УДК 621.327:621.391 DOI: 10.26565/2519-2310-2019-4-05

ОПИСАНИЕ СХЕМЫ API ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Ольга Мелкозерова, Валерия Гайкова, Сергей Малахов

Харьковский национальный университет имени В.Н. Каразина, пл. Свободы, 4, Харьков, 61022, Украина
olja.mex@gmail.com, valeriagaikova98@gmail.com, mailgate@meta.ua

Рецензент: Георгий Кучук, д.т.н., проф., НТУ «ХПИ», ул. Кирпичова, 21, Харьков, 61000, Украина.
kuchuk56@ukr.net

Поступила: Январь 2020

Аннотация. *Для оценки качества приложений применяют API тестирование. Существует большое количество инструментов для его проведения (Postman, SoapUI, Jmeter и т.п.). Универсальный подход к тестированию усложняется большими объемами данных, наличием разнообразных методик и инструментов, также необходим объект тестирования. Сложность логики работающего приложения приводит к трудностям понимания процесса тестирования, то есть это затрагивает вопрос подготовки специалистов в этой области. Для упрощения понимания, в качестве объекта тестирования было создано приложение «Калькулятор» при помощи класса Servlet (Java). Тестирование проводилось при помощи инструмента Jmeter и написания кода на языке Java. Составлен тест-план с использованием инструмента JMeter, позволяющий отправлять запрос приложению. Отмечено, что автоматизированные модульные тесты находят большее применение при разработке программного обеспечения, поэтому на языке программирования Java показана возможность написания аналогичного запроса.*

Ключевые слова: *автоматизированное тестирование; инструменты автоматизированного тестирования; технологии автоматизированного тестирования; JMeter.*

1 Введение

Многие современные приложения используют API (*Application Programming Interface*) для обеспечения их взаимодействия с сервером и интеграции друг с другом, поэтому для их тестирования необходимо владеть инструментами и техниками тестирования API, которое обладает рядом преимуществ [1]:

1. *Раннее тестирование* – разработчики приложений сначала делают API, а потом уже графический интерфейс GUI (*Graphical User Interface*). При этом обеспечивается возможность проверить логику раньше, чем появится GUI.
2. Графического интерфейса может в принципе не быть.
3. Высокая скорость – вызов одного запроса занимает доли секунды. При этом через интерфейс повторить процедуру бывает сложно.
4. Возможность автоматизации – даже если нет авто тестов на уровне API приложения, их всегда можно создать вручную, посредством PostMan и JMeter [2, 3].
5. GUI и его содержание постоянно меняется, однако логика работы приложения остается прежней.

В данной работе на примере простейшего приложения «Калькулятор» демонстрируется возможность тестирования API путем написания автоматизированных тестов на языке программирования Java при помощи инструмента JMeter.

2 Сборка приложения «Калькулятор» при помощи Servlet

Для демонстрации возможностей тестирования производительности и тестирования API на языке Java при помощи Servlet (*сервлет*) было собрано простейшее приложение «Калькулятор». Сервлет – это класс, который расширяет функциональность класса HttpServlet и запускается внутри контейнера сервлетов [4].

Сервлет размещается на сервере, однако чтобы сервер мог использовать сервлет для обработки запросов, сервер должен поддерживать движок или контейнер сервлетов (*Servlet Container/Engine*). Например, Apache Tomcat по сути является контейнером сервлетов,

поэтому он может использовать сервлеты для обслуживания запросов.

Для обработки запроса в HttpServlet определен ряд методов, которые мы можем переопределить в классе сервлета:

- *doGet*: обрабатывает запросы GET (*получение данных*);
- *doPost*: обрабатывает запросы POST (*отправка данных*);
- *doPut*: PUT (отправка данных для изменения);
- *doDelete*: DELETE (*удаление данных*);
- *doHead*: обрабатывает запросы HEAD.

Для создания приложения используем сборщик проектов Maven. Maven - фреймворк для автоматизации и сборки проектов на основе описания их структуры в файлах на языке POM (*Project Object Model*), который является подмножеством XML [5].

Структура проекта отображена ниже, на рис. 1, а скриншот приложения, которое подлежит тестированию, представлен на рис. 2.

