**Training**

1. **Các tool cần chuẩn bị:**
   1. NetBean: https://netbeans.org/
   2. Tortoisegit(Có link trong phần 4)
   3. Beyond Compare(Có sẵn trong thư mục)
   4. Run code:đọc phần 4 để biết run code như thế nào
2. **1 số thuộc tính sẽ dùng trong AngularJS:**

* Chạy thư mục basic example để học các thuộc tính cơ bản:
  + ng-model
  + ng-options
  + ng-blur
  + ng-change
  + ng-click
  + ng-repeat
  + ng-repeat-start
  + ng-repeat-end
  + ng-required
  + ng-style
  + ng-include…

1. **Cấu trúc của 1 câu API:**

|  |
| --- |
| host/api/table\_name?cm=method&dt={“table\_name”:{parameters\_of\_table},”company”:{“company\_id”:40743},”languages”:{“language\_id”:”vi”},”page”:{“page\_index”:1,”page\_size”:10}} |

* Host: tên miền như pms-dev.wahsis.net, pos-dev.wahsis.net, dev.wahsis.net…
* Table\_name: tên bảng trong cơ sở dữ liệu.
* Method: bao gồm các phương thức: **add, update, delete, search, add\_list**… Sẽ có 1 số method khác do anh Khiêm định nghĩa.
* Company: sẽ có thuộc tính là company\_id. Company\_id sẽ lấy từ mã code công ty khi đăng nhập.
* Languages: ngôn ngữ hiện tại chỉ bao gồm vi và en.
* Page: dùng để phân trang. Page\_size : số phần tử trong 1 trang, Page\_index : trang số mấy.

1. **Các bước thêm 1 chức năng trong template:**

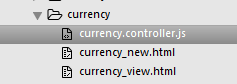
**Ví dụ:** Code chức năng currency

|  |
| --- |
| [http://pms-dev.wahsis.net/api/currency?cm=list&dt={"company":{"company\_id":40743}](http://pms-dev.wahsis.net/api/currency?cm=list&dt=%7b%22company%22:%7b%22company_id%22:40743%7d),  "languages":{"language\_id":"vi"}} |

* **Bước 1:**
  + Tạo thư mục chứa 2 file html và js. Tên thư mục, tên file html và js sẽ đặt theo tên bảng và chức năng của file đó.

Ví dụ: Tên bảng là currency.

* + - Thư mục : currency.
    - HTML
      * currency\_view.html: chứa table để hiển thị list danh sách và các điều kiện search.
      * currency\_new.html: Chứa các thẻ html cần thiết cho người dùng nhập liệu.
      * currency.controller.js: Nơi xử lí code bằng js.

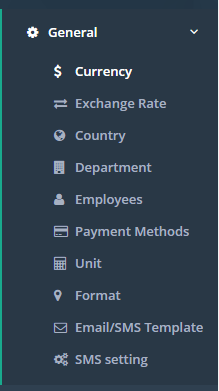


* **Bước 2:**
  + Khai báo file js vừa tạo trong trang index.html.

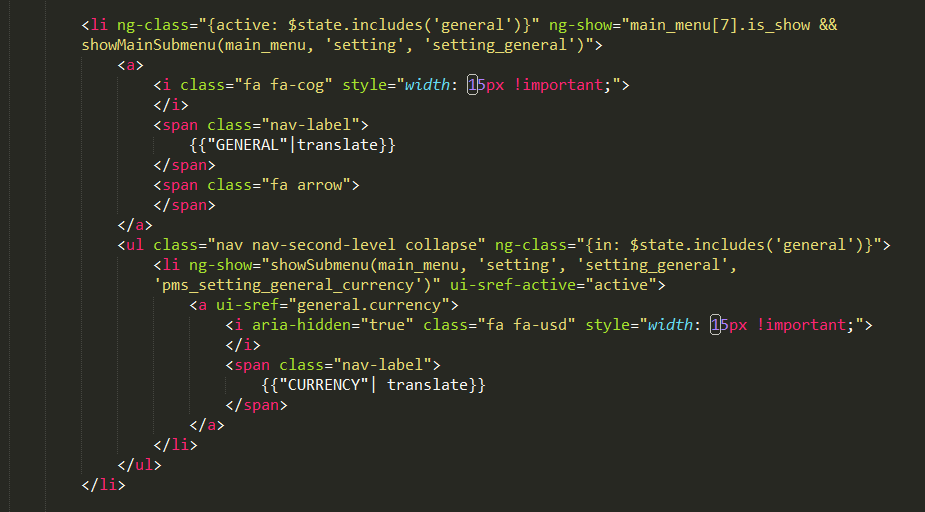


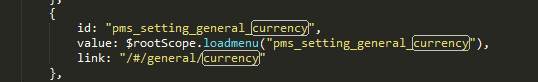
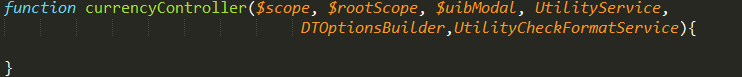
* **Bước 3:**
  + Tạo menu cho chức năng mới:
    - view->common-> navigation.
    - js-> controller.
    - js->config.

1. **Xác định nơi cần tạo menu mới.**

C:\Users\MyPC\Desktop\Zalo_ScreenShot_20_3_2017_1723239.png

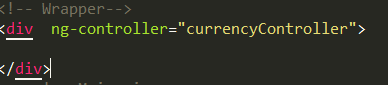
Có URL:

1. **Khai báo menu mới trong view->common-> navigation:**
   1. Do menu currency là menu con của general nên khi ta khai báo ui-sref thì ta phải thêm general vào trước currency=> general.currrency.

