We wpisie na temat podstaw Firebuga zapowiedziałem przyjrzenie się bliżej niektórym funkcjom i wtyczkom. I właśnie teraz to zrobimy.

Za pomocą Firebuga uzbrojonego w FirePHP (i FireQuery, ale to nie będzie miało aż takiego znaczenia) naprawimy prostą aplikację AJAXową - konkretnie funkcjonalność sprawdzenia dostępności loginu przy rejestracji.

Aplikacja powstała w oparciu o Zend Framework z wykorzystaniem jQuery. Użycie frameworków sprawia, że pisanie aplikacji jest bardzo proste i szybkie (choć okupione wcześniejszą nauką frameworka). Dodatkowo, dzięki obsłudze AJAXa w jQuery, sprawa będzie jeszcze prostsza.

Ale żeby nie było tak kolorowo, zasiałem do aplikacji pewien błąd. Spójrzmy, jak przebiegało debugowanie tej aplikacji, a za chwilę jeszcze kilka słów na ten temat.

http://www.youtube.com/watch?v=TH5DibnOdxo

Jak już wspomniałem, błąd został zasiany, aby pokazać prawdziwe debugowanie, a nie tylko gdybać. Oczywiście to, że znałem przyczynę błędu uprościło debugowanie ;) ale nadal zobaczyliśmy wszystkie ważne etapy - błędne działanie, wstawienie elementów diagnostycznych, diagnozę i naprawę.

Ważne fragmenty kodu

Konfiguracja w pliku application.ini

Aby móc korzystać z FirePHP należy wyposażyć Firebuga w odpowiednią wtyczkę, ale także dostosować serwer do wysyłania odpowiednich danych. W Zend Framework to bardzo proste,

co widać na filmie. Przy wykorzystaniu innych frameworków lub nie korzystaniu z frameworka w ogóle, odsyłam na stronę http://www.firephp.org/ po więcej informacji

{codecitation class="brush:text"} resources.db.params.profiler.enabled = true resources.db.params.profiler.class = Zend_Db_Profiler_Firebug resources.log.firebug.writerName = "Firebug"{/codecitation}

WAŻNE - w przeciwieństwie do tego, co widać na screencascie, nie należy tak konfigurować środowiska produkcyjnego. Więcej na temat konfiguracji środowiska w cyklu nt Zend Framework

Bootstrap ZF - inicjalizacja loggera

{codecitation class='brush:php'}protected function _initLog() { if ($this->hasPluginResource('log')) { $log = $this->getPluginResource('log')->getLog(); Zend_Registry::set('log', $log); } } {/codecitation}

Szablon kodu w NetBeans - wywołanie loggera

{codecitation class='brush:php'}Zend_Registry::get('log')->debug($result);{/codecitation}

Logowanie do konsoli Firebuga z poziomu JavaScript

{codecitation class='bruch: javascript'}console.log(dane);{/codecitation}

Inne poziomy logowania błędów to console.debug, console.info, console.warn i console.error.

Ajax jQuery

{codecitation class='brush: javascript'}$(function(){ $('#login').blur(function(){ //blur to opuszczenie pola formularza $.ajax({ 'url' : '/users/isavaliable', //adres 'type' : 'post', //typ żądania 'dataType' : 'json', //typ zwracanych danych 'data' : { 'login' : $('#login').val() //dane wysyłane, w tym wypadku, POSTem }, 'success' : function(response){ //funkcja obsługująca odpowiedź o statusie 200 if (response.success){ //tutaj success to pole odpowiedzi sformułowanej w PHP $('#login').css('border-color', 'green'); $('#submit').attr('disabled', false); } else { $('#login').css('border-color', 'red'); $('#submit').attr('disabled', true); } } }) }) });{/codecitation}

W tym ostatnim kodzie, prawdopodobnie lepsze od użycia styli byłoby nadanie odpowiednich klas - będzie to czystsze pod względem semantyki strony.