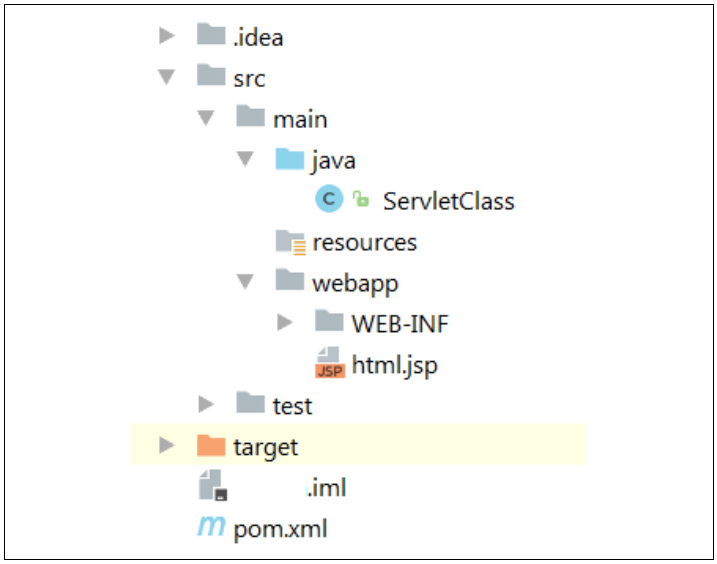

Рис. 1 – Структура проекта *Servlet*

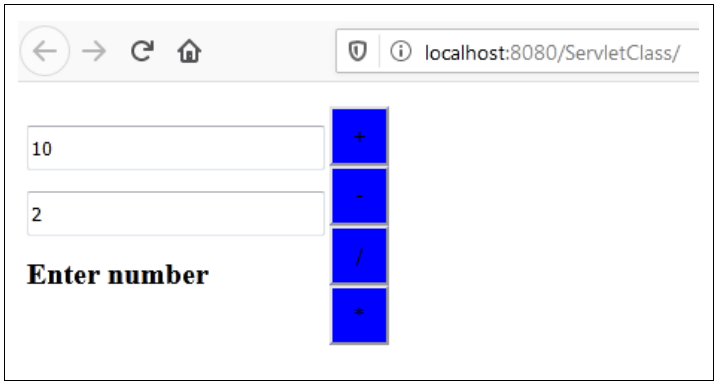

Рис. 2 – Скриншот работающего приложения «*Калькулятор*»

3. Написание API запросов к приложению

3.1 Использование приложения Jmeter для тестирования производительности и написания запросов.

Apache JMeter – открытое программного обеспечение, которое используется Java для тестирования функционального поведения и оценки производительности приложений [2].

Рассмотрим пример, в котором для исследования производительности приложения использован Apache JMeter версии 2.11.20151206 (см. Рис. 3).

Как было отмечено выше, тестированию подвергалось приложение «Калькулятор», тест-план которого представлен на Рис. 3-4. Перечень элементов, из которых возможен синтез тест-планов в JMeter выглядит следующим образом:

- Группы потоков (*Thread groups*);
- Логические контролеры (*Logic controller*);
- Типовые контроллеры (*Sample generating controller*);
- Слушатели (*Listeners*);
- Таймеры (*Timers*);
- Соответствия (*Assertions*);
- Конфигурационные элементы (*Configuration elements*).

Thread groups – начальные точки любого тест-плана (Рис. 3). Все контроллеры и образцы должны быть в группе потоков. Другие элементы, такие как слушатели, могут располагаться под тест-планом, в котором они применяются для всех потоков групп. Элемент группы потоков управляет количеством потоков, который JMeter будет использовать для выполнения теста.

