Debugging Java
With Eclipse

Killing Bugs in your Java Code with the
Eclipse Debugger
Agenda

- Debugging
- Debugging Perspective
- Debugging Java Applications
- Debugging Remotely
Debugging

- **Jargon** - shotgun debugging n. the making of relatively undirected changes to software in the hope that a bug will be perturbed out of existence. This almost never works, and usually introduces more bugs.
Debugging

- **Reference.com** - debugging: To search for and eliminate malfunctioning elements or errors in:
  - debug a spacecraft before launch;
  - debug a computer program.
Debugging

- Wikipedia - Debugging is a *methodical* process of *finding* and *reducing* the number of bugs, or defects, in a computer program or a piece of electronic hardware thus making it *behave as expected*. Debugging tends to be harder when various subsystems are tightly coupled, as changes in one may cause bugs to emerge in another.
Good Old Days

- GDB - C/C++/Objective-C
  - other GNU languages as well
  - Command line - but very powerful
  - Cool theme song :-)}
Bad Old Days

- JDB
  - painful to use, slow, not very feature rich
- Logging was easier
  - Thus everyone went to logging
Agenda

- Debugging
- Debugging Perspective
- Debugging Java Applications
- Debugging Remotely
Debugging Perspective
Debugging View

Controlling the VM

- View threads
- Start/Stop/Pause the program
- Step through your program
Variables View

See the state of your app

- View any value in your application
Breakpoints View
Manipulate Breakpoints

- Edit Breakpoints
Breakpoint Properties
Expressions View
Examine Program State

• View Variables

```
"user"= SiteUserHibernate (id=34)
  id= null
  indexes= ArrayList (id=42)
  passwd= "insert-test"
  portfolios= ArrayList (id=54)
  uname= "insert-test"
  version= Integer (id=55)

net.dudney.domain.hibernate.SiteUserHibernate@a666bf
```
Display View
Manipulate Program State

- Manipulate Variables

```
user.getUname()
(java.lang.String) insert-test
```
Agenda

- Debugging
- Debugging Perspective
- Debugging Java Applications
- Debugging Remotely
Methodical

- Test Driven Development
- Unit Testing
- Developer Testing
- Good Requirements Definition
  - Use Cases lead to Test Cases
Setting Breakpoints

- Line
- Method
- Class load
- Exception
- Watch Point
Setting Line Breakpoints

- Double Click in the margin

```java
/**
 * @param name Must not be null. Must not be longer than 20 characters.
 * @throws IllegalArgumentException if the name is invalid
 */

public void setName(String name) throws IllegalArgumentException {
    validateName(name);
    name.length();
    this.name = name;
}
```
Setting Method Breakpoints

- Double Click in the margin

```java
public Product(String code, String name, double price) {
    validateCode(code);
    validateName(name);
    validatePrice(price);
    this.code = code;
    this.name = name;
    this.price = price;
}
```
Method Breakpoint Properties

Suspend on:
- Method Entry
- Method Exit

Suspend Policy:
- Suspend Thread
- Suspend VM
Class Loading
Breakpoints
Exception Breakpoints
Watch Point Breakpoints

- Double Click in the margin on a field
Watch Point Properties

Field: name

- Enabled

- Hit Count:

Suspend on:

- Field Access
- Field Modification

Suspend Policy:

- Suspend Thread
- Suspend VM
Enough Presentation

Let's Debug

- JUnit Test
- Drives code
- Helps find bugs
- Control over execution
Agenda

- Debugging
- Debugging Perspective
- Debugging Java Applications
- Debugging Remotely
Remote Debugging

- Get VM listening for connections
  - Use `-Xdebug` to start and listen
  - Use `-Xrunjdwp:args` for more options

Example

- Use `-Xdebug -Xrunjdwp:transport=dt_socket,server=y,address=8111,suspend=n`
Enough Presentation

Let's Debug

- Cactus Test
  - Drives code
  - Helps find bugs
  - Control over execution
Conclusion

- IDE Debuggers are powerful and full featured
  - Stop Logging
  - Remote Debugging
    - Works like a champ and prevents costly redeploy to add logging statements