- Ukázková aplikace je ke stažení na https://www.wug.cz/brno/akce/836-WUG-Days-2016/program
  u přednášky „Jak na testovatelné webové aplikace“ jako „AddNodeWizard.zip“.
- Vyžaduje Visual Studio 2015.

rite testable web applications

Jiří Tomek
jiri.tomek@solarwinds.com

solarwinds
Unexpected Simplicity™

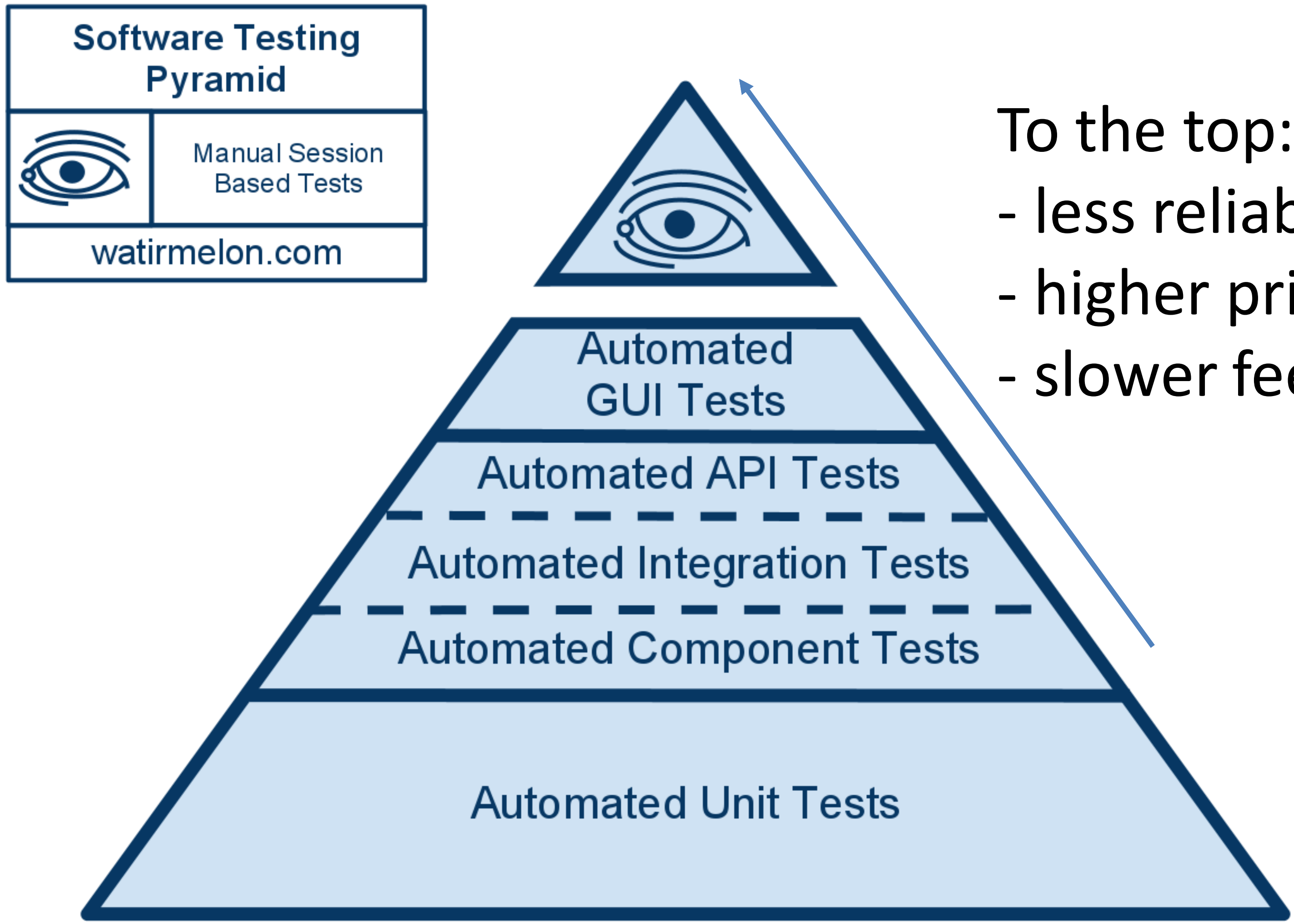
Agenda

- Testing? Why should I care?
- Test pyramid
- Different kinds of tests
  - Unit tests
  - Integration tests
  - End to end tests
- Sample application
  - AngularJS, ASP.NET WebApi 2

Why testing?

