UNIT TESTING SERVER CALLS

init() function in controller

```
this.init = function() {
    var self = this;
    OnlineMortgage.get(function(obj) {
        $scope.onlineMortgage = obj;
        self.initMortgageAmount();
    });
};
this.init();
```

```
var scope, onlineMortgageStrategyCtrl, onlineMortgageMock;

// Before each unit test, instantiate a new instance of the
// controller
beforeEach(inject(function($rootScope, $controller) {
    // define the mock for onlineMortgage service
    onlineMortgageMock = {
        tranches: []
    };
    scope = $rootScope.$new();
    scope.onlineMortgage = onlineMortgageMock;
    onlineMortgageStrategyCtrl =
    $controller('OnlineMortgage.StrategyCtrl', {
        $scope: scope
    });
}));
```

```
describe('server calls', function() {
    var mockBackend;

    beforeEach(inject(function($httpBackend) {
        mockGetResponse = {
            "pk": 517543,
            "is_refinancing": true,
            "wished_mortgage_amount": 500000
        };
        mockBackend = $httpBackend;
        // get url from django params
        mockBackend.expectGET(window.urls.onlineMortgageDetailUrl)
            .respond(200, mockGetResponse);
    }));

    it('OnlineMortgage get test', function() {
        scope.onlineMortgage = undefined;
        // Simulate a server response
        mockBackend.flush();
        expect(scope.onlineMortgage.toString())
            .toEqual(mockGetResponse.toString());
    });

    afterEach(function() {
        // Ensure that all expects set on the $httpBackend
        // were actually called
        mockBackend.verifyNoOutstandingExpectation();
        // Ensure that all requests to the server
        // have actumockBackendally responded (using flush())
        mockBackend.verifyNoOutstandingRequest();
    });
});
```

UNIT TESTING SERVER CALLS

The $http service internally uses the $httpBackend to make the actual XHR requests.

The angular-mocks.js file provides a **mock $httpBackend service** that

1. prevents server calls

2. gives us hooks to set expectations

   mockBackend.expectGET('/api/note').respond(200, mockGetResponse)

3. allow to trigger responses

   flush()