QA Hiring - Technical Exercises

Introduction

QA Engineers at Spartez are our (not so) secret weapon. They not only help teams continually improve the quality of the software they create, but they also help teams do it more effectively. In short, our QA engineers are essential for shipping quality software faster.

Our developers know this. QA is routinely voted the most satisfying aspect of how we develop software. As such, developers constantly pull QA engineers into collaborating on their work. This is not a company where code is thrown over the wall to QA, nor a place for people who don’t want to be deeply involved in the development process.

To be effective, our QA engineers need strong testing, technical and collaboration skills. The aim of this set of exercises is to allow you to demonstrate some of those.

To complete, please:

1. Create a **private** repository on Bitbucket, Atlassian’s code hosting service: https://bitbucket.org/
2. Upload your files to that repository.
3. Under Settings->Access Management for the repository, grant Admin access to the “atlassian-qa” user.
4. Let us know the link to your repository via email.

Exercise 1 - Test Automation

Goal

To demonstrate your ability to design and implement valuable automated tests.

Task

Write two automated browser-based tests for Confluence Cloud, our hosted wiki product:

- One that tests that a user can create a new page.
- One that tests that a user can set restrictions on an existing page.

Details

- Sign up for a free trial of Confluence Cloud at https://www.atlassian.com/software/confluence/try/ . This will provide you with 7 days of free access to a hosted Confluence instance. No credit card is needed.
- Use Selenium 2.0 (aka WebDriver): http://docs.seleniumhq.org/projects/webdriver/
- Implement the page objects pattern: http://code.google.com/p/selenium/wiki/PageObjects - or tell us why your alternative approach is better.
- Write your test in any language WebDriver supports.

Submitting the Results

Once you've written your tests, run them successfully and are happy with the result:

- Push your tests to your Bitbucket repository.
- Document any test pre-requisites, assumptions made and any issues found in the repo's ReadMe file.

Exercise 2 - Exploratory Testing

Goal

To demonstrate your approach to manually testing a feature without using specifications or scripted test cases.

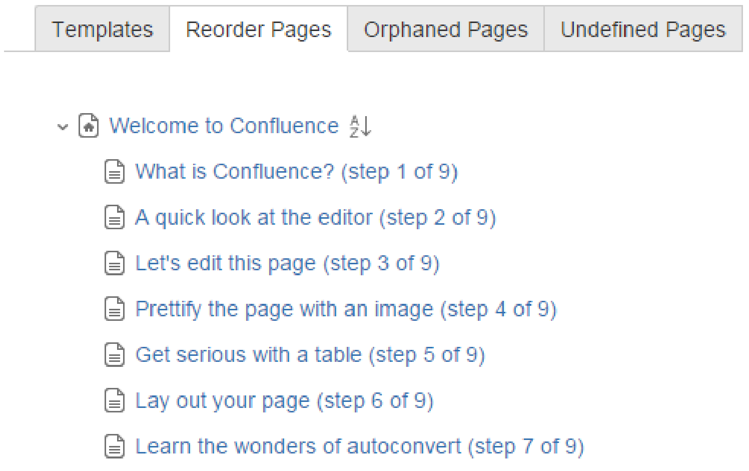
Task

Perform 30 minutes of exploratory testing of the Confluence Page Tree feature to investigate if this feature is of high quality.

Decide what further testing of the feature would need to be performed before you would ship it to production.

Details

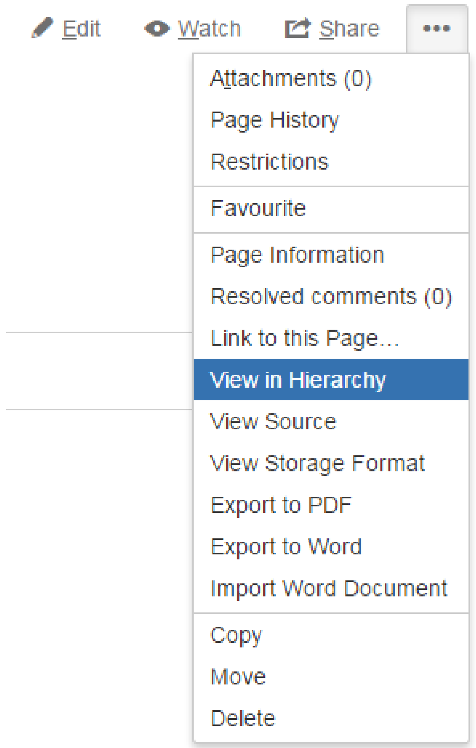
- Use the same Confluence instance that you created for the previous exercise.
- You will be testing the Page Tree. This feature displays all the pages in a space in an expandable and collapsible tree. It allows the user to rearrange these pages by drag-dropping.



Welcome to Confluence
- What is Confluence? (step 1 of 9)
- A quick look at the editor (step 2 of 9)
- Let's edit this page (step 3 of 9)
- Prettify the page with an image (step 4 of 9)
- Get serious with a table (step 5 of 9)
- Lay out your page (step 6 of 9)
- Learn the wonders of autoconvert (step 7 of 9)

An example page tree - your data may be different

- This feature can be accessed by clicking “View in Hierarchy” from the “...” menu in the top-right corner on any page in a space.



How to access the Page Tree from any Confluence page

- The scope of this exercise is limited to the tree view feature only, under the “Reorder Pages” tab. You should not test the other tabs (Orphaned Pages, Undefined Pages, etc) or other Confluence features, unless they will help you to assess the quality of this feature.
- You should explore to discover other functionality offered by this feature.
- As this exercise is aimed at assessing your exploratory testing approach and not the total number of bugs found, there is little value in continuing beyond the allocated 30 minutes.

Submitting the Results

After completing your exploratory test session, create a file containing:

- A short description (1-2 sentences) of the approach you took to your testing.
- A bulleted list of the scenarios you tried, even if they were successful.
  - E.g "Reordered pages under the same parent. Dragged pages between parents."
- A quick description of any bugs you found. It's OK to not have found any – we're more interested in your test ideas and approach than what you executed in the 30 minutes.
- A list of further testing areas that you would want completed before you would be comfortable shipping this feature to 40,000 production instances. Consider the risks of the feature as you understand it.

Once completed,

- Push the file to your Bitbucket repository.