Continuous Integration

Integrating External Tools

Overview

- Techniques for integrating external tools
- How to fail the build
- Displaying custom reports

External Tools

- Test runners
- Code coverage
- Static code analysis
- Deployment

Test Runners

- Running tests
- Failing the build on failed test(s)
- Displaying test results

Demo

Coverage

- **Measure percentage of production code executed by tests**
- **Fail build if coverage < desired percent**
- **Typically 60 to 80%**
- **Above 90% typically diminishing returns**
  - Hard to test edge cases
  - Integration with 3rd-party code
  - Difficult to test UI-related code

100% Coverage Myth

```
public class Calculator {
    public long Add(int x, int y) {
        return x + y;
    }
}

[Test]
public void CanAddTwoNumbers() {
    var calc = new Calculator();
    var result = calc.Add(1, 2);
    Assert.That(result, Is.EqualTo(3));
}
```

100% COVERAGE

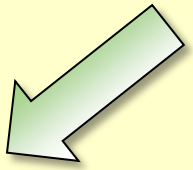
100% COVERAGE

100% != 100% ERROR FREE

100% Coverage Myth

```
public class Calculator {
    public long Add(int x, int y) {
        return x + y;
    }
}

[Test]
public void CanAddTwoLargeNumbers() {
    var calc = new Calculator();
    var result = calc.Add(int.MaxValue, int.MaxValue);
    Assert.That(result, Is.EqualTo(2L*int.MaxValue));
}
```



Integrating Coverage

- Running coverage on tests
- Failing the build on low coverage
- Displaying coverage results

Demo

Static Code Analysis

- Running static code analysis
- Failing the build on failed metric
- Displaying metric results

Demo

Summary

- Integrating via the command line
- Integrating via custom build tasks
- Failing the build by exit code
- Failing the build by analyzing output
- Generating reports
- Displaying reports