- » Tests define the behavior of application
- » Allow safe changes and refactoring
- » Simplify debugging
  - Debugging against unit test is much cheaper than against live application
- » Improve code quality
- » What to test?
  - Consider gain/price ratio
  - Parts that have high impact on functionality bring more benefit

Test pyramid



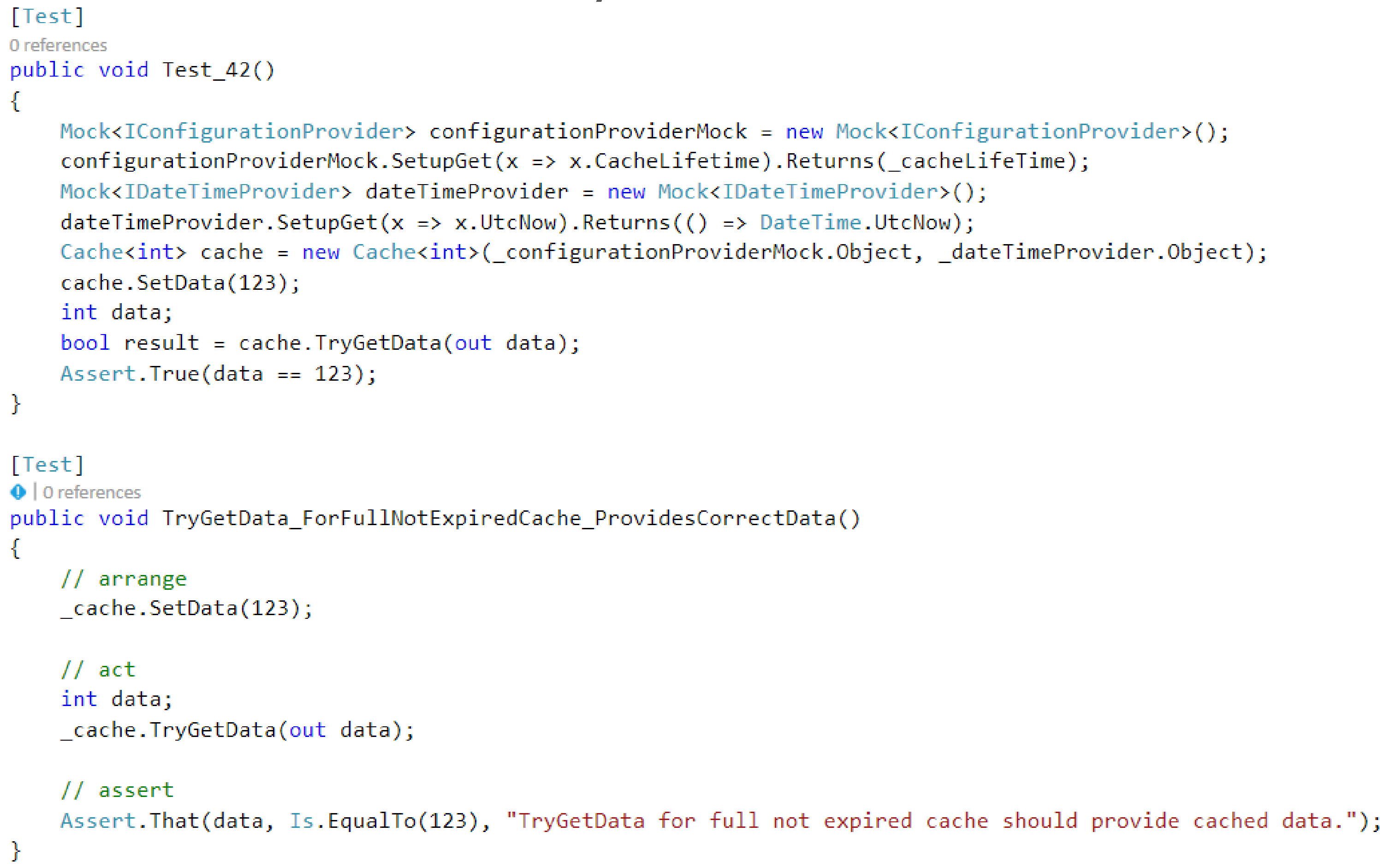
To the top:
- less reliable
- higher price
- slower feedback

