BREDEX
Software-Entwicklung und Beratung

Automated functional testing with keywords

Hans-Joachim Brede, BREDEX GmbH

© 2008 Bredex GmbH Version 2.3

BREDEX
Software-Entwicklung und Beratung

Agenda

- The aim & challenges of functional testing
- Keyword-driven testing → the answer?
- Examples

BREDEX
Software-Entwicklung und Beratung

Why do we need functional tests?

- **JUnit Tests**

  Written by developers

  Don‘t test requirements

- **Tests through the API**

  Don‘t test that GUI and workflows work as specified

- **All green for JUnit & API tests**

  Doesn‘t mean that the application is correct

BREDEX Software-Entwicklung und Beratung

The aims of functional GUI testing

- **Check requirements as the user will see them**

  … software is designed for the user!

- **Complete workflows via the GUI**

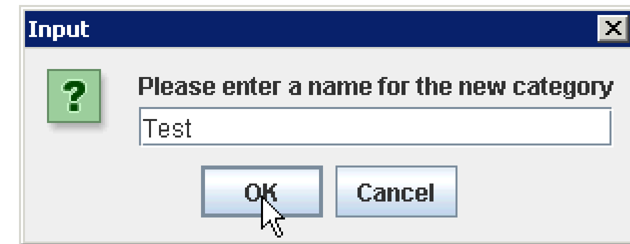
  Does the *happy path* work?

  Do different ways of working give the same results?

  Is the application consistent for the user?

  Are false inputs correctly handled?

  Can the user cause the application to crash?



BREDEX
Software-Entwicklung und Beratung

The challenges of automation

- **Tests have to be in-step with development**

  Created close / in parallel to development

  Can run as soon as the feature is delivered

  → prompt feedback on quality and acceptance

  → early error recognition (lower costs to fix)

Software-Entwicklung und Beratung

The challenges of automation

- **Maintenance**

  Workflows, GUI, requirements change frequently

  Tests have to keep running despite changes

- **Test creation**

  Support test **planning** and test **design**

  Quick and easy

  Black-box (from user's perspective, not developer's)

- **Readable**

  For business users – can check tests against requirements

Why not record my tests?

- **Delay: have to wait until the application is ready**
- **Can only test what already works**

  And tests the implementation, **not** the requirements
- **Inflexible and bound to unnecessary details**
- **Redundancy in the test**

  Similar actions recorded twice and more
- **Need to spend time programming to ensure maintainability**

BREDEX
Software-Entwicklung und Beratung

Keyword-Driven Testing

- **Based on the same principles as development**
  - Do it once and only once!
  - Modularity from the outset
  - Reusing modules makes tests easier to maintain
- **GUI tests are made up of recurring actions**
  - Lend themselves well to being tested with keywords
  - Each keyword executes a certain action / actions
  - → library of keywords
  - More complex keywords made by combining other keywords

Software-Entwicklung und Beratung

What‘s behind the keyword?

- **In some frameworks (e.g. FIT, Quality Center)**
  Code written by automation experts in the team
- **In other tools (e.g GUI*dancer*)**
  Library of basic keywords is present in the tool
  Tests aren‘t written in program code

Benefits for the development process

- **Tests are readable**
  Can be checked against the requirements
- **Prompt feedback**
  Keyword creation can happen before code is written
  Tests run on regular builds → errors found early
- **Maintenance reduced**
  Reused modules → central changes update the whole test
  No programming work to maintain tests
- **Modularity**
  Easy to change, add and delete modules in the test

What to bear in mind with keywords

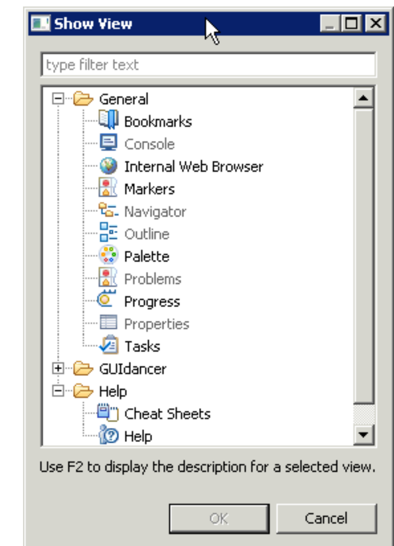
- **Test design is important**
  - What is reused
  - Flexible modules
- **Finding keywords**
  - Library must be well structured
  - Naming conventions help test team to write tests
- **Keyword explosion?**
  - Think carefully about when a keyword is necessary

Eclipse examples

Show view

Show View [param: path-to-view]

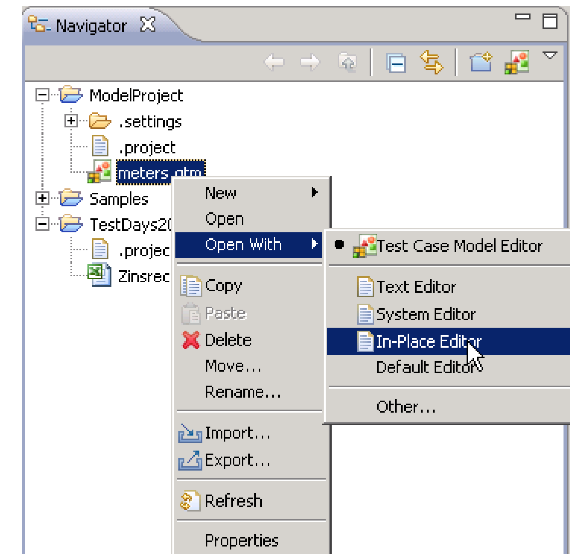
Component	Action	Data
Menu	Select item from menu	Window/Show View/Other...
-	Wait for window	Show View
Tree	Select item from tree	<path-to-view>
Ok Button	Check enabled	True
Ok Button	Click	Once



More Eclipse examples

Open Editor

Open Editor for Any Item in Any View [param: view, path-to-item, context-menu-path]		
Component	**Action**	**Data**
Reuse Keyword: Show View		<view>
<Tree>	Check existence of item	<path-to-item>
<Tree>	Select item from tree	<path-to-item>
<Tree>	Select from context menu	<context-menu-path>



BREDEX
Software-Entwicklung und Beratung

Summary / results

- **10% of project cost for test automation**
- **10% the above for maintenance**
- **Exponential growth of test coverage through reuse**

… now I understand why you don't need capture-replay