* 1. **Khai báo trong js-> controller:**
     + -  id : khác với những id menu đã tạo và phải trùng với trong **navigation**
       - value: tham số này được tạo trong csdl dùng để phân quyền cho menu
       - link : do currency là con của general nên nó sẽ có dạng /#/general/currency
  2. **Khai báo url cho menu trong js->config.**
     + - state phải có name giống với ui-sref trong navigation.
       - template url: là đường dẫn dẫn đến file html của currency.
       - Phần còn lại là các thư viện sẽ sử dụng trong chức năng đó.
* **Bước 4: Định nghĩa controller**
  + **Trong file currency.controller.js, ta sẽ định nghĩa 1 controller mới cho chức năng này**.
    - **$scope:** là đối tượng JavaScript đặc biệt có vai trò liên kết controller và view. Nó chỉ có sự ảnh hưởng trong phạm vi của controller đó.
    - **$rootScope:** là bậc cao nhất nên sẽ bao quát hết các $scope bên trong nó, nó không giống với $scope là chỉ ảnh hưởng trong phạm vi của controller.
    - **$http:** hiểu 1 cách đơn giản thì nó dùng để call api như ajax.
    - **API\_URL:** là 1 hằng số được định nghĩa sẵn. Ở đây nó chứa giá trị**:** [**http://pms-dev.wahsis.net/api/**](http://pms-dev.wahsis.net/api/)
    - **$uibModal:** được khai báo để thao tác với popup.
    - **UtilityService:** đây 1 service angularjs được tạo ra bởi người dùng, ở đây chúng tôi tạo ra nó để chứa các hàm dùng chung.(có thể vào thư mục utilities/utilities.service.js trong project pms để tham khảo thêm các tham số cần cho các hàm đó). Dưới đây là 1 số hàm được định nghĩa sẵn hay dùng:
      * **Decentralization:** phân quyền cho chức năng đó.
      * **getListObject:** được sử dụng khi câu api chỉ có 2 parameters là company và languages, dùng để load danh sách
      * **getListObjectWithParam:** được sử dụng khi câu api chỉ có từ 3 parameters trở lên.
      * **getListObjectDev:** được sử dụng khi câu api chỉ có 2 parameters là company và languages **và gọi qua server dev**, dùng để load danh sách.
      * **getListObjectWithParamDev:** được sử dụng khi câu api chỉ có từ 3 parameters trở lên **và gọi qua server dev.**
      * **getListCoreConfigData:** load những dữ liệu mặc định đã đươc config trong db lên.
      * **addObject:** dùng cho câu api add new.
      * **updateObject:** dùng cho câu api update.
      * **deleteObject:** dùng cho câu api delete.
      * **searchObject:** dùng cho câu api search. Ngoài ra cũng có thể dùng câu **getListObjectWithParam.**
    - **DTOptionsBuilder:** đây là 1 service dùng cho datatable, khi ta cần dùng datatable trong html thì bắt buộc phải khai báo nó trong controller và khởi tạo nó với cú pháp.

|  |
| --- |
| **$scope.dtOptions = DTOptionsBuilder.newOptions();** |

* + - **UtilityCheckFormatService:** đây cũng là 1 service được định nghĩa sẵn dùng để format 1 số thứ cần thiết như số, ngày tháng năm…(có thể vào thư mục utilities/utilities.service.js trong project pms để tham khảo thêm các hàm và các tham số cần cho các hàm đó).
      * **defaultFormat:** hàm chuyển đổi kiểu ngày tháng năm trên server trả về thành định dạng được setting bởi user**.**
      * **change\_date\_to\_save:** hàm chuyển đổi kiểu ngày tháng năm thành kiểu định dạng của server, thường được dùng trong các câu add new và update.
      * **check\_date\_from\_to:** hàm kiểm tra 2 ngày from và to.
      * **change\_number:** hàm chuyển đổi kiểu số int trên server trả về thành kiểu định dạng tiền tệ.
      * **change\_number:** hàm chuyển đổi kiểu số int trên server trả về thành kiểu định dạng tiền tệ.
  + **Trong file currency\_view.html, ta khai báo ng-controller đúng với tên controller được định nghĩa trong js để bắt đầu cho 1 chức năng mới. Những thẻ html nào nằm trong div dưới đây mới binding được với controller đó.**.



* + **Khai báo tiếp tên controller đó trong file js=>controller.js**
* **Bước 5: Tổng hợp cách call api, translate, popup, phân trang.**
  + **Cách gọi câu api get list chỉ có 2 parameters là company và languages:**
    - Trong project pms thường sẽ dùng:

|  |
| --- |
| UtilityService.getListObject(‘table\_name’,language).then(function(  respone){  if(respone.data.err === 0){  // kết quả trả về thành công.  $scope.result=respone.data.dt.x  }else {  // kết quả trả về thất bại và ở đây sẽ show popup thông báo lỗi  swal(‘Warning’,’Error’,’warning’);  return;  }  }) |

* + - Trong project pos:

|  |
| --- |
| $http.get(API\_URL+”currency?cm=list&dt={company:{company\_id:”+com\_id+”},languages:{language\_id:”+language+”}”) |

* + - com\_id và language: được lưu sẵn trong cookies khi đăng nhập.
  + **Cách gọi câu api get list chỉ có 3 tham số trở lên:**

Ví dụ ta có câu api sau:

|  |
| --- |
| **http://pms-dev.wahsis.net/api/currency?cm=search&dt={“ currency “:{“currency\_code”:”usd”},"company":{"company\_id":40743},**  **"languages":{"language\_id":"vi"}}** |