Unit tests

- Unit test is code as any other
  - Needs to be maintained
  - Can be debugged
- Tests just one class/component
- Should be:
  - Easy to write
  - Readable
  - Reliable
  - Fast
  - Independent
  - Deterministic
  - No need for special test code
- It’s cheapest kind of test

Readable

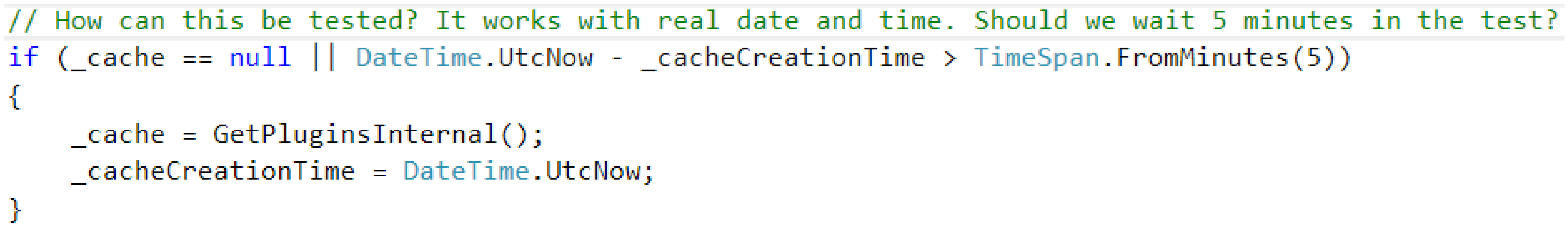
- » Name of the test clearly says what it does
- » Code is clean and easily understandable



```
[Test]
0 references
public void Test_42()
{
    Mock<IConfigurationProvider> configurationProviderMock = new Mock<IConfigurationProvider>();
    configurationProviderMock.SetupGet(x => x.CacheLifetime).Returns(_cacheLifeTime);
    Mock<IDateTimeProvider> dateTimeProvider = new Mock<IDateTimeProvider>();
    dateTimeProvider.SetupGet(x => x.UtcNow).Returns(() => DateTime.UtcNow);
    Cache<int> cache = new Cache<int>(_configurationProviderMock.Object, _dateTimeProvider.Object);
    cache.SetData(123);
    int data;
    bool result = cache.TryGetData(out data);
    Assert.True(data == 123);
}

[Test]
| 0 references
public void TryGetData_ForFullNotExpiredCache_ProvidesCorrectData()
{
    // arrange
    _cache.SetData(123);

    // act
    int data;
    _cache.TryGetData(out data);

    // assert
    Assert.That(data, Is.EqualTo(123), "TryGetData for full not expired cache should provide cached data.");
}
```

Enemies of testable code

- Static classes/methods
  - You can’t replace them with custom logic for testing



```
// How can this be tested? It works with real date and time. Should we wait 5 minutes in the test?
if (_cache == null || DateTime.UtcNow - _cacheCreationTime > TimeSpan.FromMinutes(5))
{
    _cache = GetPluginsInternal();
    _cacheCreationTime = DateTime.UtcNow;
}
```

- Singleton
  - Only one instance holding the state
  - Can’t parallelize tests
  - Can’t reset to default state
- Global state
  - Environment variables
  - Configuration files

