Тестирование в жизни программиста

JSP & Servlets
3 уровень, 0 лекция

ОТКРЫТА

Зачем нужно тестирование программистам

Ближайшие пару уровней будут посвящены тестированию в том ключе, в котором оно нужно **именно программистам**. Но сначала давай выясним, что такое тестирование и зачем оно нужно.

Применительно к ПО можно сказать, что задача тестирования проверить, что программа:

- делает то, что она должна делать
- не делает того, что она делать не должна

Второй пункт, кстати, не менее важный, чем первый, но об этом немного позже.

Давай начнем с первого пункта. Что значит “программа делает то, что она должна делать”?

Во-первых, кто-то должен составить список всех сценариев использования программы. Во-вторых, в них нужно описать, как должна происходить работа программы, как должен вести себя пользователь, и какие ожидаются результаты. Дальше можно не продолжать.

Как только мы написали “как должен вести себя пользователь”, вся идея написания хорошей документации рассыпалась. Люди не машины, более того, люди очень часто ведут себя с софтом как им угодно. Никто не начинает знакомство с техникой с изучения инструкции. Это факт.

Поэтому мы получаем новый факт: особенность работы программного обеспечения состоит в том, что у него очень много различных сценариев работы. Некоторые из них очевидны, другие можно задокументировать, третьи можно предположить, о четвертых можно догадаться, ну а остальные 50% вам даже в голову не придут.

С точки зрения программиста большая часть ошибок — это и не ошибки вовсе. Ошибка — это когда программа работает не так, как должна или как ожидалось. А есть куча ситуаций, когда непонятно, как программа должна работать, или сценариев, которые друг другу противоречат…

Сценариев бесконечно много, и в продукте всегда будут случаи, когда программа ведет себя не так, как ожидалось (программист-то написал код всего для пары десятков сценариев). Поэтому можно утверждать, что **в любой программе всегда есть баги** и **любой продукт можно улучшать бесконечно**.

Дальше все упирается в целесообразность. Сначала программист исправляет самые крупные баги, затем баги поменьше — и так далее. И, наконец, настает этап, когда владелец продукта считает, что дальше работать над ним экономически нецелесообразно.

Но вернемся к ошибкам, которые все признают ошибками: программа явно делает что-то не то, упала, сломала что-то и т. д. Такие ошибки условно можно разделить на 3 категории: большие, средние и маленькие.

И очень часто случается, что программист работает над исправлением средних или даже маленьких ошибок, хотя в проекте есть еще куча более серьезных проблем. **Он просто их не нашел**, поэтому и работает над самыми большими, о которых знает.

Поэтому в любом проекте должны быть тестировщики. Эти люди специально учатся смотреть на продукт под разными углами. Так можно заметить больше сценариев работы программы. Их задача — находить ошибки и записывать их (чтобы не находить одну и ту же ошибку по нескольку раз).

Тестирование — это процесс, направленный на поиск ошибок. Эти ошибки должны быть найдены, описаны и приоритезированы. Только после приоритизации ошибок можно говорить об эффективном процессе улучшения ПО.

Не забывай, **первый шаг к решению проблемы — это признать наличие проблемы**. Нельзя исправить ошибку, о которой не знаешь.

Автоматизация тестирования

Думаю, мы все согласились с тем, что тестирование — это важно, так что давай посмотрим на тестирование как программисты. А как на тестирование смотрят программисты? Программисты автоматизируют работу других людей. Последняя профессия, которая исчезнет, будет профессия программиста.

Мы автоматизируем любые процессы, с которыми сталкиваемся. Значит, нужно автоматизировать и тестирование. А как автоматизировать поиск ошибок? Краткий ответ: никак. Но тут нам на помощь опять приходит то, что мы программисты.

Процесс разработки программного обеспечения состоит из постоянного внесения в него изменений. как раз в процессе постоянного внесения изменений программисты очень часто ломают то, что до недавнего времени отлично работало.

И тестировщики вместо того, чтобы искать новые ошибки, вынуждены постоянно проверять, на поломали ли мы с вами то, что давно и хорошо работало. Так называемое регрессионное тестирование. Именно этот тип тестирования автоматизировать можно и нужно.

Тут весь софт можно поделить на две части:

- программа взаимодействует с человеком
- программа взаимодействует с другой программой

Первый вариант автоматизируется сложнее, для него нужны специальные тестировщики-автоматизаторы, их еще называют QA Automation или Software Test Engineer.

А вот второй вариант автоматизировать можно и нужно самостоятельно. Если у тебя есть часть программы, которая:

- хорошо работает
- уже протестирована
- выполнена в виде отдельного модуля или логического блока
- не планирует меняться
- от нее зависят другие модули или программы
- поломка функционала дорого обойдется

Рекомендую выделить время и написать для нее тесты, которые зафиксируют ключевые аспекты ее текущей функциональности. На это будет **разумно выделить 5% твоего рабочего времени**, или 1 день в месяц.