* + - Trong project pms thường sẽ dùng:

|  |
| --- |
| * UtilityService.getListObjectWithParam(‘table\_name’,’method’, dataJson).then(function(respone){} |

*Method ở trên có thể là add, update, get\_list, list, search... do anh Khiêm định nghĩa*

* + - * Bước 1: Ta phải định nghĩa 1 object chứa currency\_id:1
      * Bước 2: đưa object về dạng json
      * Bước 3: gọi câu api với hàm trên

|  |
| --- |
| * //**Ở đây mình đang hardcode nhưng trong project thì phải làm động.** * // **Khai báo object** * $scope.currencyObject={currency\_code:”usd”}; * $scope.companyObject={company\_id:40743}; * $scope.languagesObject={language\_id:’vi’} * **//Đưa các object về dạng json** * **//Trong câu api trên sẽ có 3 params là: currency, company và languages** * **$scope.dataJson**=**JSON.stringify ({currency:** $scope. currencyObject**, company:** $scope.companyObject**, languages:** $scope.languagesObject **});** * **//call api với hàm** * UtilityService.getListObjectWithParam(“**currency**”,” **search**”, **$scope.dataJson**).then(function(respone){ * if(respone.data.err === 0){ * // kết quả trả về thành công. * $scope.result=respone.data.dt.x * }else { * // kết quả trả về thất bại và ở đây sẽ show popup thông báo lỗi * swal(‘Warning’,’Error’,’warning’); * return; * } * }) |

* + - Trong project pos:

|  |
| --- |
| $http.get(API\_URL+”currency?cm=search&dt={currency:{  currency\_code:’usd’},company:{company\_id:”+com\_id+”},languages:{language\_id:”+language””}”) |

* + **Cách gọi câu api add hoặc update:**

Ví dụ ta có câu api sau:

|  |
| --- |
| **http://pms-dev.wahsis.net/api/currency?cm=add&dt={“ currency “:{“currency\_code”:”usd”,”currency\_name”:”** **United States dollar”,”** **symbol”:”$”},"company":{"company\_id":40743}}** |

Trước tiên ta phải tạo 1 form cho người dùng input với 3 thẻ input lần lượt có ng-model là: currency\_code, currency\_name, symbol.

* + - Trong project pms thường sẽ dùng:

|  |
| --- |
| * **Add new:** * UtilityService.addObject (‘table\_name’ , dataJson).then(function( * respone){} |
| **Update:**  UtilityService.updateObject (‘table\_name’ , dataJson).then(function(  respone){} |

* + - * Bước 1: đưa object về dạng json
      * Bước 2: gọi câu api với hàm trên.

|  |
| --- |
| * // **Khai báo object chứa 3 params trên và dùng scope để get dữ liệu từ form cho object:** * $scope.object={ * currency\_code:$scope.currency\_code, * currency\_name:$scope.currency\_name, * symbol:$scope.symbol * } * $scope.companyObject={company\_id:40743}; * $scope.languagesObject={language\_id:’vi’} * **//Đưa object về dạng json** * **//Trong câu api trên sẽ có 3 params là: currency, company và languages** * **$scope.dataJson**=**JSON.stringify ({currency:** $scope. object**, company:** $scope.companyObject**})** * UtilityService. addObject (“**currency**”, **$scope.dataJson**).then(function(respone){ * if(respone.data.err === 0){ * // kết quả trả về thành công. * $scope.result=respone.data.dt.x * }else { * // kết quả trả về thất bại và ở đây sẽ show popup thông báo lỗi * swal(‘Warning’,’Error’,’warning’); * return; * } * }) |
| * Update tương tự add new nhưng ta phải truyền thêm languages trong câu JSON. **$scope.dataJson**=**JSON.stringify ({currency:** $scope. object**, company:** $scope.companyObject**, languages:** $scope.languagesObject **})** |

* + **Khai báo và sử dụng popup:**

*Thư mục* ***popup example*** *có code minh hoạ nhưng không thể run được vì không đủ thư mục*

* + - **Open popup:**

|  |
| --- |
| * function openPopup(grid, ser) { * var modalInstance = $uibModal.open({ * templateUrl: 'setting/currency/currency\_new.html', * controller: crudCurrency, * windowClass: "animated fadeInDown", * draggable: true, * size: 'md', * locked: true, * resolve: { * grid: function() { * return grid; * }, * item: function() { * return ser; * } * } * }).result.then(function() { * // hàm này chỉ thực hiện sau khi close popup * // với $uibModalInstance.close(); * $scope.getData(); * }); * } |

* + - **Close popup:**

Có 2 cách:

* + - * **Cách 1: Sẽ thực hiện result trong lúc openpopup sau khi đã close popup:**

|  |
| --- |
| $uibModalInstance.close(); |

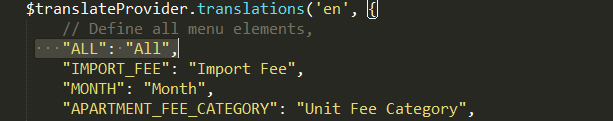
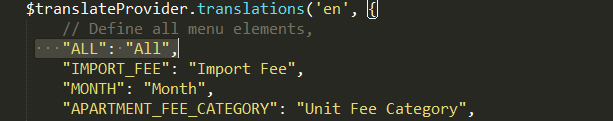
* + - * **Cách 2: Sau khi close popup thì không thực hiện result:**

|  |
| --- |
| $uibModalInstance.dismiss(); |

**Lưu ý:**

* Trong hàm open popup chúng ta sẽ thấy có 1 nơi khai báo **controller** trong đoạn code trên. Lúc này chúng ta phải tạo ra 1 function controller mới và gán tên đó vào trong popup. Tất cả những chức năng thực hiện trên popup đó sẽ được xử lý trên controller mới này.(xem thêm ví dụ trong folder để có thể hiểu rõ hơn).
  + **Translate:**

