WEB TESTING APPLICATION WITH PHP AUTOMATED TOOL

Iulia Ștefan and Ioan Ivan

Department of Automation, Technical University, Cluj-Napoca, Romania
Iulia.Stefan@aut.utcluj.ro, ionut.ivan1@gmail.com

ABSTRACT

The web applications development has experienced an explosive growth in variety and complexity during the past decade. Most web-based applications are modelled as three tier architecture, the client side experience remaining virtually unchanged, while server-side is updated. However, client-side architecture can change with unexpected results. Consequently, testing procedures should support continue improvements to pursue the current trends and technology. This paper presents an automated tool for testing client-side component of web applications. The testing data is extracted using a crawler. Adopting several procedures, the general aspect of the page is analysed (CSS regression testing). All of the content is tested, including links, images, forms, and scripts. The resulted test cases are automatically created, leaving the user with the option to decide over their usage.

KEYWORDS

Automated testing, web application, PHP

1. INTRODUCTION

The development of the web applications has become an important area in the field of software engineering. Web applications have a core set of specific characteristics like modularity, by which different functionalities of the same product are written in an in-dependent manner. Their reusability for different tasks in the same application is a viable cost reduction solution. A notable advantage is that modules can be created using a variety of technologies deter-mining the performance improvements.

The techniques used for developing web applications have become more and more varied. The enhancement in mobile devices industry in the past few years has created the need for web applications to accommodate both regular clients, and mobile ones. Web application responsiveness has increased because data transfer has been made more and more efficient. The new technologies have driven the validated concepts to new heights and old technologies have been updated to keep up with the fast pace. JavaScript libraries are the best example, by their later evolution.

The diversity of development technologies has led to a growth in web application testing technology. In spite of this fact, testing has become more difficult. A complete solution can consist of several modules written in several languages, there is no single tool for testing the whole application. Web applications are made from static and dynamic components. The static

Natarajan Meghanathan et al. (Eds) : CSEN, ADCO - 2014
pp. 15–21, 2014. © CS & IT-CSCP 2014 DOI : 10.5121/csit.2014.41002

pages can be tested automatically with the use of crawlers or other spider-like tools. The dynamic content is difficult to be analysed in an automated approach.A short list of techniques is available. A record and replay tool is one solution. Such software tools record an initial testing scenario and then generate test scripts for automated regression testing. A major drawback of this technique is that if there is a change in the user interface, the generated scripts have a greater chance of failing. Another disadvantage is the cost associated to manual recording of scripts. However, it is relatively easy to deploy and it is autonomous, allowing the tester to deal with other tasks. In this paper, an automated tool that extracts information from the user interface and submits it to the user analysis is described. In section 2, a short description of similar tools is provided. In section 3, the aspects of testing activities are reviewed. Section 4 provides an insight related to the development of the present-ed tool. In section 5, experimental results are de-tailed, followed by section 6 with conclusions and last section, References.

2. GETTING STARTED

The domain of software testing represents an interesting research domain and application development.

Every element of a web application is considered an object [1]. The test cases are generated using as starting point the data flow chart.

The comparison between developing tests cases by programing or by capture-relay is emphasized by [2]. After several experiments, the conclusion evidenced the advantage of cost reduction in test maintenance by programing approach and the advantage of reduced time allocated when development was centered on the capture-relay approach. The tools used for functional web testing in this case were Selenium IDE and Selenium WebDriver.

The development of a new testing tool was enhanced as presented in [3] by using models. The test cases can be automatically or manually selected by request. In [4], the authors adopt a different approach. The Ajax events are used to design a state based-approach using the DOM as the template. Generated events determine the transition between states, and the interaction chain generates the test cases. An approached based on image analysis [5] detects defect images and broken links. Several testable objects are identified by the application decomposition. The image processing techniques could be an important aspect for automated CSS testing. The main concept is the usage of a static test copy of the web page when testing in order to separate the functionality from the shape.

The PhantomCSS tool is also presented. The tool generates an image of a portion of the page and the then establishes differences against the original.

