging": {
      "cursors": {
        "before": "MTc3NTQ3MTg0NjA1MTg2NgZDZD",
        "after": "MTc3NTQ3MTg0NjA1MTg2NgZDZD"
      }
    }
  },
  "link": "https://www.facebook.com/app_scoped_user_id/145872240080
  "is_verified": false,
  "significant_other": {
    "name": "Alex Galeczka",
    "id": "10210777928898847"
  },
  "relationship_status": "In a relationship",
  "website": "http://lorenstewart.me/",
  "picture": {
    "data": {
      "is_silhouette": false,
      "url": "https://scontent.xx.fbcdn.net/v/t1.0-1/p50x50/1446308
    }
  },
```

```
  "photos": {
    "data": [
      {
        "created_time": "2015-05-11T12:23:08+0000",
        "id": "994251753920995"
      },
      {
        "created_time": "2014-06-07T01:55:50+0000",
        "name": "Worker bee",
        "id": "810452288967610"
      },
// the list goes on
    ],
    "paging": {
      "cursors": {
        "before": "T1RrME1qVXhOelV6T1RJd09UazFPakUwTXpFek5Ea3dORGs2
        "after": "TXpReE1USXlOREkxT1RBNE5UZAzVPakV6TWpjNU1EQTFOakE2
      }
    }
  },
  "feed": {
    "data": [
      {
        "message": "Hey all. Don't just delete Uber's app. Delete yo
        "created_time": "2017-01-30T02:20:41+0000",
        "id": "1458722400807259_1445514005461432"
      },
      {
        "message": "I'm teaching a React workshop at the end of Febr
        "story": "Loren Stewart shared an event.",
        "created_time": "2017-01-29T21:00:17+0000",
        "id": "1458722400807259_1445248875487945"
      },
      {
        "story": "Loren Stewart updated his cover photo.",
        "created_time": "2017-01-04T02:56:35+0000",
        "id": "1458722400807259_1418730148139818"
      },
      {
        "story": "Loren Stewart at Artists Palette, Death Valley.",
```

```
        "created_time": "2016-12-31T04:19:52+0000",
        "id": "1458722400807259_1415002355179264"
      }
// the list goes on
    ],
    "paging": {
      "previous": "https://graph.facebook.com/v2.8/1458722400807259/
      "next": "https://graph.facebook.com/v2.8/1458722400807259/feed
    }
  }
}
```

Mặc dù có rất nhiều thông tin đã được trả về, tuy nhiên vẫn còn lại khá nhiều thông tin bị bỏ sót. Ví dụ, không có url ảnh nào trong thư viện ảnh. Để lấy url của 1 ảnh, bạn phải tạo 1 request khác (giống như là url Graph phía trên), sử dụng id của ảnh.

Dưới đây là code để request thêm dữ liệu từ 1 trong những ảnh thu được ở response trên.

```
const options = {
// let's assume id below is 994251753920995, the
// id for first photo listed in the prior response
  method: 'GET',
  uri: `https://graph.facebook.com/v2.8/${req.params.id}`,
  qs: {
      access_token: user_access_token,
      fields: 'link, comments, tags'
  }
};
request(options)
  .then(fbRes => {
    res.json(fbRes);
  })
```

Danh sách các trường mà bạn có thể request từ 1 đối tượng là ảnh được liệt kê ở đây. Đoạn mã trên request url của ảnh (`'link'`), comment (`'comments'`) và những người được tag vào trong ảnh (`'tags'`). Response trả về:

```
{
  "link": "https://www.facebook.com/photo.php?fbid=994251753920995&s
  "tags": {
      "data": [
          {
              "id": "1458722400807259",
              "name": "Loren Stewart",
              "created_time": "2015-05-11T12:57:29+0000",
              "x": 74.513885498047,
              "y": 32.429630279541
          },
          {
              "id": "10210777978898885",
              "name": "Alex Galeczka",
              "created_time": "2015-05-11T12:57:29+0000",
              "x": 85.694442749023,
              "y": 25.903705596924
          }
// the list goes on
      ],
      "paging": {
          "cursors": {
              "before": "MTQwMzgxOTIzNjMxNjczMQZDZD",
              "after": "MTAxNTQxNjMwNTc1MzE4NTYZD"
          }
      }
  },
  "likes": {
      "data": [

          {
              "id": "10155079062198541",
              "name": "Ian Stewart"
          },
          {
              "id": "10210777978898885",
              "name": "Alex Galeczka"
          }
// the list goes on
      ],
```

