In-Class Activity 5: Formal Specification Based Testing

Overview

- Create tests based on (semi) formal specification of user behavior
- Use popular testing tool, Cucumber (http://cukes.info)
- Use extension for web testing, Capybara

Goals

1. All goals from ICA-1
2. All goals from ICA-3
3. Know how to run and write tests using Cucumber and Capybara
4. Learn test code reuse
5. Use test coding best practices

“Top 5 Cucumber best practices”

1. Write declarative features:

Scenarios should be written like a user would describe them. Beware of scenarios that only describe clicking links and filling in form fields, or of steps that contain code or CSS selectors. This is just another variant of programming, but certainly not a feature description.

“Top 5 Cucumber best practices”

2. Insert a narrative

Narratives describe in about one sentence what a feature does. Typical narratives contain a benefit for the user, a role that needs the feature and the feature itself. Narratives are important to envision **why you are implementing a feature** in the first place. They also give a short overview of the feature so others get a rough understanding what it is about without reading the scenarios.

“Top 5 Cucumber best practices”

3. Avoid conjunctive steps

When you encounter a Cucumber step that contains two actions conjuncted with an “and”, you should probably break it into two steps. Sticking to **one action per step makes your steps more modular and increases reusability.** This is not a general rule though. There may be reasons for conjunctive steps. However, most of the time it’s best to avoid them.

“Top 5 Cucumber best practices”

4. Reuse step definitions

In Cucumber you can reuse steps in other steps. This comes in handy when a step extends another step’s behavior or defines a superior behavior that consists of multiple steps. You should try to **reuse steps as often as possible.** This will improve the maintainability of your app: If you need to change a certain behavior, you just need to change a single step definition.

“Top 5 Cucumber best practices”

5. Use backgrounds wisely

If you use the same steps at the beginning of all scenarios of a feature, **put them into the feature’s Background.** Background steps are run before each scenario. But take care that you don’t put too many steps in there as your scenarios may become hard to understand.

Helpful Tips

1. Install via synaptic:
   1. Ruby
   2. Ruby-dev
2. Use “gem” to install:
   1. cucumber
   2. capybara
3. Learn to parameterize your step definitions
4. Bash has a for loop!
5. Think of features as use cases

Blank

Happy Dasara!

Implement the Following

For each scenario, create/run a feature that verifies the correct behavior. Show the coverage report.

1. Create a new user and be able to log in with that account
2. Add two items to the shopping cart and verify they are present in the cart
3. Delete item from shopping cart and verify it is no longer present in the cart
4. A search for a term returns a correct result
5. The guest user is not able to access the admin menu

Grading

100 total points

- 15 points for the quiz
- 20 points for in-class checkoff (late credit until 10/21)
- 15 points for complete automation
- 50 points for the five scenarios

Other notes:

- Rerun ICA 3 – coverage should be the same
- Reuse test steps where possible
- Test features should be independent

Submit

1. Shell script
2. Feature and step definitions
3. Coverage reports for ICA 5