*Ta sẽ vào js->translate.js để translate cho 1 keyword nào đó. Trong project hiện tại của chúng ta chỉ có 2 ngôn ngữ là English và Vietnamese nên ta sẽ khai báo keyword ở 2 chỗ trong file này*

* + - Ví du: Mình cần translate từ “Tất cả”. Mình sẽ đặt cho nó keyword là **ALL**, và mình sẽ khai báo nó như sau:
    - *Và trong HTML khi cần sử dụng, ta sẽ dùng với cú pháp sau :*

|  |
| --- |
| **{{ “keyword”|translate }}** |

**Ví dụ :**

* + **Phân trang:**

*Thư mục* ***paginate example*** *có code minh hoạ nhưng không thể run được vì không đủ thư mục.*

* + - Trong HTML, ta sẽ thêm đoạn code sau trước **<table>**và sau **</table>**

|  |
| --- |
| * <div class=" text-right " id="div\_paging" style="margin-bottom: 0px;padding-top:0 ;background-color:transparent !important;padding-left: 0px" > * <button ng-disabled="currentPage1 == 1" ng-click="load\_data(4)"class="btn btn-default btn-xs black"><i class="ion-arrow-left-a"></i> * {{"FIRST"|translate}} * </button> * <button ng-disabled="currentPage1 == 1" ng-click="load\_data(1)"class="btn btn-default btn-xs black"><i class="ion-chevron-left"></i> * {{"PREVIOUS"|translate}} * </button> * {{"ITEM\_PER\_PAGE"|translate}}: <select id="page" ng-model="item\_per\_page" ng-change="item\_per\_page2()" ng-options="x.value as x.text for x in item\_per\_page1" ></select> * {{"PAGE"|translate}} <select id="page" class="page" ng-model="currentPage1" ng-change="load\_data(2)" ng-options="x.val as x.test for x in list\_page" ></select> * {{"TOTAL\_PAGE"|translate}}: {{total\_page}} * <button ng-disabled="currentPage1 == total\_page || total\_page == 0" ng-click="load\_data(3)"class="btn btn-default btn-xs black"><i class="ion-chevron-right"></i> * {{"NEXT"|translate}} * </button> * <button ng-disabled="currentPage1 == total\_page || total\_page == 0" ng-click="load\_data(5)"class="btn btn-default btn-xs black"><i class="ion-arrow-right-a"></i> * {{"LAST"|translate}} * </button> | {{"TOTAL\_RESULT"|translate}}: {{total\_row}} * </div> |

* + - Trong controller của chúng ta:

|  |
| --- |
| * var json\_value = getCookieJson("pms-dev-format"); * $scope.item\_per\_page = parseInt(JSON.parse(json\_value).item\_per\_page); * $scope.item\_per\_page\_list = JSON.parse(json\_value).item\_per\_page\_list; * $scope.page\_length = page\_length; * $scope.currentPage1 = 1; * $scope.item\_per\_page1 = []; * for (var j = 0; j < $scope.item\_per.length; j++) { * $scope.item\_per\_page1.push({ * value: $scope.item\_per[j], * text: $scope.item\_per[j] * }); * } * $scope.page = {page\_index: 1,page\_size: $scope.item\_per\_page}; |

* + - ***Ví dụ : Page và API***

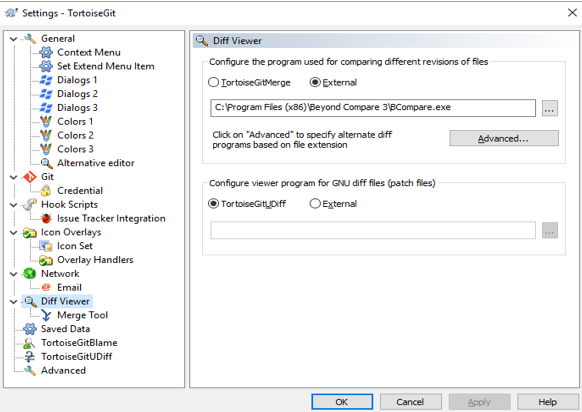
|  |
| --- |
| * **//hàm get Data từ API có phân trang** * **$scope.getData=function(){** * $scope.json = JSON.stringify({ * rooms: $scope.str\_search\_rooms, * company: $scope.company, * page: $scope.page, * languages: $scope.languageTemp, * apartment\_sales\_price: $scope.str\_search\_contract * }); UtilityService.getListObjectWithParam('apartment\_sales\_price', 'report\_to\_date', $scope.json).then(function(response) { * if (response.data.err === 0) { * $scope.periodListList = response.data.dt.apartment\_sales\_payment\_period\_list; * **//tổng số phần tử và trang trên server trả về** * $scope.total\_page = response.data.dt.page.total\_page; * $scope.total\_row = response.data.dt.page.total\_row; * $scope.list\_page = []; * for (var i = 1; i <= $scope.total\_page; i++) { * var j = { * val: "", * test: "" * }; * j.val = i; * j.test = i; * $scope.list\_page.push(j); * } * } * }) * } * **// hàm này thực hiện khi click chọn số phần tử của 1 trang trong html** * **$scope.item\_per\_page2 = function()** { * $scope.page = { * page\_index: 1, * page\_size: $scope.item\_per\_page * }; * $scope.getData(); * } * **// hàm này thực hiện khi nhấn các button next, first, last, previous** * **$scope.load\_data = function(loai)** { * console.log($scope.str\_search) * if ($scope.currentPage1 === undefined)   $scope.currentPage1 = 1; * if (loai === 1) { * $scope.currentPage1 = $scope.currentPage1 - 1; * } * if (loai === 3) { * $scope.currentPage1 = $scope.currentPage1 + 1; * } * if (loai === 4) { * $scope.currentPage1 = 1; * } * if (loai === 5) { * $scope.currentPage1 = $scope.total\_page; * } * $scope.page = { * page\_index: $scope.currentPage1, * page\_size: parseInt($scope.item\_per\_page) * }; * $scope.getData(); * } * **//Nếu chức năng có hàm search** * $scope.search = function() { * $scope.currentPage1 = 1; * $scope.page = { * page\_index: 1, * page\_size: parseInt($scope.item\_per\_page) * }; * $scope.getData(); * } |