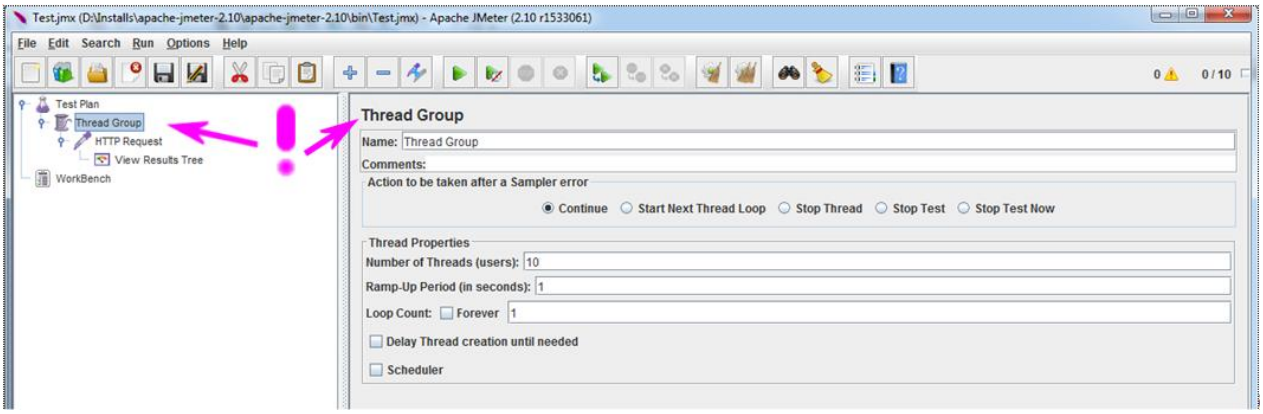

Рис. 3 – Заполнение формы Thread Group *(количество пользователей – 10)*

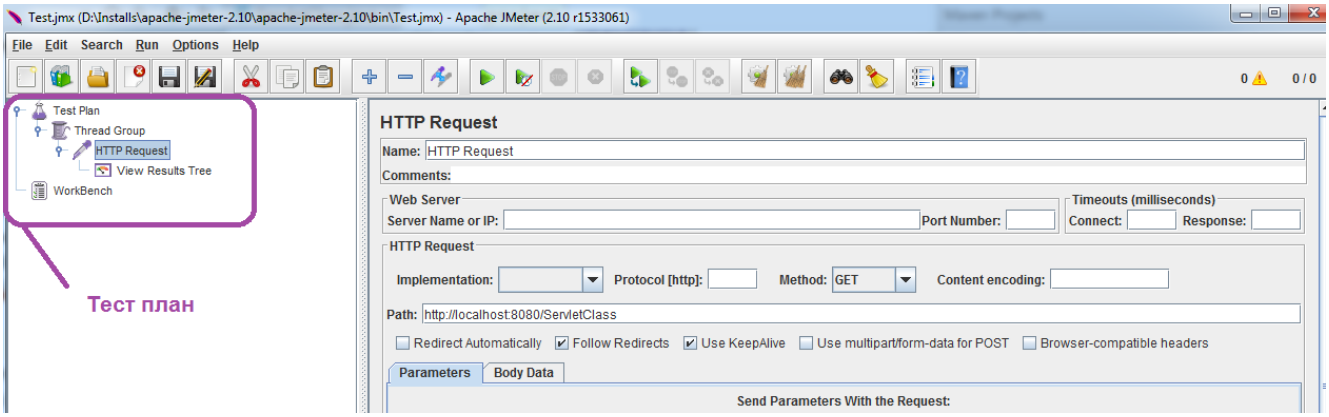

Рис. 4 – Заполнение формы HTTP запроса

Для отправки запроса необходимо заполнить форму HTTP запроса (Рис. 4), в котором следует указать ссылку на работающее приложение *(http://localhost:8080/ServletClass/).*

Listener – это компонент, который показывает результаты образцов. Сами результаты могут отображаться в дереве, таблице, графах или быть записаны в лог-файл. В данной статье для записи результата использовался Listener *Tree Graph Results.*

Tree Graph Results – дерево ответов всех образцов, позволяет увидеть отклик для всех образцов. В дополнение к показу отклика, индицируется время этого отклика (см. *вкладку «Sampler Result» на Рис. 5*), код отклика («200»), *HTML формат приложения, Рис. 6.*

Jmeter используется для написания API запросов, но его также используют для тестирования производительности. В целом могут быть определены различные типы тестов, которые зависят от целей тестирования [1, 6].

Термин «*тестирование производительности*» (*Performance Testing*) включает в себя любое тестирование, которое сфокусировано на производительности системы (*ответной реакции, скоростных показателях*) или компонентов под различными объемами и характерами нагрузки.

Нагрузочное тестирование (*Load Testing*) фокусируется на способности системы справляться с возрастающими уровнями ожидаемой нагрузки, возникающей в результате запросов конкурирующих пользователей или процессов. Кроме того, это и исследование робастности (*запаса прочности*) – способности системы сохранять заданные показатели качества, как при допустимых пределах нагрузки, так и при их некотором превышении.

Стрессовое тестирование (*Stress Testing*) сосредоточено на способности системы или компонентов поддерживать пиковые нагрузки, которые находятся на уровне или превышают

пределы ожидаемых. Фактически, это исследование поведения приложения при «аномальных» изменениях нагрузки. Данный тип тестирования также используется для того, чтобы оценить способность системы поддерживать такие показатели, как: - доступная компьютерная мощность; - доступная полоса пропускания; - доступная память.