```
        "paging": {
            "cursors": {
                "before": "MTAxMDIyODA1NzY0MTM0OTgZD",
                "after": "MTAyMTExMDY1Nzc2NTQxNjIZD"
            }
        }
    },
    "comments": {
        "data": [
            {
                "created_time": "2015-05-11T15:29:01+0000",
                "from": {
                    "name": "Dee Abraham",
                    "id": "10210508246374852"
                },
                "message": ".......   https://youtu.be/sGiZoE5eqZc",
                "id": "994251753920995_994327993913371"
            },
            {
                "created_time": "2015-05-11T19:24:23+0000",
                "from": {
                    "name": "JD Woodrow",
                    "id": "657459424444412"
                },
                "message": "Wish I were there!",
                "id": "994251753920995_994404017239102"
            }
  //the list goes on
        ],
        "paging": {
            "cursors": {
                "before": "WTI5dGJXVnVkRjlqZAFhKemIzSTZAPVGswTXpJM09Ua
                "after": "WTI5dGJXVnVkRjlqZAFhKemIzSTZAPVGsxT1RReU9US
            }
        }
    },
    "id": "994251753920995"
}
```

Phần lớn dữ liệu thu được khá dễ hiểu. Các thuộc tính `x` và `y` được gói trong đối tượng `tags` là tọa độ của tag trong ảnh. Chú ý rằng đó cũng là pagination cho `likes`, `comments`, `tags`.

Url của ảnh đã được liệt kê trước. Nếu bạn chỉ cần url của ảnh, chỉ cần include `'link'` vào trường thuộc tính trong request (sẽ được đề cập trong phần 5 dưới đây).

4. Truy vấn trang mà người dùng quản lý

Đây không hẳn là request mà bạn sẽ thường sử dụng, tuy nhiên nếu ứng dụng của bạn cần quản lý các trang mà người dùng có quyền admin thì ví dụ này sẽ hữu dụng.

Trong ví dụ dưới đây, chú ý rằng **thuộc tính `fields` không được include** bởi vì dữ liệu mà bạn request đã bị ẩn đi trong url.

Truy vấn người dùng ở phần 3 cũng trả về những thông tin này bởi vì access token người dùng có permission cho các thông tin này, và `'accounts'` cũng được cụ thể trong thuộc tính `fields` của request.

Nếu bạn có permission `accounts`, và bạn chỉ muốn 1 danh sách các tài khoản người dùng quản lý (hơn là dữ liệu của tất cả người dùng), đây sẽ là truy vấn bạn cần thực hiện.

```
const options = {
// let's assume id below is 1458722400807259,
// which is my user id
  method: 'GET',
  uri: `https://graph.facebook.com/v2.8/${req.params.id}/accounts`,
  qs: {
      access_token: user_access_token
  }
};
request(options)
  .then(fbRes => {
    res.json(fbRes);
  })
```

Response trả về:


```
{
  "data": [
      {
          "access_token": "youwish!",
          "category": "Internet Company",
          "name": "Codemore",
          "id": "1775471846051866",
          "perms": [
              "ADMINISTER",
              "EDIT_PROFILE",
              "CREATE_CONTENT",
              "MODERATE_CONTENT",
              "CREATE_ADS",
              "BASIC_ADMIN"
          ]
      }
  ],
  "paging": {
      "cursors": {
          "before": "MTc3NTQ3MTg0NjA1MTg2NgZDZD",
          "after": "MTc3NTQ3MTg0NjA1MTg2NgZDZD"
      }
  }
}
```


Request trên trả về các trang Facebook tôi quản lý. Tương tự như các response khác, không có url của các trang. Muốn có thì phải tách ra thành nhiều request con.

Loại permission người dùng tương ứng với từng trang/tài khoản cũng được liệt kê trong response, có thể sẽ hữu ích với ứng dụng của bạn.

Quá nhiều access token.

Khi bạn truy vấn tài khoản người dùng (ví dụ các trang họ quản lý), mỗi tài khoản sẽ trả về 1 access token. Nếu bạn muốn thực thi bất cứ hành động nào ở trang đó, bạn sẽ cần các token tương ứng.

Ví dụ, nếu tôi muốn post nội dung đến trang Codemore Facebook, tôi cần phải sử dụng access token của trang này trong request POST (chứ không phải access token người dùng). Một khi sử dụng token người dùng để lấy token trang, token trang sẽ có các permission. Với 1 số trang, tôi có thể đăng được nội dung, 1 số khác thì tôi chỉ có quyền đọc.

5. Post nội dung

Dưới đây là ví dụ cho việc post text, ảnh hay video (kích thước dưới 1 Gb) đến trang hoặc feed của người dùng. Hãy bắt đầu với việc post text.

Post text