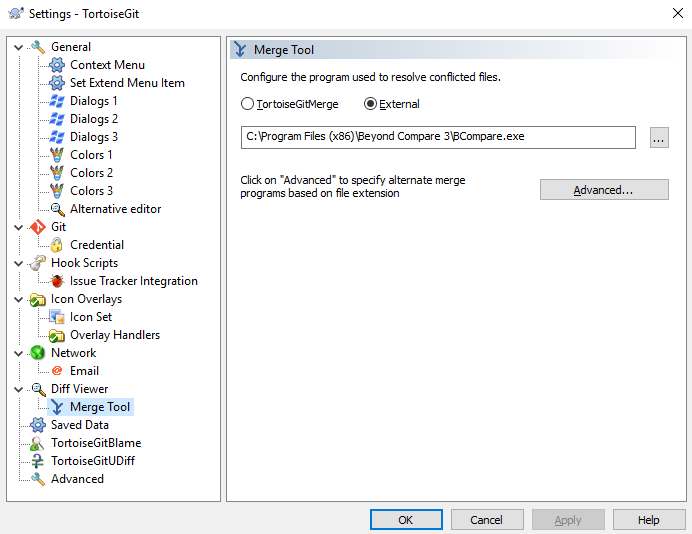
1. **Làm việc với git:**

* **Bước 1**
  + Tải **Tortoisegit** và cài đặt nó từ trang này: [**https://tortoisegit.org/**](https://tortoisegit.org/)
  + Cài đặt **Beyond compare** trong thư mục.
* **Bước 2**

*Setup* ***Beyond compare với Tortoisegit***

* + Chuột phải ra màn hình desktop và chọn **Tortoisegit => Settings.**
  + Tại mục Diff Viewer. Ta sẽ setup như sau:

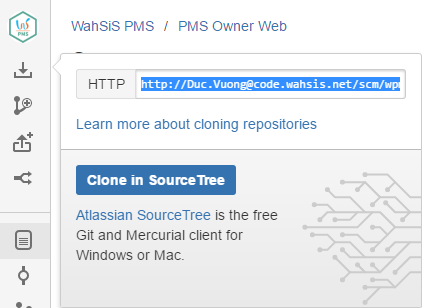




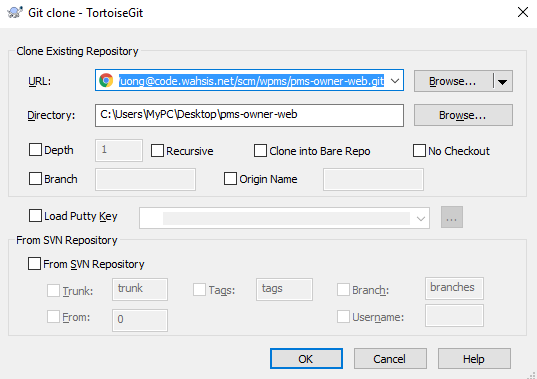
* + Click **OK.**
* **Bước 3**

***Clone source code từ web về.***

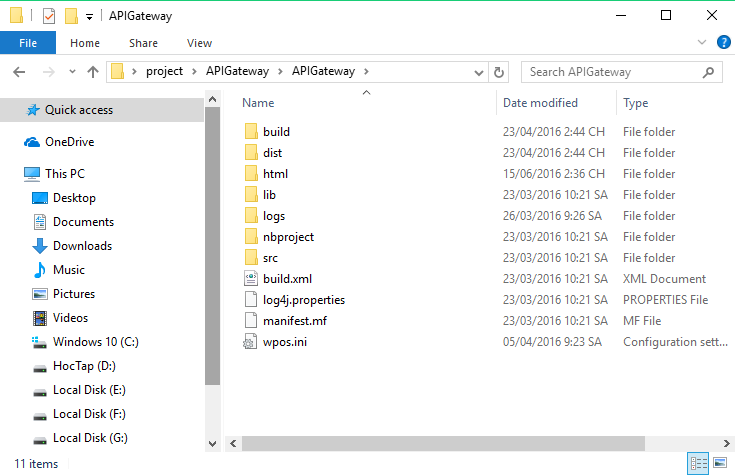
* + Vào trang : <http://code.wahsis.net/projects>
  + Chọn project và click clone để lấy đường dẫn.



* + Chuột phải ra màn hình desktop và chọn **Git Clone**, ta sẽ paste đường dẫn đó vào **URL** và chọn nơi lưu trữ => Ok.