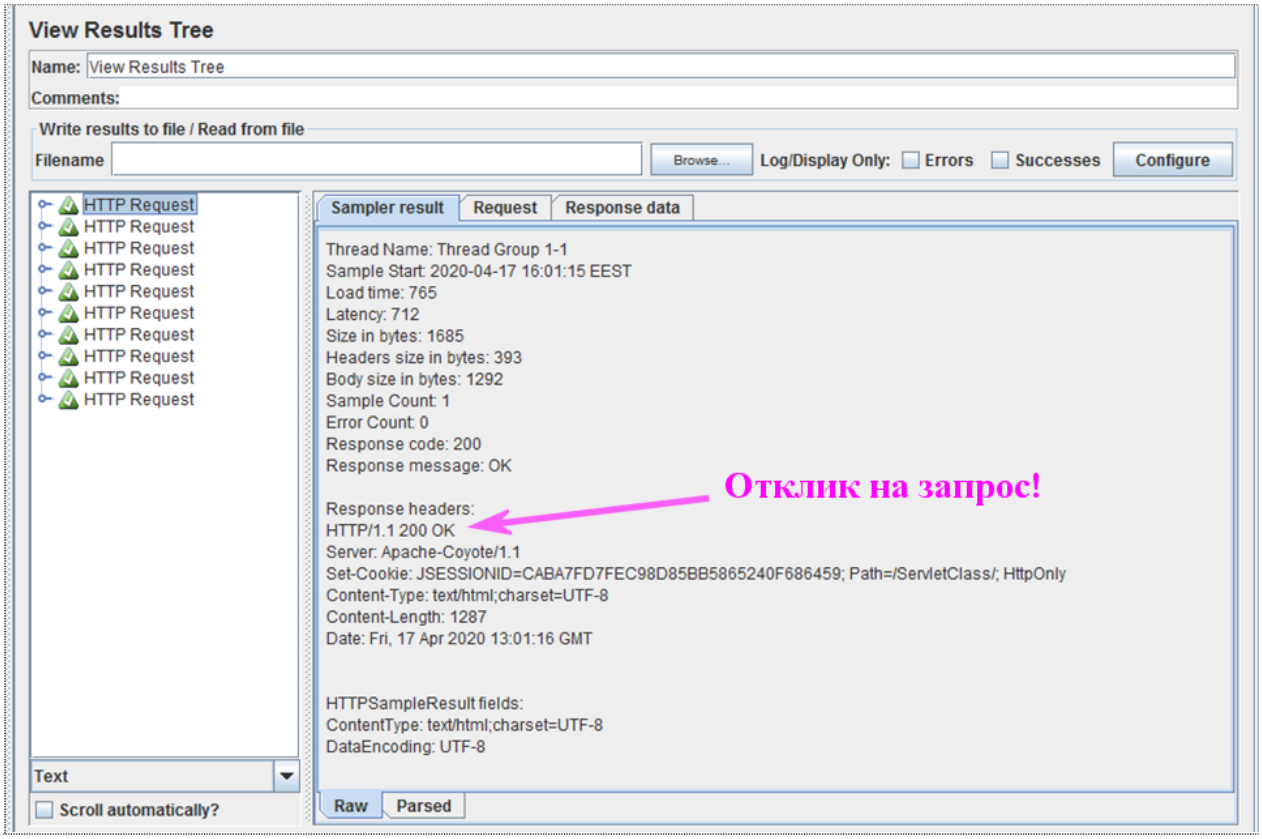

Рис. 5 – Результат GET запроса

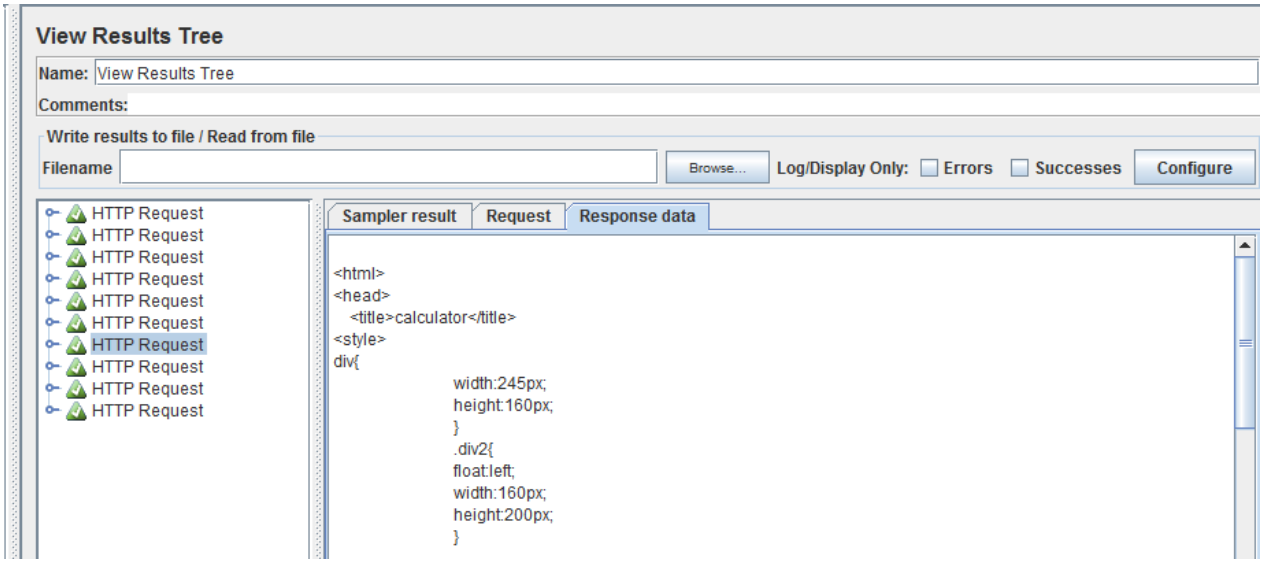

Рис. 6 – Результат GET запроса *(виден HTML-текст приложения)*

Тестирование масштабируемости (*Scalability Testing*) проверяет способность системы увеличивать свою производительность. Цель этих тестов – определить способность системы «расти» (*например, с большим количеством пользователей или с большим объемом данных*) без нарушения заданных для нее требований по производительности. Другими словами, это

исследование способности приложения *увеличивать показатели производительности в соответствии с увеличением количества доступных приложению ресурсов*.

Объемное тестирование (*Volume Testing*) – исследование производительности системы при обработке различных объемов данных.

Всплесковое тестирование (*Spike Testing*) – основывается на способности системы корректно отвечать при внезапных всплесках пиковых нагрузок и ее последующему возвращению к номинальному состоянию.

Тестирование на выносливость (*Endurance Testing*) фокусируется на стабильности системы. Этот тип тестирования проверяет наличие проблем с ресурсами (*например, нехватка памяти, соединение с базой данных, потоками*), которые влияют на производительность, или могут служить причинами сбоев в критических точках.

Тестирование надежности (*Reliability Testing*) – очень близкое по своему смыслу к предыдущему виду тестирования. Это проверка способности приложения выполнять свои функции в заданных условиях на протяжении заданного времени или заданного количества операций.