SOLID

- SOLID principles
  - Single responsibility principle
    - Class does just one thing
  - Open/Closed principle
    - Class is opened for extension but closed for changes
    - Easy to add new functionality without need to touch existing code
  - Liskov substitution principle
    - Class can be replaced by its subclass without affecting functionality
  - Interface segregation
    - Interface should be small and focused
  - Dependency Inversion
    - Concrete classes depend on abstract interfaces, not vice versa

Integration tests

- Test more classes or components together
- Focused on component integration
  - Correct use of API
- Good to use when
  - Testing component alone is too complex to setup
  - Component integration is not trivial
  - To test complex workflow
- More expensive than unit tests but still good
  - Slower
  - Harder to maintain

End-to-end tests

- Run against live application
- Test whole application or its major part
- Use public interface of application
  - API
  - UI
- Most expensive
  - Time to setup the test
  - Time to setup the environment
  - Execution time
- Protractor - http://www.protractortest.org/

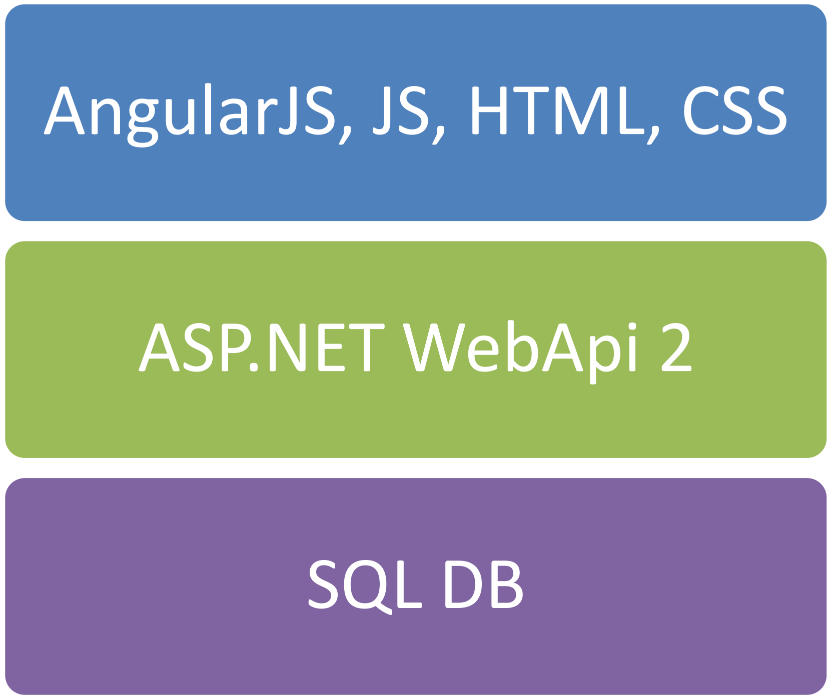
UI tests

- Test UI behavior
- Without real backend
  - Mock API
- Protractor HTTP mock - https://github.com/atecarlos/protractor-http-mock

