VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ANALÝZA A OZNÁMENÍ O NOVÝCH RESULTCLOUD VÝSLEDCÍCH

ANALYSIS AND NOTIFICATION OF NEW RESULTCLOUD SUBMISSIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE BOHDAN IAKYMETS
AUTHOR

VEDOUCÍ PRÁCE Mgr. Bc. HANA PLUHÁČKOVÁ
SUPERVISOR

BRNO 2016

Abstrakt

Abstract

The main goal of the project is to design and develop a mechanism for analyzing and notifying users about interesting changes in new uploaded submissions. The mechanism must support a few types of notifications and have possibility to extend types. The changes must also have interface for presentation results. Interested people would be able to set notifications to interesting results of the analyzes. It is important and useful because the largest part of test results are not interesting and useless information, like same testing results. Thus, the main goal of the analyzing is to find the interesting results and show them to user.
Analyzing of submissions is very important because a lot of results are useless, in most cases are the same data, so it doesn't give any important information. Analyzing helps save developer's time, it finds useful information and notifies developers or other interested people about it and thus anyone at any time can easily find needed information, or to see statistics of project.
Firstly, I must learn inner architecture of ResultCloud. How it works. It helps me better use all opportunities in design and programming that mechanism.

Klíčová slova

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Citace

Bohdan Iakymets: Analýza a oznámení o nových ResultCloud výsledcích, bakalářská práce, Brno, FIT VUT v Brně, 2016

Analýza a oznámení o nových ResultCloud výsledcích

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

....
Bohdan Iakymets
3. května 2016

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant, apod.).

Obsah

Kapitola 1

Introduction in ResultCloud

1.1 What is ResultCloud

ResultCloud is a system for management of long-term testing results. This means that ResultCloud collects testing results for some project and compare it, so developer can easily find difference between them. Currently, there don't exist new, modern instruments for the collecting and the presentation testing results in readable form. As written by Fillip Matys: "Tools which solve that problem [3], are too old and fall behind all modern applications. One of the biggest problem of that tools is no opportunity to expand and with growing market of different mobile devices with internet connection not able to present data in responsive form." So all instruments which we have now is too old, and doesn't extendable. But in ResultCloud parsing and management doing by modules. Each module is written for one type of testing results. For example module "DejaGnu summary v1.0" can parse and show only SystemTap results.

ResultCloud is a complex system. ResultCloud consist of two parts frontend and backend. Frontend is a part on the client side, made with the use of AngularJS. Angular in asynchronous format connecting with backend part. Backend also dividing on the smaller part. On the top of hierarchy are Controllers. Angular connect directly with controllers. Controllers has only one mission is to get request and parsed data give to Services. Services are very important part, they get a data and using different database entities, other services, plugins for returning a result to the controller, which return it to client.

For connecting with database exists Driver which first of all connecting to database, and after that connection using data access object (DAO). DAO consists all basic methods for working with database. Every entity must have DAO, which inherit base DAO, and then system will work with entity through entity's DAO. Results of SELECT query converting to TSE (Test-Suite-Entity) object. TSE object helps easier work with entities.

Data organization in ResultCloud is represent in Plugins. Plugin is the system for parsing and saving test's results of specific format. In the past plugins has all needed for self-installation, parsing and vizualization data, now vizualization is shared. Each plugin has own implementation of Parser to parse input data, it is saved to a Project, Submission, etc. Hierarchy of shared entities is following: Projects contains Submissions, Submissions is a results of a single series of tests, which also divide to Categories, Categories are divided to TestCases, and TestCases to Results.

In this bachelor work I will analyzing Submissions, like their Results, and other stuff that may be interesting for people.

1.2 What is Submission

As I wrote at the top, the submission is a single testing result. The smallest part of every submission is Result. This part contains result of a single test from series of tests. All other part like Categories, TestCases are only organization unit.

There are two ways to import a new Submission: first is from a web page, second is using the API. Then will include plugin for parse submission file. Every plugin has a class Parser, for parsing files and putting them into Database (DB). When client send file to `ImportController` or to `import` class, it call `ImportService`, which find needed plugin in DB, then include plugin's class `Parser` and call method *ParseImport*. Parser return to `ImportService` `SubmissionTSE` object, which consists all parsed data as TSE objects. `ImportService` than save it to DB and return successful result to Controller or API class.

Kapitola 2

Analyzer Mechanism Design

All analyzers must somehow unite into one working system. There must be mechanism for that. Mechanism must be not complicated and easy to extending. Thus it must easy to controll all analyzers and work with their results.