Конкурентное тестирование (*Concurrency Testing*) фокусируется на влиянии ситуации, при которой одновременно происходят множественные специфические действия (*например, одновременная работа большого количества пользователей*). При этом тестировании исследуется поведение системы в ходе обработке большого количества одновременно поступающих запросов, что вызывает конкуренцию между запросами за имеющиеся ресурсы (*базу данных, память, канал передачи данных, дисковую систему*). Иногда под конкурентным тестированием, также, понимают исследование работы многопоточного приложения и корректность синхронизации действий.

Тестирование мощности (*Capacity Testing*) определяет, как много пользователей может обслуживать система.

В целом при тестировании производительности следует различать статическое и динамическое тестирование. Статическое тестирование, зачастую, более важно для тестирования производительности, чем тестирование функциональной пригодности. Это является следствием того, что очень много критических дефектов вносятся в архитектуру и дизайн системы еще на этапе ее проектирования. Дефекты могут возникать из-за недопонимания или недостатка знаний дизайнеров и архитекторов. Кроме того, функциональные требования, цели использования, ожидаемая нагрузка и существующие ограничения могут быть недостаточно изучены по причине банального дефицита времени.

После запуска тест-плана, можно увидеть соответствующий отклик на запрос *(сообщение «200 OK» на Рис. 5)*.

