Test-Driven Development (TDD) for Eclipse RCP

Red-Green-Refactor!

5/27/2009
Kevin P. Taylor
Kevin P. Taylor

- President and Founder, Obtiva
- Blog: http://ktaylor.name
- Former Editor: http://java.about.com
- Committer on Eclipse’s Glimmer project
Test-Driven Development is a tool for building quality software.
We are all professional coders on a journey.

We’re all looking to add just one more tool to our tool belt
But, danger lurks in the Eclipse Platform for TDDers.
It just seems too hard.
We’ll learn how to leverage existing tools and techniques to make TDD possible in Eclipse.

Easy, maybe?
How are We Doing?

• Who uses JUnit consistently?
  - For Eclipse RCP applications?
• Who writes their tests first?
• Who writes their tests after?
  - Who finds time?
Our plan for today.

- What is test-driven development?
- The tools of the trade
TDD is writing your tests first.

It is a discipline

//obtiva
TDD has a heartbeat.

- **Red** - Write a test that fails
- **Green** - Make it pass
- **Refactor** - Remove duplication
  (rinse. repeat.)
There are 3 benefits of doing TDD.

- API design
- Coding confidence
- DRY code (Don’t Repeat Yourself)
Eclipse has put some hurdles in place.

- Separation of concerns
- OSGi wires bundles at runtime
- SWT Widgets are difficult to mock
Here are our tools of choice.

- JUnit and JMock
- PDE JUnit
- Model-View-Presenter (MVP) pattern
JUnit and JMock

- Use for classes that are not dependent on Eclipse Platform API
- Particularly useful for testing business logic in model classes
PDE JUnit (Plug-in JUnit)

- Executed by a special test runner that launches another Eclipse instance in a separate VM
- Your tests can call the Eclipse Platform API, along with methods from your own plug-in
Enable testing by creating layers.

- MVC
- MVP
Model-View-Controller (MVC)
Model-View-Presenter (MVP)
MVP will make your life easier.

- Separate concerns
- Minimize untested GUI code
Don’t forget some functional testing!

- TDD is not a silver bullet
- Functional tests exercise code end-to-end:
  Presenter <-> Model <-> DB
Functional and GUI testing tools are available.

- SWTBot
- TPTP
- IBM Rational Tester
Exercise:

1. Whiteboard a GUI design
2. Use PDE JUnit to test drive a GUI ViewPart with a ListViewer and a toolbar “refresh” button
3. Use JUnit to test-drive a Presenter, Model, and cmdline (stub) View
4. Wire up the GUI and Presenter with PDE JUnit tests
Conclusion

• TDD in Eclipse RCP takes extra discipline
• Red, Green, Refactor
• Leverage JUnit, PDE JUnit, and MVP
• Throw in some functional tests
Questions? Want source code?
Email me.
References

- SWTBot – http://www.eclipse.org/swtbot/