3. TESTING TECHNIQUES

The main objective of testing activities is to detect and fix faults and demonstrate the product quality against stakeholder's requirements and application's specifications. Several stages are necessary: the test planning, the goal definition, the general strategy selection, the test execution, and the result's analysis. The testing process tends to start early during the development stage and continues after deployment. For the web applications, the requirements could be reduced in size and test models are not established.

There are two main approaches regarding software testing based on code visibility and are summarized further on.

3.1 White Box Testing

In white box testing, the source code is available in its entirety to the tester giving extra insight on the application and removing the not clarified particular piece of code. White box testing is most useful for internal unit verification. The tests are designed to highlight the application's behavior related to specific units.

However, the tools available for this approach are restricted to a handful of programming languages and technologies: java, C#, C++, PHP, python and several others.

If the frameworks are not maintained and updated constantly, the incompatibility with newer technologies is unavoidable.

3.2 Black Box Testing

The black-box testing approach (BBT) is also known as client-side analysis; there is no code available. The structure of the website becomes available by sending requests to the server and examining responses. This method cannot make a difference between static and dynamic pages without appropriate tools.

The simplest form of black box testing is to start running the software and make observations if expected and unexpected behavior is be easily distinguishable. Abnormal behavior, such as crashes are easy to spot. As soon as the cause has been determined to be part of the program, relevant information can be passed to a competent party for fixing.

Another, more advanced form of black box testing is the use of checklists. The checklists represent specifications about expected behavior based on applied known inputs. The term input stands for any action or resource provided at the beginning or during the runtime. If problems are discovered, specific actions are implemented in order to fix the issue.

The developed tool offers support for the black box testing techniques identifying test cases to be considered when creating tests. Regarding CSS verification, regression testing is to be used. Regression testing aims to uncover new faults in existing functional or non-functional parts of an application after changes such as patches, upgrades or configuration modifications have been made. A common method of regression testing is the rerunning of previous tests to check if the behavior of the application has changed in conflict with the existing specifications. This method can be used to test the correctness of a program as well as the quality of the output. A downside of regression testing is the cost. In theory, a complete test suite should be run after each update of the application, which is too resource-intensive to be applied [6]. In a real development and testing environment, a minimal set of tests is devised to check the specific functionalities that have been altered.

Other than client specifications, there is no need for discovering the entire application structure. The only situation where the server-side is involved is when the link response is checked. For the form and button analysis, the proposed tool is using pre-generated data (for form input).

4. DEVELOPMENT

A simple web application used for online order placement is used as software under test (SUT). The SUT's structure consists of several PHP files linked together. The content (images and text) is retrieved from a database. A common, simple style is used throughout the website. It is used because it offers a minimal structure, simple Ajax processing and a simple CSS file, which makes

it easier to analyze initial results. When all major issues have been resolved, larger, more complex websites should be tested.

Using a PHP page as an interface, the tester is able to choose between several options as shown in the Figure 2.

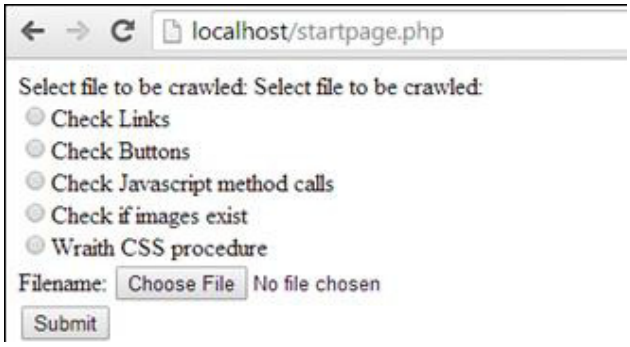

Figure 1. The user interface page for test option

The links and images are analyzed first. Using phpCrawl 0.81, an open source PHP library functions package, the target page is read and its content is saved for further use. From this content, links can be extracted. The application workflow is presented in the Table 1.

As shown in Figure 1, based on the server response when the crawler attempts to access them, links are classified as “good” (200 response), “bad” request (400 response), “forbidden” (403 response), not found (404 response) or not allowed (405 response). For the forbidden or not allowed responses, the user is prompted to check the rights for the called page. For the bad syntax response, the link is submitted for direct analysis. For the not found response, a mock page can be generated using information from the link itself.