3.2 Формирование API запроса на языке программирования Java

С целью формирования GET запроса на языке программирования Java, для работающего приложения, достаточно написать следующий код (*структура проекта приведена на рис. 7*):

```
public class Http {
    private static final Logger LOG = LogManager.getLogger(Http.class);
    public static void main(String[] args) throws IOException, SAXException,
ParserConfigurationException {
        String query = "http://localhost:8080/ServletClass";
        HttpURLConnection connection = null;
        try {
            connection = (HttpURLConnection) new URL(query).openConnection();
            connection.setRequestMethod("GET");
            connection.setUseCaches(false);
            connection.setConnectTimeout(2500);
            connection.setReadTimeout(2500);
            connection.connect();
            LOG.info(connection.getResponseMessage());
```

```
            LOG.info(connection.getResponseCode());
            StringBuilder sb = new StringBuilder();
            if (HttpURLConnection.HTTP_OK == connection.getResponseCode()) {
                BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                String line;
                while ((line = in.readLine()) != null) {
                    sb.append(line);
                    sb.append("\n");
                }
                String s = sb.toString();
                LOG.info(s);
            }
        } catch (ProtocolException e) {
            e.printStackTrace();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

В результате запуска программы в консоль выведется отклик «OK», код сообщения «200», а также HTML формат интерфейса калькулятора (*см. Рис. 8*).

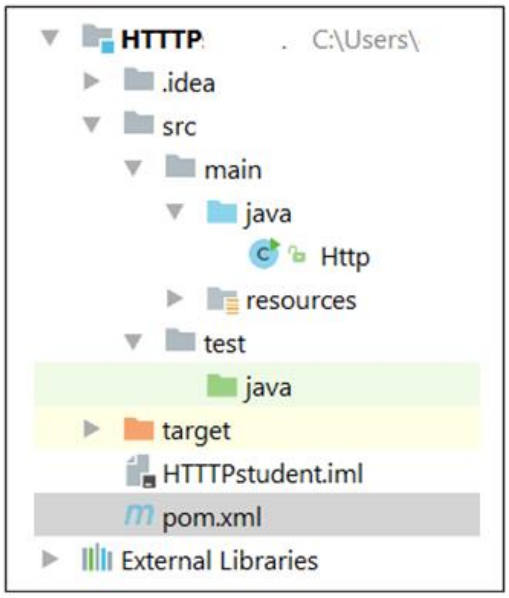

Рис. 7 – Структура проекта