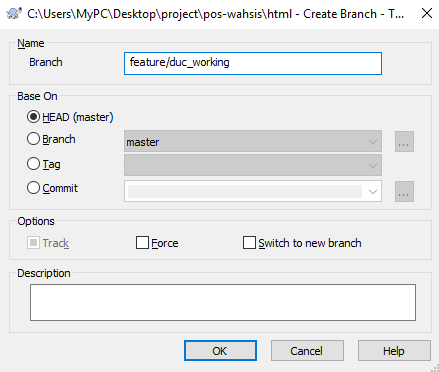
* + Sau khi clone xong chúng ta sẽ giải nén thư mục **APIGateway.zip**, xoá thư mục html có sẵn trong đó và paste thư mục clone từ git vào đây, đổi tên thư mục đó thành **html.**



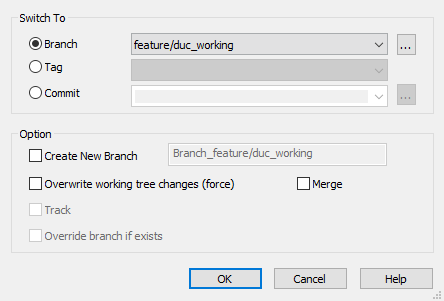
* + Tiếp theo ta sẽ mở netbean => open folder =>chọn tới đường dẫn đến thư mục Apigateway và ok.
  + Run code : chọn API GATEWAY và click button Run hoặc F6
    - Nếu netbean báo port đã sử dụng rồi, bạn vào thư mục html-> wpos.ini và đổi **port\_listen** lại.
  + Trên trình duyệt sẽ gõ: **localhost** hoặc **localhost:port** để chạy
* **Bước 4**

***Tạo branch, merge và commit code.***

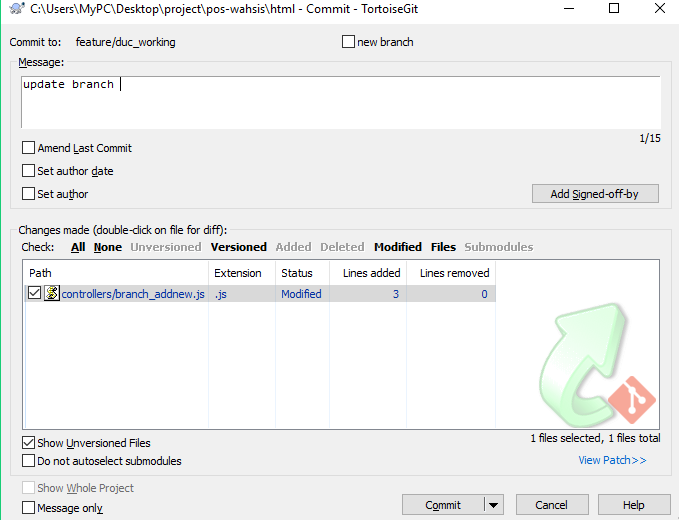
* + Chuột phải vào thư mục **html**, chọn **TortoiseGit** => **Create Branch** và nhập tên branch cho riêng bạn => **OK**.

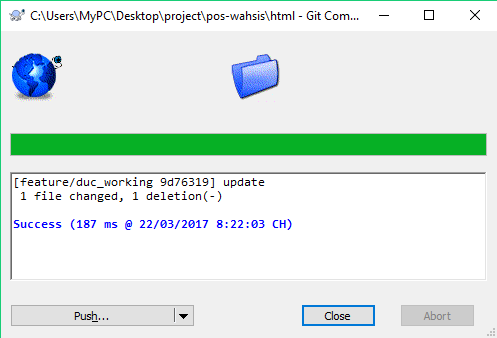


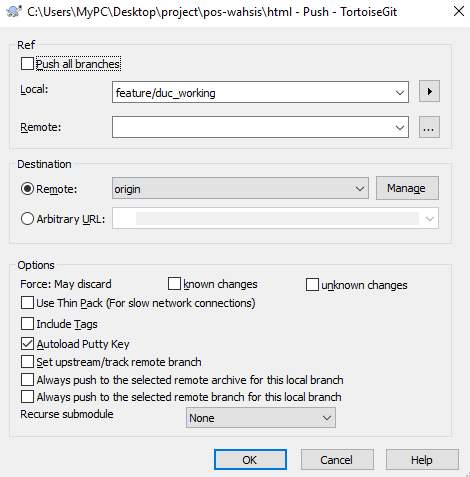
* + Chúng ta lại tiếp tục **HTML** => **TortoiseGit** => **Swich/Check out => Chọn branch của bạn** => OK và bắt đầu code.

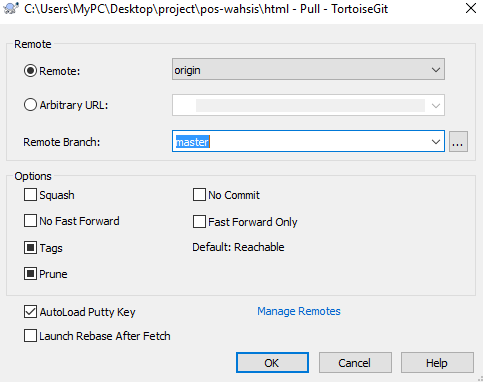


* + Sau khi chúng ta đã code hay chỉnh sửa gì đó xong và cần commit code. Trước tiên nên zip lại file HTML, nếu merge code lỗi hay gì đó ta sẽ giải nén ra lại để không mất code. Nếu đã thành thạo rồi thì bỏ qua bước này.
  + Tiếp đó, ta chuột phải vào thư mục **HTML** => **Git commit-abcxyz** => comment vào textarea message những thay đổi của bạn => **commit => push => OK**