2.1 Architecture

There two type of architecture: module and built-in. Module type mean that Mechanism would be divided to the modules, like "Divide and Conquer", one of the main advantage is easy extending. Second type is built-in, which means mechanism would be built-in whole ResultCloud system, one of the main advantage of this type is working speed.

I prefer first method, because difference in speed beetwen them would be to small, but easy extedning advantage is that what mechanism need. So lets start from main part, kernel of whole mechanism, `AnalyzerController`.

2.2 AnalyzerController

`AnalyzerController` would get all existing analyzers and use it. Mechanism also would provide entity for saving analyze data. Analyzer can't work with DB because in practice it is usually to divide work between separated modules, like "Divide and Conquer", so analyzer should only analyze input data and visualize it, that all. Centralized method good for that case because user don't need to load needed analyzers and work with DB, all this stuff do `AnalyzeController`.

2.3 Analyzer entity

In the picture presents entity Analyzer, which contain four attributes. Attribute *Submission* would have ID of the submission, that analyzer results belongs to. *Project* is alternative attribute to *Submission*, it would have ID of the project. *Analyzer* contain machine analyzer ID. And *Result* contain analyzer results, it is text attribute, every analyzers have own output results format.

2.4 AnalyzerController structure

As can be seen in the image, `AnalyzerController` is center part of whole analyzer mechanism. When application start `AnalyzerController` find all available analyzers, it is good for optimization. In `AnalyzerController` would be implement easy analyzer controlling. One method must run all analyzers which support current submission's plugin, and return result which `AnalyzerController` write to DB. Methods for vizualizating data.

2.5 Analyzers

2.5.1 Analyzer design

For right connecting with analyze controller, analyzer must have first of all static constant attribute with unique machine ID (under it ID, analyzer would be identified in Analyze entity), method for getting and processing data (name of the method must be the same in all Analyzer classes) and two functions for vizulizating data (Visualize, VisualizeSingle). Vizualizating functions will get data from AnalyzerController which get it from DB and return it in JSON.

Method for processing data get in parameters: array of submission, new submission and plugin name. It must return ValidationResult object, with string in Data attribute, or array of string if it has a few resuls, or it can return empty result, with null in Data attribute.

Also analyzer have attribute called *is_interesting* which contain status of previous analyzing, and if analyzing results will be interesting it return true, in other case false, it need to notifying about only interesting submissions.

2.5.2 Analyzer vizualisation

Every submission has own analyzing results, that results will be presenting in individual page. Every analyzer have some space on that page. Because different analyzers use different methods for vizualization there is needed to specificate different possibilites to vizualize information. As mentioned in first chapter for vizualizating data ResultCloud use AngularJS and templates, so every analyzer must have own template and AngularJS directive. On the image can be seen analyzers results page layout.

2.5.3 Kinds of Analyzer

Next step is propose some kind's of analyzers. Here is analyzers which results would be interesting for programmers.

- Find strange changes like if result has a long time the same value and than it change
- Check a changes in tests, like if some test which is contained in all previous submissions just dissapear
- Check if some test had a long sequence of some bad value like FAIL or ERROR and then take a PASS, but after take FAIL or ERROR again
- Check changes from UNTESTED to some result
- Check if presented a new tests

- Check strange changes like FAIL -> ERROR
- Check if count of bad results is get maximum

Kapitola 3

Analyzer Mechanism Implementation

Mechanism implemented in PHP and JS, becaue that languages was used for implementation ResultCloud.

3.1 Structure

Because analyzers are not a plugins or any else components in ResultCloud, analyzers will be *extentions*. Whole system have own directory *analyzing*. Which contain one directory for analyzers - *analyzers*, and one for templates - *templates*. Root directory also would contain `AnalyzeController`.

Analyzing starts only when new submission would be inserted into DB, in `ImportService` class.

3.2 Entity

Analyzer entity was converted into ResultCloud acceptable format. As a result was created three classes: `AnalyzerDao`, `AnalyzerTSE`, `AnalyzerService` and edited table installation class. `AnalyzerDao` class for working with *Analyzer* table. `AnalyzerTSE` class for easy working with `AnalyzerDao` returned data. `AnalyzerService` class for different more complicated operations with data.

3.3 Analyzing

3.3.1 AnalyzeController

`AnalyzeController` is a kernel of whole analyze mechanism. `AnalyzeController` implemented like static class (but PHP does not support static classes, thus all methods are static), because create more than one class object unnecessarily.