```
const id = 'page or user id goes here';
const access_token = 'for page if posting to a page, for user if pos

const postTextOptions = {
  method: 'POST',
  uri: `https://graph.facebook.com/v2.8/${id}/feed`,
  qs: {
    access_token: access_token,
    message: 'Hello world!'
  }
};
request(postTextOptions);
```

Post ảnh

```
const id = 'page or user id goes here';
const access_token = 'for page if posting to a page, for user if pos

const postImageOptions = {
  method: 'POST',
  uri: `https://graph.facebook.com/v2.8/${id}/photos`,
  qs: {
    access_token: access_token,
    caption: 'Caption goes here',
    url: 'Image url goes here'
```

```
  }
};
request(postImageOptions)
```

Post video

Nếu video có kích thước hơn 1Gb, sẽ cần tách nhỏ dung lượng để upload. Do điều kiện không cho phép nên tôi sẽ không post bất cứ video dung lượng lớn nào. Ví dụ này sẽ sử dụng 1 video dung lượng dưới 1Gb.

```
const id = 'page or user id goes here';
const access_token = 'for page if posting to a page, for user if pos

const postVideoOptions = {
  method: 'POST',
  uri: `https://graph.facebook.com/v2.8/${id}/videos`,
  qs: {
    access_token: access_token,
    description: 'Caption goes here',
    file_url: 'Video url goes here'
  }
};
request(postVideoOptions);
```

Upload media mà không thêm vào feed

Việc thêm ảnh hay video vào thư viện của người dùng/trang mà không hiện nó trên feed là hoàn toàn khả thi. Để làm việc đó, trong thuộc tính `qs` của request, thêm thuộc tính `no_story` và đặt nó là `true`. Nếu không, Facebook sẽ mặc định gán nó là `false`, việc upload sẽ làm hiện media trên feed.

Lấy url của post

Sau khi POST, bạn có thể sử dụng `id` hay `post_id` trả về để tạo 1 request khác. Upload 1 ảnh sẽ trả về 1 đối tượng với các thuộc tính `id` và `post_id`, trong khi upload video hay post text sẽ chỉ trả về thuộc tính `id`.

Dưới đây là 1 số ví dụ cho các response:

```
// example video/text post response
{
  id: '323262661405199'
}

// example image post response
{
  id: '323262661405199',
  post_id: '309953479402795_323262661405172'
}
```

Để lấy được 1 url, trường bạn cần request chính là `'permalink_url'`.

```
const fb_id = 'the returned post_id or id goes here';
const type = '`video` or `post`, see comment below'

const getUrlOptions = {
  method: 'GET',
  uri: `https://graph.facebook.com/v2.8/${fb_id}`,
    qs: {
      access_token: user_access_token,
      // type should be 'post' for image or text,
      // and should be 'video' for a video url
      type: type,
      fields: 'permalink_url'
    }
};
return request(getUrlOptions)
  .then(fbUrlRes => {
    const permalink = JSON.parse(fbUrlRes).permalink_url;
      if(video) {
        permalink = `https://www.facebook.com${permalink}`;
      }
      return {postUrl: permalink};
  })
```

POST ảnh và text sẽ trả về url đầy đủ. Tuy nhiên response video trả về chỉ là phần cuối của url, do đó bạn cần tự dựng lại url (xem dòng thứ 19). Để dựng 1

video url, bạn cần thêm vào phía trước giá trị trả về chuỗi `'https://www.facebook.com'` .

Bài viết được dịch từ: http://lorenstewart.me/2017/03/12/using-node-js-to-interact-with-facebooks-graph-api/?utm_source=nodeweekly&utm_medium=email

PHP vs Node.js - Cuộc chiến giữa hai công nghệ lập trình web
13/10/2015 Hồ Sỹ Hùng

BLOG HOME

7 Quy tắc cấu trúc ứng dụng Node.js
24/07/2017 Đinh Thiên Phúc

*Bởi **Đinh Thiên Phúc***

Web development has changed the way I feel about life. Maybe the next time will be yours.

1 Comment Sort by **Newest**

Add a comment...

Liyu Xia

a ơi bây giờ mình vẫn có thể dùng api graph để đăng lên tường người khác được ạ?

Like · Reply · 40w

Facebook Comments plugin

Địa chỉ

Số 14, ngõ 4, Nguyễn Đình Chiểu, Hai Bà Trưng, Hà Nội
Giờ mở cửa: **từ 9:30 - 18:00**