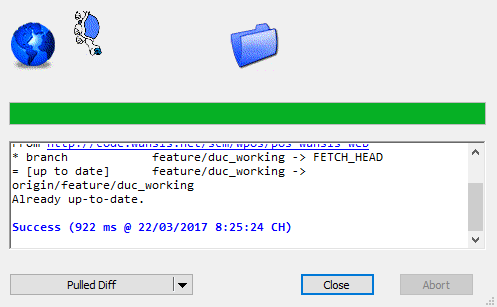




* + Ta lại chuột phải vào thư mục **HTML** => **TortoiseGit** => **PULL** =>ở **Remote Branch** ta sẽ chọn **master**=> **OK**. (lúc này code từ git sẽ đè lên code ở branch của chúng ta

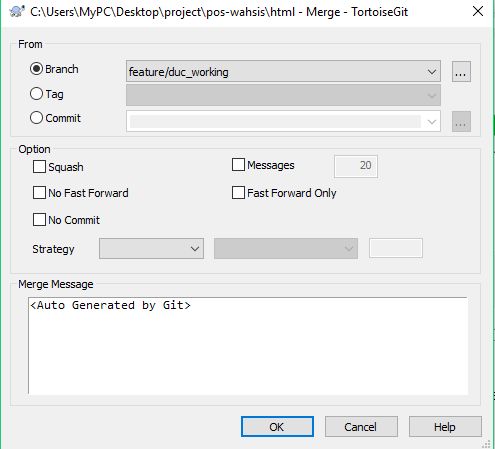
**Ở đây sẽ có 2 trường hợp xảy ra:**

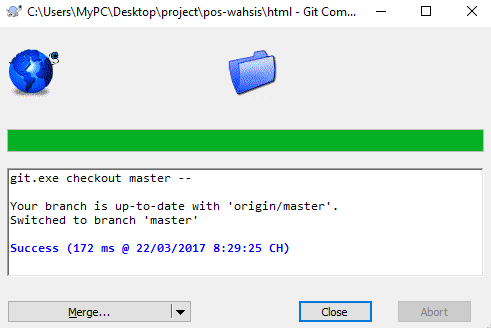
* + - **TRƯỜNG HỢP 1: KHÔNG BỊ CONFLICT CODE**

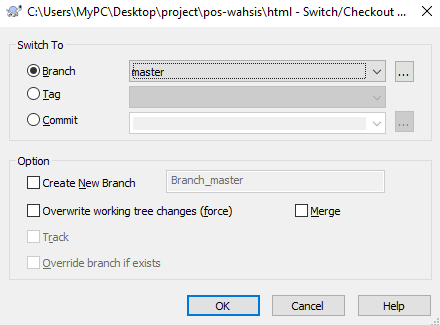


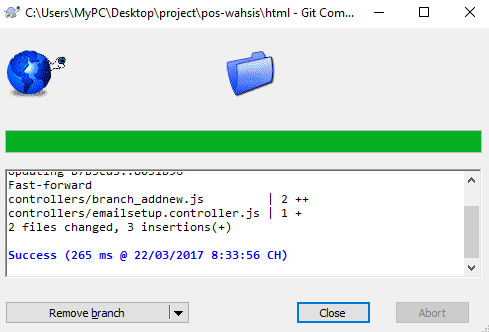
Lúc này thư mục html của bạn vẫn có **màu xanh**, tức là không bị conflict code với ai.

* + - Chọn **HTML** => **TortoiseGit** => **Swich/Check out =>** tại branch chúng ta sẽ chọn **Master =>Ok=> Merge =>Ok=>Close.**

