AngularJS Variable Flow

Index

<ul class="nav navbar-nav">

<li>

<a ui-sref="Categories">Categories</a>

</ul><!--Categories Item-->

Controller

angular.module('app')

.controller('TransactionsController', ['$scope', '$state', '$stateParams', 'transactions',

function ($scope, $state, $stateParams, transactions) {

$scope.accountId = $stateParams.accountId;

$scope.transactions = transactions;

}]);

View for that Controller

<button class="btn btn-primary" ui-sref="CreateTransaction({accountId: accountId})">

Create A New Transaction

</button>

<th>Category</th>

</tr>

<tr ng-repeat="transaction in transactions">

<td>{{transaction.Amount| currency:'$'}}</td>

<td>{{transaction.AbsAmount| currency:'$'}}</td>

<td>{{transaction.ReconciledAmount| currency:'

<a class="btn btn-primary" ui-sref="EditTransaction({transactionId: transaction.Id})">

<span class="glyphicon glyphicon-pencil" aria-hidden="true">Edit or Delete Transaction</span>

</a>

</td>

Create Controller

angular.module('app')

.controller('CreateTransactionController', ['$scope', '$state', '$stateParams', 'accountsService', 'categories',

function ($scope, $state, $stateParams, accountsService, categories) {

$scope.categories = categories;

$scope.accountId = $stateParams.accountId;

$scope.transaction = {

AccountId: $scope.accountId,

Updated: null,

UpdatedByUserId: null,

CategoryId: null

};

//CREATE TRANSACTION

$scope.createTransaction = function createTransaction() {

accountsService.createTransaction($scope.transaction).then(function () {

$state.go('Transactions', {accountId: $scope.accountId});

});

}

}]);

Create View

</div>

<div class="form-group">

<label class="control-label col-md-3" for="CategoryId">Select a Category:</label>

<select type="text" class="form-control col-md-9" ng-model="transaction.CategoryId" ng-options="category.Id as category.Name for category in categories"></select>

</div>

</form>

<br />

<br />

<div>

<input type="button" class="btn btn-primary" value="Create Transaction" ng-click="createTransaction()">

<input type="button" class="btn" value="Cancel" ui-sref="getTransaction({id: account.Id})">

</div>

</div>

Factor(ies) for Accounts

'use strict';

angular.module( 'app' )

.factory('accountsService', ['$http', function ($http) {

var factory = {};

factory.getAccounts = function (houseHold) {

var options = { params: { houseHold: houseHold } };

return $http.get('/api/accounts/GetAccounts', options).then(function (response) {

return response.data;

});

};

factory.getTransactions = function (accountId) {

var options = { params: { accountId: accountId } };

return $http.get('/api/transactions/GetTransactions', options).then(function (response) {

return response.data;

});

};

factory.getAccount = function (accountId) {

var options = { params: { id: accountId } };

return $http.get('/api/accounts/GetAccount', options)

.then(function (response) {

return response.data;

});

};

factory.createAccount = function (account) {

return $http.post('/api/accounts/CreateAccount', account).then(function (response) {

return response.data;

});

};

factory.deleteAccount = function (id) {

var options = { params: { id: id } };

return $http.delete('/api/accounts/DeleteAccount', options).then(function (response) {

return response.data;

});

};

factory.editAccount = function (account) {

return $http.put('/api/accounts/EditAccount', account).then(function (response) {

return response.data;

});

};

API Accounts

// GET: api/Accounts

[HttpGet]

[Route("GetAccounts")]

public Task<IList<Accout>> GetAccounts(string houseHold)

{

return db.FindAccountsByHouseHold(houseHold);

}

[HttpGet]

[Route("GetAccount")]

public async Task<Accout> GetAccount(int id)

{

return await db.SelectAccountAsync(id);

}

// POST: api/Accounts/Create

[HttpPost]

[Route("CreateAccount")]

public async Task<int> CreateAccount(Accout account)

{

var user = await um.FindByIdAsync(HttpContext.Current.User.Identity.GetUserId<int>());

return await db.InsertAccountAsync(account);

}

// PUT: api/Accounts/Edit

[HttpPut]

[Route("EditAccount")]

public async Task EditAccount(Accout account)

{

await db.UpdateAccountAsync(account);

}

[HttpDelete]

[Route("DeleteAccount")]

public async Task DeleteAccount([FromUri] int Id)

{

await db.DeleteAccountAsync(Id);

}

Interface

namespace AngularTemplate.InsightUserStore.DataAccess

{

[Sql(Schema = "dbo")]

public interface IAccountDataAccess

{

// auto procs

Task<int> InsertAccountAsync(Accout account);

Task UpdateAccountAsync(Accout account);

Task<Accout> SelectAccountAsync(int Id);

Task DeleteAccountAsync(int Id);

Task<IList<Accout>> FindAccountsByHouseHold(string HouseHold);

Task<decimal> SumTransactionsByAccount(int AccountId, string HouseHold);

C# Model

namespace FinalTemplate.Models.DataModels

{

public class Accout

{

public int Id { get; set; }

public string HouseHold { get; set; }

public string Name { get; set; }

public decimal Balance { get; set; }

public decimal ReconciledBalance { get; set; }

}

SQL