The page only echoes any parameters that are passed to the page. This mock page is only a placeholder until the real page can be added to the server. The images are processed in a similar manner. Their existence is checked, and if they cannot be found, a temporary picture is inserted as a placeholder. A syntax check is performed and if there are any function callbacks in the <img> tag, the function name is searched for. If the function does not exist in the extracted content, the user is notified. An option is available to ignore any callbacks and only check for basic syntax. Forms and buttons are also extracted from the page content. To verify any validations that may be employed in the form, a correct set of inputs is given. If the response is good, the form is considered correct.

Forms and buttons are also extracted from the page content. To verify any validations that may be employed in the form, a correct set of inputs is given. If the response is good, the form is considered correct. This creates a term of comparison. Then, a wrong set of inputs is given. If there is an event such as a redirect, or a simple html required validation, the user is notified. A syntax check is also performed. The amount of time spent for CSS regression testing can be lengthy if the process is not automated. Verifying different browser resolutions can be time consuming in testing. By using automated tools, the testing time can be cut down dramatically, with fewer bugs making their way into ready to deploy products or live applications.

For the CSS verification, Image Difference Comparison is used. As the customer settles on a general design, a mock-up page can be created for later editing. This starting point can be compared to an existing template. Snapshots are taken from both the mock-up and the template at

the same state. These snapshots are then compared and differences are highlighted. For more detailed results, snapshots can be taken for each element in particular (i.e. a snapshot of the footer section).

Table 1. Application workflow

Steps	Operation	Description
1	Start page	Entry point of application
2	Acquire content	Extracts the content of page for analysis.
3	Content verification	Includes images and links.
4	Acquire template for aspect checking	Applies CSS regression testing to the page.
5	Compare empty page to template	Applies CSS regression testing to page as a whole.
6	Break down template and page into individual components and redo tests.	This ensures every element is correct.
7	Dynamic content testing	Ensures that content is generated properly.
8	Parse page for scripts and external functions	Find where the functions are being called, if any exist.
9	Find external functions by name	Compares function calls to a list of function names from external libraries (jQuery, prototype, etc.)
10	Find functions by name	Either finds function definitions in page or any external file.
11	Compare expected response with actual response	Can be done using image comparison techniques.
12	Check validators	Important for safety.

A more focused diff can be generated in this manner, but it may take away the more general view. This remains a user preference. Again, there is no need to know the whole structure of the site. This process can be repeated at any stage of the development process. For this issue, Phantom CSS is used because there is no current need to have a very complicated tool for analysis. It handles the image comparison section required by the CSS testing. As an alternative, Wraith front end regression testing tool can be employed. It is somewhat similar in functionality to Phantom CSS. The difference lies in the components. While both tools use PhantomJS as a headless browser to move through the site, Wraith uses a tool called ImageMagick for snapshots. Wraith also has the ability to create screenshots of different resolutions.

Other responses are more difficult to automatize because of either syntax or server issues. The Image Difference Comparison method had serious initial issues. Because it is based on image comparison, even the slightest modification compromised the results. For example, in a full page test suite, a 2 pixel wide padding added around the footer caused all the tests to fail. In such situations, individual component testing proved to be more reliable and more time efficient. Warnings were added to the user interface, but the choice is left to the user.

In the first runs, tests failed continuously because of the images in the site. This called for a different strategy. After the creation of the mock page, all visual content was replaced with a

blank space and then the page was compared with the template. The method yielded improved results, and false test failures were mostly removed.

5. EXPERIMENTAL RESULTS

The link analysis method has proven effective especially when the 404 response is returned. Mock pages are generated automatically if the user wishes to do so. After choosing the option from the user interface page as presented in Figure 1, the tester receives a response as shown in Figure 2.

