IBM Innovate 2010
QM-1510A

Michael Freeman
Applications Development Analyst,
Lender Processing Services
mlfreeman@gmail.com

Latisha Aguilar
Applications Development Manager,
Lender Processing Services
lzaguilar@gmail.com

Innovate2010
The Rational Software Conference

Let's build a smarter planet.

The premiere software and product delivery event.
June 6–10 Orlando, Florida

Agenda

- Why are GUI testing applications unstable?
- How can we minimize the impact of these issues?
- Eclipse Tips (and Demo)
- Helper Code Demo

- Feel free to ask questions at any point.

Why are GUI testing applications unstable?

- Timing differences from run to run
- New version changes
- General Windows insanity

Timing Differences From Run To Run

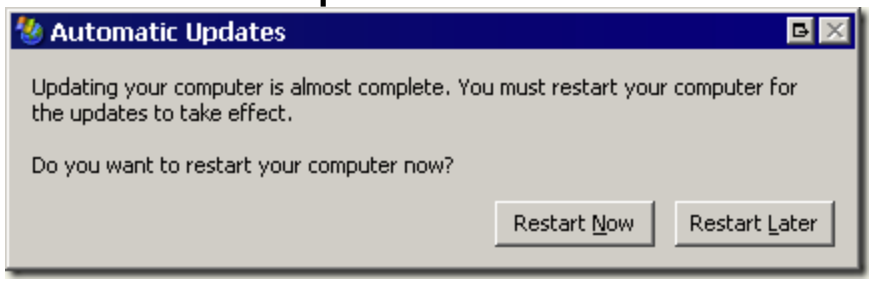
- CPU speed differences on RFT boxes
- Available RAM differences on RFT boxes
- Other background apps running on RFT boxes
  - e.g. AV on one box vs no AV on another
  - e.g Terminal Services sessions (on server Windows systems)
  - e.g. Eclipse open vs RFT from command line
- Network issues
  - Application server CPU load
  - Network traffic
  - Webapps: JavaScript processing time

New Version Changes

- UI Changes, both visible and invisible
  - Visible: new buttons / fields / etc
  - Invisible: developer refactoring causes existing objects to change properties
- Business Logic changes
  - New application behavior
- Bugs
  - e.g. Regression in webapp leads to browser JavaScript error alert

General Windows Insanity

- Dialogs that may not have been configured to not appear
  - Automatic Update reminders



  - Browser dialogs (e.g. AutoComplete, et al.)

General Windows Insanity

- GUI events failing to occur for no apparent reason
  - Clicks not registering
  - Characters getting dropped for no reason when RFT tries to type something

How can we minimize the impact of these issues?

- 6 basic rules we follow at LPS to make our RFT scripts as stable as possible
- One overall idea: Our RFT scripts are development projects.

Rule #1: Work surgically

- Grab your TestObjects at the last possible second before you need them
- Work on them as quickly as possible.
- Release them as soon as you can.

TestObject Grabbing

- Don’t map…find() instead.
  - Maps store unnecessary information and depend on information that isn’t really beneficial.
  - Maps can’t be merged easily. They are complex XML documents.
  - When you find(), try not to find() from RootTestObject.
    - It’s slow, relatively…but it is faster than it used to be.
    - If you must, put .class as the first property in your subitem.
      - We found that made a 75% speed up in find timings.

Acting Quickly

- Do all possible computations before find().
  - Compute things to type in, look up data, whatever.
    - This ensures you're ready to go once objects are found.
- Once you have the object, you should work on it and release it before trying to do anything else or work with any other objects.

Cleanup

- Unregister everything. Manage your objects.
- If you wrap your code in try-catch blocks, put your unregister() calls inside finally blocks so no matter what they will run.
  - Example:

```
TestObject[] objects = null;
try {
        objects = someObject.find(atDescendant("prop1", "val1", "prop2", "val2"));
        //do something
} catch (Exception e) {
        //log an error
} finally {
        unregister(objects); // ensure objects will always get unregistered.
}
```

  - RationalTestScript.unregister() doesn't care if it gets passed a *null* reference. It won't throw a NullPointerException.

Rule #2: Wrap up everything inside functions

- Ensures consistency, reuse, and clarity of driver script.
- Consistency:
  - Calling one function guarantees same actions always happen and same things always get logged.
- Reuse:
  - You're only writing a given action / group of actions once.
- Clarity:
  - login(username, pass); is easier to read than
    userName.setText(username);
    password.setText(password);
    loginButton.click();

Rule #2: Wrap up everything inside functions

- Exception handling is easy now.
  - Try-catch blocks are all relatively small and specific
    - Error messages become useful.
- You can leverage do-while loops to help ensure actions happen (call the function repeatedly until something changes)
  - Works around the general Windows insanity mentioned earlier
- Consider making generic wrappers for generic FT actions (e.g. public static int click(GuiTestObject gto, Object…args) wraparound for GuiTestObject.click())
  - If you develop and use these you can have consistent exception handling and logging for every widget you encounter everywhere

Rule #2: Wrap up everything inside functions

- Sometimes you have to wrap up actions in multiple layers.
  - Single action wrappers around actual FT actions
  - Widget-specific wrappers around single action wrappers
  - Screen wrappers around commonly used groups of widget wrappers
  - App-wide wrappers around common screen wrappers

Rule #3: Use regular expressions when possible

- This gives you leeway if things change slightly (e.g. layout shifts but functionality is the same)
- Many web app frameworks have patterns to their UIs
  - You can exploit this to make operations on multiple objects easier

Rule #4:
Pick only a minimal set of properties when defining your objects

- Some properties are less likely to change than others
- .class will always be there in every domain, every object
- Don’t be afraid to go to your developers and insist they add properties to help you. If they do this, they will probably end up helping themselves out also in the end.