```
"C:\Program Files\Java\jdk1.8.0_131\bin\java" ...
2020-04-19 12:51:21 [INFO ] Http:25 - OK
2020-04-19 12:51:21 [INFO ] Http:26 - 200
2020-04-19 12:51:21 [INFO ] Http:36 -
<html>
<head>
    <title>calculator</title>
<style>
div{
    width:245px;
    height:160px;
    }
    .div2{
    float:left;
    width:160px;
    height:200px;
    }
```

Рис. 8 – Результат написания GET запроса

4 Выводы

1. API тестирование обладает рядом преимуществ по сравнению с GUI тестированием. В работе рассмотрена возможность проведения данного способа тестирования с использованием двух различных подходов: – посредством применением Jmeter; – путем написания кода на языке программирования Java. Отмечено, что JMeter, также, находит свое применение для целей тестирования производительности приложений.

2. Акцентировано внимание на то, что тестирование производительности – сложный и ответственный этап разработки программного обеспечения.

3. Рассмотренная классификация видов тестовых испытаний является, фактически, полнофункциональной замкнутой системой для всесторонней оценки качества программных продуктов.

4. При динамическом системном тестировании приложений, тесты разрабатываются на основе функциональных требований исходя из уровня предполагаемой нагрузки на проектируемую систему.

5. Одним из наиболее эффективных инструментов, обеспечивающим возможность автоматизированной реализации количественной оценки показателей качества программного обеспечения, является приложение JMeter. Основные этапы использования данного инструмента можно рассматривать, как «дорожную карту» для соответствующих специалистов-тестировщиков.

6. Приведенная схема тестирования производительности веб-приложений, иллюстрирует возможность получения количественных оценок для последующего анализа производительности в условиях различных уровней нагрузки.

7. Удобство и практическая ценность JMeter заключается в высокой информативности и наглядности результатов проведенных испытаний (*таблицы и графики*), что создает хорошие условия для всестороннего анализа выполнения требований по производительности.

8. Следует обратить внимание на то, что этап составления тест-плана является особо ответственной процедурой.

9. Существенным преимуществом инструмента JMeter является его приспособленность для целей осуществления API тестирования. Подобное тестирование позволяет не только оценить комфортность перспективной эксплуатации будущего программного продукта, но и выявить возможные дефекты в логике выполнения его вычислительных алгоритмов.

10. Следует обратить внимание на тот факт, что аналогичные тесты можно проводить и с привлечением платформы Intellij Idea Java для написания модульных API тестов, что, на данный момент времени, является более востребованным направлением в тестировании.

Ссылки

[1] Тестирование программного обеспечения. Базовый курс/ Святослав Куликов. [Электронный ресурс] – Режим доступа: http://svyatoslav.biz/software_testing_book/ - 28.12.2017.
[2] Jmeter URL: https://jmeter.apache.org/ - 20.04.2020.
[3] Postman URL: https://www.postman.com/ - 20.04.2020.
[4] URL: https://metanit.com/java/javaee/4.1.php - 19.04.2020.
[5] Maven repository https://mvnrepository.com/ - 20.04.2020.
[6] URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 - 20.04.2020.

Рецензент: Георгій Кучук, д.т.н., проф., НТУ «ХПІ», Харків, Україна.
E-mail: kuchuk56@ukr.net

Надійшло: Січень 2020.

Автори:

Ольга Мелкозьорова, к.т.н., старший викладач кафедри, ХНУ імені В.Н. Каразіна, Харків, Україна.
E-mail: olja.mex@gmail.com

Валерія Гайкова, студентка факультету комп'ютерних наук, ХНУ імені В.Н. Каразіна, Харків, Україна.
E-mail: valeriagaikova98@gmail.com

Сергій Малахов, к.т.н., ст. наук. співробітник, доцент кафедри, ХНУ імені В.Н. Каразіна, Харків, Україна.
E-mail: mailgate@meta.ua

Опис схеми API тестування програмного забезпечення.

Анотація. Для оцінки якості додатків використовують API тестування. Існує велика кількість інструментів для його проведення (Postman, SoapUI, Jmeter і т.і.). Універсальний підхід до тестування ускладнюється великим обсягами даних, наявністю різноманітних методик та інструментів, також повинен бути об'єкт тестування. Складність логіки додатку, що працює, призводить до труднощів розуміння процесу тестування, тобто це стосується питання підготовки фахівців в цій галузі. Для спрощення розуміння в якості об'єкту для тестування було створено додаток «Калькулятор» за допомогою класу Servlet (Java). Тестування проводилося за допомогою інструменту JMeter та написання коду на мові Java. Складено тест-план з використанням інструменту JMeter, що дозволяє відправляти запити додатку. Відзначено, що автоматизовані модульні тести знаходять більше застосування при розробці програмного забезпечення, тому на мові програмування Java показана можливість написання аналогічного запиту.

Ключові слова: автоматизоване тестування; інструменти автоматизованого тестування; технології автоматизованого тестування, JMeter.

Reviewer: George Kuchuk, Doctor of Sciences (Eng.), Full Prof., NTU "KhPI", Kharkiv, Ukraine.

E-mail: kuchuk56@ukr.net

Received on January 2020.

Authors:
Olga Melkozerova, Ph.D., Senior Lecturer of the Department, V. N. Karazin Kharkiv National University, Ukraine.
E-mail: olja.mex@gmail.com
Valeria Gaykova, Computer Science Student, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine, Ukraine.
E-mail: valeriagaikova98@gmail.com
Serhii Malakhov, Ph.D., Senior Research, Associate Prof. of the Department, V. N. Karazin Kharkiv National University, Ukraine.
E-mail: mailgate@meta.ua

Circuit description of API interface for software testing.

Annotation. They use the testing API to evaluate the quality of applications. There are a large number of tools for its implementation (Postman, SoapUI, JMeter, etc.). The universal approach to testing is becomes complicated by the large amount of data, the availability of various techniques and tools, and there should also be a test object. The complexity of the application logic makes it difficult to understand the testing process; this concerns the training of specialists in this field. To simplify understanding, in the article, the Servlet (Java) Calculator application was used as a test object. Testing was done using the Jmeter tool and code writing in Java. A test plan was developed using a JMeter tool that allows you to submit application requests. It is noted that automated unit tests are more useful in software development, so the Java programming language shows the ability to write query algorithms.

Key words: Automated testing; Automated testing tools; Automated testing technologies; JMeter.