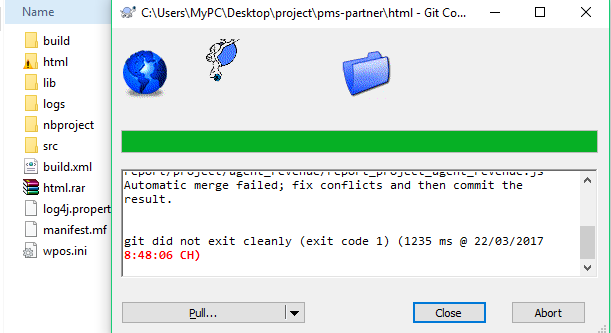
****

****

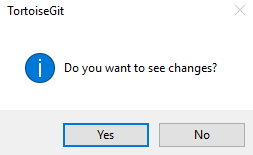
****

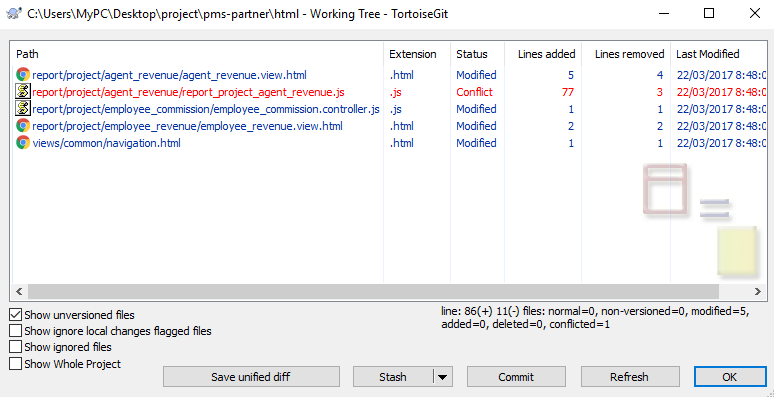
****

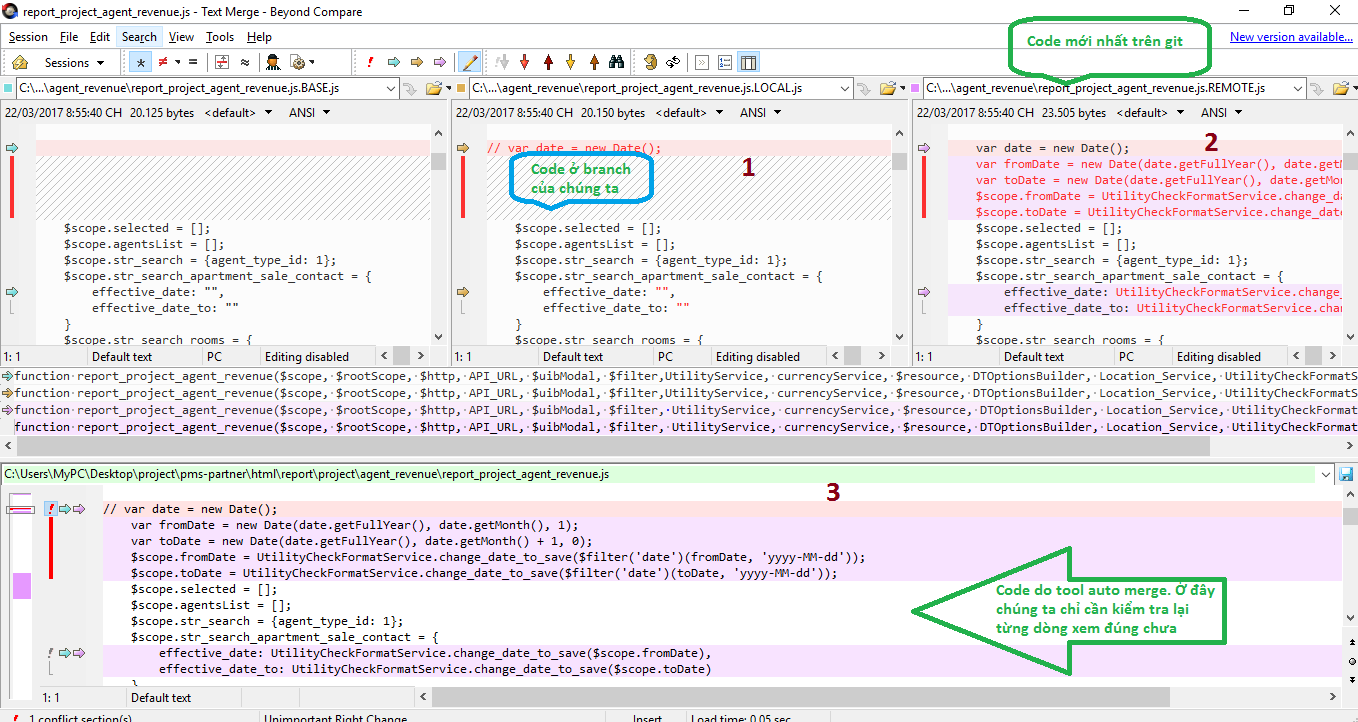
* + - * Cuối cùng **HTML** => **TortoiseGit** => **PUSH** => **OK** => **Close**=> **TortoiseGit =>**  **Swich/Check out => Chọn lại branch của mình(không cần tạo tiếp)=> Code.**
    - **TRƯỜNG HỢP 2: CONFLICT CODE(html sẽ có dấu chấm than màu vàng)**



Ở đây bạn thấy git sẽ thông báo **GIT ĐI NOT EXIT CLEANLY** và thư mục **html** bị **chấm than.**

* + - Chúng ta click **close** => **yes**

 **GIT sẽ thông báo cho chúng ta biết file nào bị conflict**

* + - Double click vào file conflict.

**Ở đây chúng ta cần so sánh code 3 nơi được mình đánh dấu. Xem code ở branch(1) với code mới trên git(2) và code ở phần dưới do auto merge(3), chúng ta thay đổi code nào ở branch(1) khác với trên git(2) thì sẽ được tô đỏ code như hình và các dấu mũi tên màu vàng(1 và 2, chúng ta cần chú ý cả mũi tên màu vàng khi check nhé). Check code ở 3 xem auto merge đã đúng chưa, nếu chưa thì ta sẽ copy code ở (1) hoặc (2) và paste xuống (3). Sau đó bấm save(nút save ở góc phải giữa hình) và close đi.**

* + - Quay lại màn hình thông báo conflict **chuột phải vào file conflict** và chọn **Resolve.** Nếu lỡ tắt rồi thì chuột phải **HTML=> TortoiseGit => Resolve => chuột phải file đã fix conflict => Resolve**. Rồi quay lại fix những file conflict khác và làm tương tự vậy cho từng file.
    - Sau khi đã fix hết conflict, các bạn nên check lại trên local mình web chạy đúng chưa, có lỗi không, nếu lỗi thì sửa trực tiếp trên netbean hoặc 1 IDE nào đó mà bạn dùng.
    - Cuối cùng, **HTML**=> **TortoiseGit** => **Commit** => **Push** => **Close** => **HTML**=> **TortoiseGit** =**>** **Swich/Check out** => tại **branch** chúng ta sẽ chọn **Master** =>**Ok**=> **Merge** => **Ok** => **HTML**=> **TortoiseGit** => **Push**
    - **HTML**=> **TortoiseGit** =**>** **Swich/Check out=> chọn branch mình=> code tiếp.**