HTML domain suggestions for properties to use in find()

1. .class
2. .id
3. .name
4. .className (sometimes)
5. .value (rarely)
6. .text (if all else fails)

- If you're really lucky on a HTML domain app every important field will have a <label for="field id">field name</label> element near it.

Rule 5: Centralize data so changes only have to be done once.

- For webapps, don't code URLs for environments directly in to your code.
- Create a single central Java enum (or at worst a Hashtable) mapping Environment Names to URLs…then address everything by that env name.

Rule 5: Centralize data so changes only have to be done once.

- Example:

```
public enum Environments {
        Dev("http://dev"), QA("http://qa"), Production("http://prod.com");
        private Environments(String URL) {
                this.URL = URL;
        }
        private String URL = "";
        public String getURL() {
                return this.URL;
        }
}
public void testMain(Object[] args) {
        Environments env = Environments.valueOf(args[0].toString());
        RationalTestScript.startBrowser(env.getURL());
}
```

Rule 5: Centralize data so changes only have to be done once.

- The script on the previous slide is now 100% immune to changes in URLs of app environments
- This is superior to RFT's startApp() function because you can leverage the Environments enum in other ways
  - Enums are a standard Java structure
  - startApp depends on a config file that isn't inside your project. This makes moving code to another machine a chore.

Rule 6: Databases are your friend

- JDBC == Good
  - Enables much more control over the testing process
    - Decrease need for data pools
      - Why have a static data pool that has to be updated when you can just look stuff up?
    - Allows you to check data at multiple points along the testing process
      - A nice value add to many projects
    - More stable than the recommended approach of automating a database client app GUI

Breakpoints

- Eclipse offers nice breakpoint functionality
  - On Class Load
  - On Method Load
  - On Specific Exception Being Thrown
  - On Any Uncaught Exception Being Thrown
  - On Any Caught Exception Being Thrown
  - Conditional Breakpoints
    - Break only when a code snippet is true
    - Break only when the result of a code snippet changes
  - Unconditional (aka normal) breakpoints
- Breakpoints are your friend

Detail Formatters

- Detail Formatters
  - When in a debug view, Eclipse will let you see all the relevant variables at that spot in execution. It normally calls toString() to render their contents when you click on one. RFT TestObjects have a toString() method that outputs information that is useless to probably everyone except the original developers. Eclipse will let you supply a blob of code to run the object through as a temporary replacement toString().

Refactor Menu

- Refactor menu will save you tons of time in most code reorganizations.
- It can move and rename classes, methods, and variables and automatically update all impacted code for you.
- You can add or remove arguments from methods with it and automatically update all impacted code with the new method signature.
- It even works with source control providers like Clearcase or Subversion so all you have to do is use Refactor -> Something and then Commit/Check In and your changes are in and still compiling…all with no more than a few mouse clicks.

Code Demo

- We have developed a large library of code to smooth over a lot of the quirks in automation.
- Now I’m going to demonstrate some of it.

Eclipse Breakpoints 101
And
Useful RFT Helper Routines

IWindow Code

- From IBM's docs: "Provides access to the native window management system; any use of this interface is likely to be platform specific."
- It has its uses.
  - It is extremely fast at finding things on screen.
  - It works in places normal TestObjects don't.
  - It enables reuse that mapped TestObjects don't.

IWindow Code Demo

- This script will open a browser to Google News and open a print dialog.
- Controlling the print dialog through IWindow objects is far faster than controlling it through TestObjects (if you even can)
- Attempts to dynamic find() the print dialog always failed on numerous test machines.
- Recording the dialog did work, but it produced something specific to the AUT. The print dialog (and others) are actually part of Windows Common Controls, so having one library to handle it no matter where it appears is ideal.

Playback Monitor Demo

- This next demo shows basic control over the RFT playback monitor.
- We created this class because we weren't satisfied with the amount of information that RFT normally shows to us.
- We can display any custom message we want.

Object Map Demo

- This next demo shows how to manipulate an RFT object map file in code.
- This bypasses the usual RFT approach of adding poorly named protected functions to the script helper (e.g. html_subform_ctrl())
- These functions can load an object from a map by either its #id or its #name property.

General Script Demo

- This final script is a simple generic demo of how we tend to organize things.
- The driver script calls upon various classes. Typically each class is one screen or page.


Questions

Daily iPod Touch giveaway

SPONSORED BY

AllianceTech
Intelligent EVENTS

- Complete your session surveys online each day at a conference kiosk or on your Innovate 2010 Portal!
- Each day that you complete all of that day's session surveys, your name will be entered to win the daily IPOD touch!
- On Wednesday be sure to complete your full conference evaluation to receive your free conference t-shirt!

www.ibm.com/software/rational