Sample application

Web application

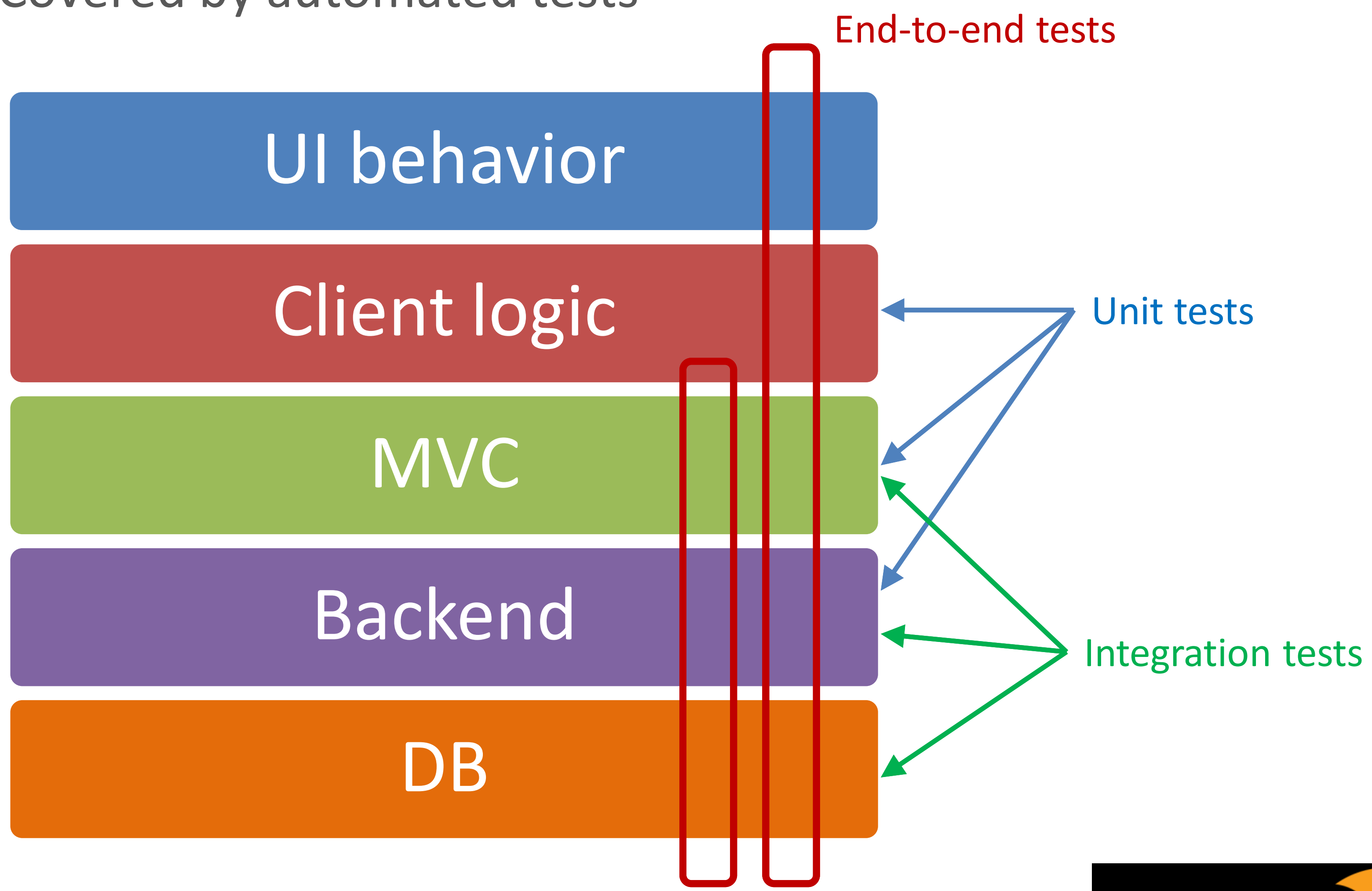
- Presentation layer
  - Web browser
  - TypeScript/JavaScript, AngularJS, HTML, CSS
- Business layer
  - Web server
  - ASP.NET WebApi 2, C#
- Data layer
  - Entity framework
  - SQL database



solarwinds

Testable?

- Covered by automated tests



Switch to VS

Useful links

- TypeScript - https://www.typescriptlang.org/
- Moq - https://github.com/Moq/moq4/wiki/Quickstart
- AngularJS – https://angularjs.org
- Protractor - http://www.protractortest.org/
- Protractor HTTP mock - https://github.com/atecarlos/protractor-http-mock
- Effort - https://github.com/tamasflamich/effort

SolarWinds

- 30+ products for IT Management
- Microsoft stack: C#, MS SQL, AngularJS
- Brno office – SolarWinds R&D center
- https://www.solarwindsmeetup.com/
- Career opportunities
  - http://solarwinds.jobs/
- Visit our booth

WE TURN COFFEE INTO CODE

solarwinds