When `AnalyzeController` be included, it execute *InitAnalyzers* method, that scan *analyzers* folder, put all available analyzers together and save it to *$AnalyzerList* attribute. Method *GetAnalyzersList* will return `LINQ` object with *$AnalyzerList*.

analyze method get all analyzers from *$AnalyzerList*, and call it analyzer method. Than returned value, or values it write to DB, and check if results are interesting by getting boolean value from analyzer method *isInteresting*, if results are interesting it add analyzer ID to *$interesting_analyzers* array. Analyzers ID which has interesting value can get by method *GetInterestingAnalyzers*. Method *analyze* get like parameters: currently uploaded submission - *$submission*, `LINQ` object with older submissions - *$submissionList* and plugin name - *$plugin*. Returning `ValidationResult` object with the analyze status.

3.3.2 Analyzer1

`Analyzer1` is simple analyzer created like example of analyzer structure. Analyzer get new submission and the last one, and compare it, if it has differences, analyze results became interesting. There are three categories of changes: GOOD, BAD, STRANGE. Output result format is JSON. It support only s̈ystemtapp̈lugin.

Every analyzers must have method *analyze*, which analyzing input submissions according to plugin name. Parameters are the same as has method *analyze* in `AnalyzeController`. Output results are in format JSON, it use JSON because of it simplicity. Attribute *$is_interesting* is boolean type, and became *true* only if analyzing results are interesting, otherwise it's false. As mentioned in previous part, analyzer has method *isInteresting*, which return value of *$is_interesting* attribute.

ANALYZER_ID is constant attribute, that contain unique analyzer ID, that ID is used in *Analyzer* table, like analyzer identifier. *JS_CONTROLLER* is also constant attribute which contain name of JavaScript file with AngularJS directive, it is used for vizualization analyzing results.

3.4 Vizualization

For vizualization data ResultCloud use AngularJS. AngularJS is JavaScript MVC (Model-View-Controller) framework, every page has own controller, thus analyzer page must have it too. `AnalyzeController.js` is file that contain controller for result page. *analyze.html* is a page template. Some of the page, that contain several sort of data, building with the simplest part *Components*, each component has individual settings, and individual Angular `directory`. Each component has `backend`, `frontend` folder and configuration file *config.xml* with all settings and supported plugins. Backend folder consists `CBuilder` class, which return prepared for presenting data.

Analyzer page would use only one component `analyzeOverview`.It wouldn't have any settings, and will support all plugins. Angular directory first of all get array of analyzing results for current submission, than for each analyzer find own Angular directory, which put analyzer data to template and present it.

`CBuilder` class for `analyzeOverview` get `stdClass` object with attribute *Submission* - submission ID. And call `AnalyzeController` method *VisualizeBySubmission*. *VisualizeBySubmission* get submission ID, for each analyzer get last inserted result, and give it to analyzer's method *VisualizeSingle*, which parse results and return it like array. Then *VisualizeBySubmission* put vizualization data together into associative array the key analyzer ID and value analyze results, and return it.

Kapitola 4

Notification design

Like in case with Analyze Controller, I would divide notifications methods to the separated classes and Notification Controller will controller them. NotificationController will collect all Notifiers with the help of static function *preLoad*. For notifing exist function *notify(title, body, bodyShort, to)*, where `title` - is title of notification message, `body` - longest body text, `bodyShort` - short body of the message not longer than 140 letters, `to` - is an array of all adresses with the key of notifier ID. Function *notify* calling notifier's function *notify*, only for notifiers which have their IDs in `to` parameter's key.

Also notifications divided to: Public and Private. Public notifications, is notifications which would be sent to public resources, like RSS or Twitter. Private notifications would be sent to private persons, like e-mail for exammple.

Private notifications have settings for each user. And user can subscribe to each private notification.

For Twitter notification is using TwitterAPIExchange with MIT license.

Kapitola 5

Notifier

Notifier is class for notification users in specific way. Notifier class is extend from BaseNotifier class.

BaseNotifier class has only one function which will be in most cases the same in private notification. This is *getSettings* function, which will returning needed settings field for each private notifier.

Notifier also have function *notify*, which get parameters from NotificationController and send all needed notification messages.

Kapitola 6

Notifiers

In that moment in ResultCloud are Email, Twitter and RSS notifiers.

Each user has setting for email notification. If he want to get that type of notifications he can turn on this option into his settings.

Twitter and RSS notifiers are public notifiers so it don't need to get settings for each user. RSS notifier show news about latest interesting submission in each project, with URL to it. Twitter has short description with URL to the page with analyzer results to.

Přílohy

Seznam příloh