# Avoid Hard-Coding in Unit Tests

**Paul D. Sheriff**
Business/IT Consultant PDS Consulting

psheriff@pdsa.com www.pdsa.com

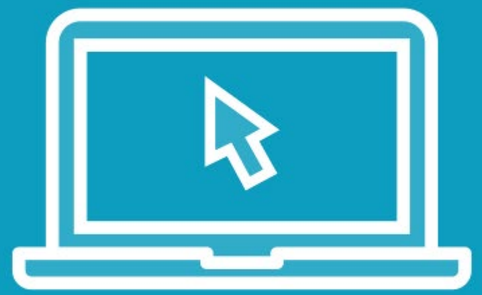
# Module Overview

- Use programming best practices
  - Constants
  - Settings file
  - Write output using TestContext property
  - Create a base test class

# Best Practices

- Don’t hard code
- Use constants
- Store data in setting files
- Setup and tear down as close to each unit test as possible
- Use comments
- Use good variable and method names

# Demo



- **Use constants**

# TestContext Property

# TestContext Property

**Create property 'TestContext' in all your test classes**

**Automatically created by unit test framework**

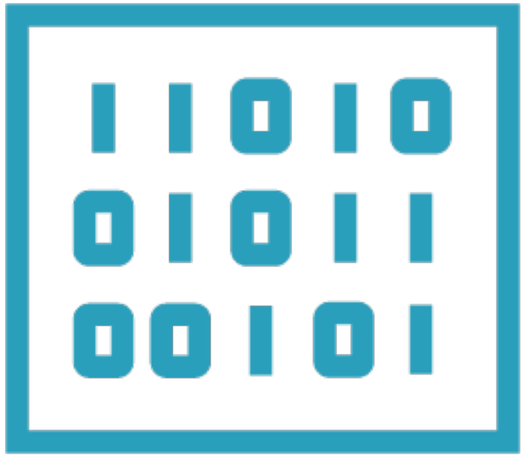
**Property is set before each test is run**

**Useful in data-driven tests**

**Useful for accessing test information**

**Useful for writing information into test**

# Properties



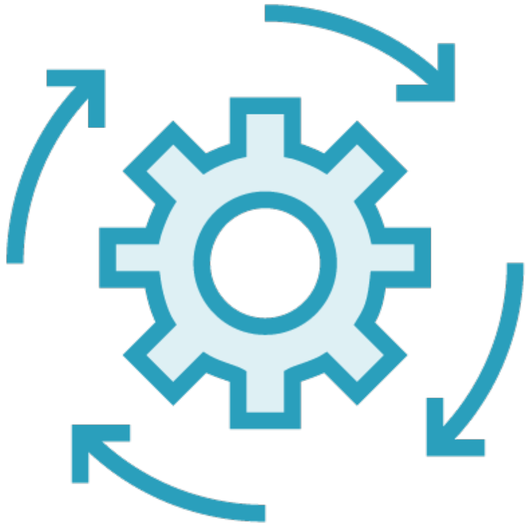
**DataConnection**

**DataRow**

**DeploymentDirectory**

**TestName**

# Methods

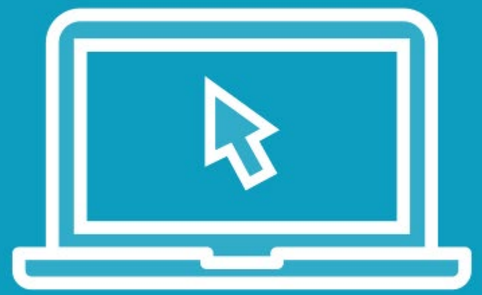


**BeginTimer**

**EndTimer**

**WriteLine**

# Demo



- **TestContext property**

# Add Settings File

# Add Settings File

**Add MyClasses.runsettings to unit test project**

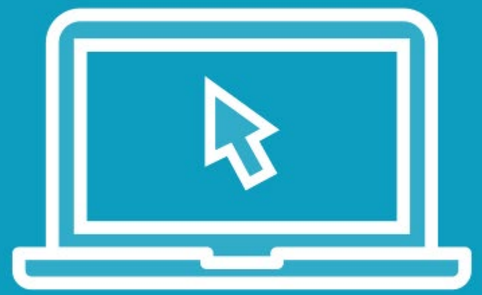
**Add <RunSettings> element**

```
<Parameter name=
 "GoodFileName"
value=
"TestFile.txt" />
```

**Set field in MyClassesTest class from config file**

**Use field in FileNameDoesExist test**

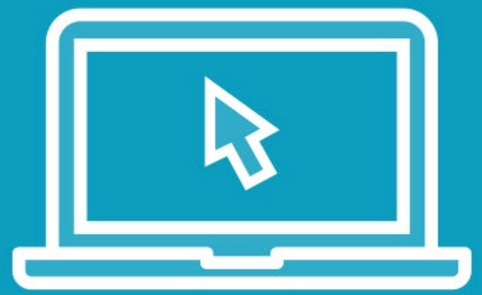
# Demo



- **Use a configuration file**

# Create/Delete File

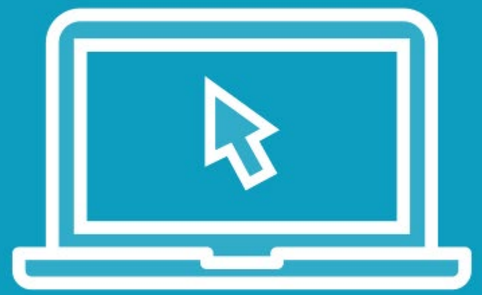
# Demo



- **Create and delete file**

# Create Base Class

# Demo



- **Create base test class**

# Module Summary

- Use good programming practices
- Use constants
- Create a TestContext property
- Use a settings file
- Setup and teardown in test
- Create a base test class

- Search for unit testing at www.pdsa.com/blog

# Up Next:
# Initialization and Cleanup