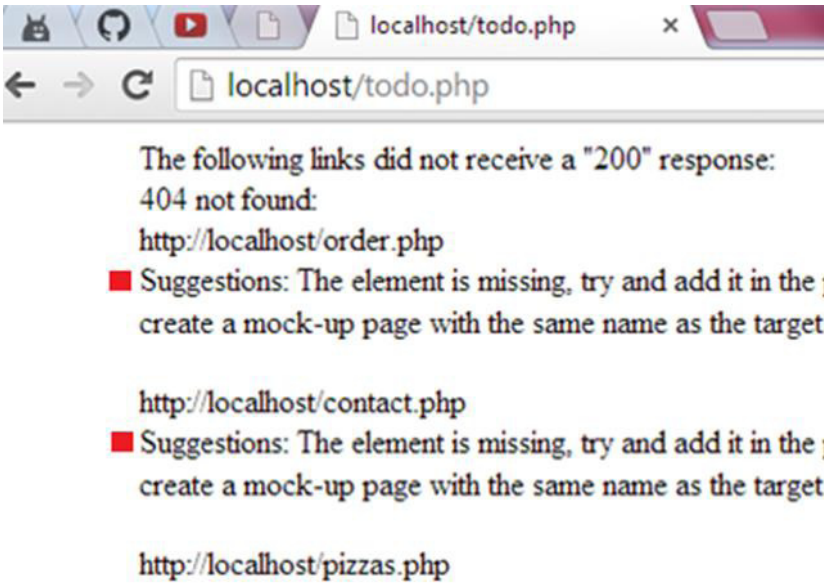

Figure 2. 404 Type error message with suggested options Web page

All links and images are obtained by the use of a crawler. They are all checked for syntax and existence. If a target of a link or an image can be replaced with a mock target, the user is to be notified.

All forms and buttons are tested. Form data is supplied by the user. Changes in the page are recorded and the user is notified. Buttons are tested for correct syntax. CSS testing generates a flat page and compares snapshots of it to a given template. Any differences are highlighted. Current work is largely focused on image processing and Ajax content analysis. The main goal now is to verify any function (i.e. JavaScript functions) for correct functionality.

The tool is under development stage and does not have a stable release. Functionalities may be added or removed to extend its usefulness.

6. CONCLUSION AND FUTURE WORK

An approach for testing web applications is presented in this paper. The front end of the application is decomposed and analyzed.

REFERENCES

[1] Kung, D., Liu, C. H. & Hsia, P.(2000) "A Model-Based Approach for Testing Web Applications", Proceedings of the Twelfth International Conference on Software Engineering and Knowledge Engineering, 6-8 July 2000, Chicago, USA, Chicago, Knowledge Systems Institute.

[2] Leotta, M., Clerissi, D., Ricca, F. & P. Tonella(2013) "Capture-Replay vs. Programmable Web Testing: an Empirical Assessment During Test Case Evolution", 20th Working Conference on Reverse Engineering, October 14-17, Koblenz, Germany, IEEE Computer Society.

[3] Boumiza, D.S. & Ben Azzouz, A. (2012) "Design and Development of a User Interface to Customize Web Testing Scenarios", Proceedings of The International Conference on Education and e-Learning Innovations, July 1-3, Sousse, Tunisia, New York, Institute of Electrical and Electronics Engineers (IEEE).

[4] Marchetto, A.,Tonella, P. & Ricca, F (2008) "State-Based Testing of Ajax Web Applications", 1st International Conference on Software Testing, Verification, and Validation, Lillehammer, Norway, April 9-11, 2008, Los Lillehammer, IEEE Computer Society.

[5] Torkey, F.A., Keshk, A., Hamza, T. & Ibrahim, A(2007) "A New Methodology for Web Testing", 2007 ITI 5th International Conference on Information and Communications Technology (ICICT 2007) - Media convergence: Moving to the next generation, New York: Institute of Electrical and Electronics Engineers (IEEE).

[6] Brooks, F (1995), "The Mythical Man Month: Essays on Software Engineering(2nd Edition)", Essex, Addison-Wesley.

AUTHORS

Eng. Iulia Ştefan, PhD. student, is a member of the Automation Department, Faculty of Automation and Computer Science, Tehnical University of Cluj-Napoca, Romania since 2006, being inv olved in several research projects covering web technologies, online platforms and software testing.

Eng. Ioan Ivan is a Master Degree student in Computer Science at the Technical University of Cluj-Napoca. His area of interest covers web technologies, testing, and more recently, the mobile development field.