Не нужно писать тесты ради тестов.

Никто не будет поддерживать твои тесты. Ни другие программисты, ни ты сам. Так никто не делает. 99% всех написанных тестов заброшены и/или отключены. Если можешь не писать тесты — не пиши. Пиши только если без них ну точно не обойтись.

Типы тестирования

Каждый программист, если он не проходил специальное обучение, сможет рассказать своими словами, что такое тестирование: проверка, делает ли программа то, что должна. Однако профессионалы в этой области выделяют целые направления (типы) тестирования.

Все тестирование, действительно, крутится вокруг надежности и готовности программ, однако чтобы лучше понять направления тестирования, давай рассмотрим несколько примеров.

Допустим, ты тестируешь типичный интернет-магазин. Тогда направления тестирования можно разбить на такие типы: тестирование производительности, функциональное тестирование, интеграционное тестирование и модульное тестирование.

Если владелец сайта решит запустить серьезную рекламную компанию, то на сайт одновременно придет куча пользователей. Вполне может быть, что сайт не упадет, но некоторые его разделы могут работать медленно или вообще перестать работать.

Для того, чтобы этого не случилось, тебе нужно заранее выявить такие проблемы и предпринять шаги для их устранения. Это делается с помощью **нагрузочного тестирования**, или его еще называют тестированием производительности.

Так же ты можешь захотеть проверить, как работает API твоего бэкенда и протестировать каждую его функцию: регистрацию, логин, добавление в корзину, обработку платежей, записи в базу данных и т. д. Все должно работать согласно ТЗ. В этом случае нужно выполнить **функциональное тестирование**.

Твой интернет-магазин скорее всего интегрирован со сторонними сервисами: рассылка писем и СМС, платежные системы, онлайн-чаты поддержки, сбор обратной связи от пользователей, рекламные системы и т. д. Чтобы убедиться, что все это работает как задумано, тебе нужно провести **интеграционное тестирование**.

И, наконец, сложные продукты часто разбивают на независимые модули. Из таких модулей можно собрать финальный продукт, как из конструктора. Если ты занимаешься разработкой такого модуля или взаимодействием таких модулей, то тебе понадобиться провести **модульное тестирование**.

Подводя итоги, можно сказать, что функциональное тестирование нужно для проверки каждой отдельной функции сайта. Интеграционное — для тестирования взаимодействия крупных модулей и систем вашего продукта. Модульное — для проверки отдельного модуля, ну а тестирование производительности — для проверки работы твоего сайта под нагрузкой.

Типов тестирования может быть еще больше: чем сложнее продукт, тем больше его аспектов разработки нуждаются в контроле.

‹ Предыдущая лекция

Следующая лекция ›

Ответить

0

Pavel Shnurov Уровень 20, Russian Federation

16 июля, 07:12

"продукты часто разбивают НЕ независимые модули"
Возможно тут тоже опечатка?

Ответить

0

Mentor-02 Backend Developer в **JavaRush** MENTOR

31 октября, 18:44

поправили

Ответить

0

Facepalm Уровень 32, Москвачкала, Россия

11 июня, 15:27

Ошибочка - Люди очень часто ведут себя с софтом КАКИМ (*КАК ИМ) угодно

Ответить

+2

Mentor-02 Backend Developer в **JavaRush** MENTOR

31 октября, 18:43

поправили

Ответить

0

Jgu4iPer4ik Уровень 32, Санкт-Петербург, Russian Federation

27 мая, 17:06

11010000 10011111 11010000 10110101 11010001 10000000 11010000 10110010 11010001 10001011 11010000 10111001 00100000 11010000 10111101 11010000 10110000 11010001 10000101 00100001

Ответить

+1

Андрей Пазюк Уровень 70, Винница, Украина

14 июля, 21:32

Jgu4iPer4ik, 11010000 10010001 11010000 10111110 11010001 10000010 11010000 10110000 11010000 10111101 00101001

Ответить

+2

GvardeeZZZ Уровень 41, Москва, Россия

15 июля, 19:31

Меня поглощает матрица11100000 00111100

Ответить

0

Anonymous #2656537 Уровень 41

17 июля, 17:52

11010000 10100111 11010000 10110101 11010000 10111011 100000 11010000 10111101 11010000 10110101 100000 11010001 10001000 11010000 10110000 11010001 10000000 11010000 10111000 11010001 10000010 101100 100000 11010000 10111101 11010000 10111110 100000 11010000 10111111 11010001 10001011 11010001 10000010 11010000 10110000 11010000 10110101 11010001 10000010 11010001 10000001 11010001 10001111 100000 11010001 10001000 11010001 10000011 11010001 10000010 11010000 10111000 11010001 10000010 11010001 10001100

Ответить

0

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

ДОСТУПНО В Google Play

Загрузите в App Store