IT’S NOT JUST FEATURE TOGGLES

Dan Piessens
ABOUT ME

- Senior Agile Consultant
- 14 Years Experience as Developer, IT Consultant, Architect, Trainer, Coach
- MS Patterns & Practices Champion
WELCOME TO
PARTS UNLIMITED

“It was a good idea at the time . . .”
WHERE THEY STARTED

- Develop
- Build Locally
- Locally Test
- Hack settings
- Copy to Server
- Wait for Someone to Yell
WHERE THEY’RE AT

- Develop
- Gather Feedback
- CI Build
- CI Test
- Package
- Deploy
WHAT MAKES THIS WORK?

- Small Work Batches
- Automated Quality Gates
  - Unit Tests, Code Coverage, Quality Checkers
- Repeatable Process
  - Goal: No Manual Steps During Deployment
“THE GOAL” FOR DEVOPS

“This book will have a profound effect on IT, just as *The Goal* did for manufacturing.” --Jez Humble, co-author *Continuous Delivery*

“This is the IT swamp draining manual for anyone who is neck deep in alligators.” --Adrian Cockroft, Cloud Architect at Netflix

“This is *The Goal* for our decade, and is for any IT professional who wants their life back.” --Charles Betz, IT architect, author “Architecture and Patterns for IT”
THE SHARED SANDBOX

- Migrate from feature branches to single trunk development
- Branching still requires merging, even if it’s small
- Feedback on branches is delayed
- Environments on Demand
  - How many environments do you need?
Trunk
ACHIEVING TRUNK DEVELOPMENT

- Introduce feature toggles
- Use modular coding practices
- Automated testing to ensure quality baseline
A TALE OF ONE (REAL) TEAM

Deployments

@dpiessens
FLEXIBLE RELEASES
FEATURE TOGGLIES

- A mechanism to switch between features at runtime
- Separates delivery from deployment
- Typically done at the UI / Service layer
- Scary?

You already have this in your application… User Login and Authorization!
HomePageModel homePageModel;

if (ConfigurationManager.AppSettings["ToggleMyFeature"] == "true")
{
    homePageModel = new HomePageModel
    {
        WelcomeMessage = "Well this is different...",
        SubMessage = "Something changed, not sure what",
        Title = "Base Page"
    };
}
else
{
    homePageModel = new HomePageModel
    {
        WelcomeMessage = "Welcome to my page!",
        SubMessage = "Now this is cool :)",
        Title = "Home Page"
    };
}
Easier to Refactor

```csharp
HomePageModel homePageModel;

if (ToggleManager.IsEnabled<ToggleMyFeature>())
{
    homePageModel = new HomePageModel
    {
        WelcomeMessage = "Well this is different...",
        SubMessage = "Something changed, not sure what",
        Title = "Base Page"
    };
}
else
{
    homePageModel = new HomePageModel
    {
        WelcomeMessage = "Welcome to my page!",
        SubMessage = "Now this is cool :)",
        Title = "Home Page"
    };
}
```

It's a Class!
CODE BASED TOGGLERS

- Find references more easily
- Switch strategies
- Detect nesting, dependencies
TOGGLER ARE TECH DEBT

TECHNICAL DEBT EVERYWHERE

TECHNICAL DEBT? YOU KEEP USING THOSE WORDS.

I DO NOT THINK THEY MEAN WHAT YOU THINK THEY MEAN
OR...

WORKED FINE IN DEV

OPS PROBLEM NOW
TOGGLES AND DEPLOYMENT

- Toggles do stuff! = Has a performance impact
- Must correlate changes to runtime feedback
- Flip toggles via deployments
DEPLOYMENT KATA

- Take and application and walk the deployment process
  - Doesn’t need to be difficult
- Must be done as a cross-functional team!
- No “problem clusters”
- Helps teams absorb practices and domain knowledge
IT’S NOT THAT EASY!
BUT YOUR APP LOOKS LIKE THIS!
OR MAYBE MORE LIKE THIS

Azure Active Directory

Azure Storage

Azure SQL DB

Input file

Payment system

Output file

Integration service

File share

Scans service

Web site

Windows event log

User profile

LDAP query

Database

Active Directory

Browser

Azure VPN
INVERSION OF CONTROL
EXAMPLE: STORAGE

User
Uploads Image

Website

IStorageService

File System

Cloud Storage

Configuration Change
WHERE THIS HELPS

- Hybrid Application Development
  - One provider in local datacenter, the other in the cloud
- Testability
- Separation of Concerns
- Development vs. Production modes
EXTENDING TO FEATURE TOGGLES

- Control your container setup with toggles

- Separate container setup stages
  - Initial infrastructure setup
  - Toggle configuration
  - Application dependency setup
COMMAND & QUERY RESPONSIBILITY SEGREGATION

A CQRS Journey

1. Our domain
2. Decomposing the domain
3. Bounded context #1: Orders and Registrations
4. Extending and enhancing the Orders and Registrations bounded context
5. Preparing for the V1 release
6. Versioning our system
7. Adding resilience and optimizing performance
8. Lessons learned

Testing
Task-based UIs

Event sourcing
Domain-driven design

CQRS

Sagas and process managers
Bounded contexts
Domain events

HOW IT WORKS

- Updates write to an event store
- Workers process events
- Aggregate state is persisted
- System reads aggregate state
DONT use this for every system

Do research and read

Build the pattern incrementally
  - Pro Tip: Try a small application

Think about deployment
TO THE DATABASE!
3 STAGE UPDATES

Orders

<table>
<thead>
<tr>
<th>Id</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10001</td>
</tr>
<tr>
<td>2</td>
<td>10002</td>
</tr>
<tr>
<td>3</td>
<td>10003</td>
</tr>
</tbody>
</table>

Web Service
3 STAGE UPDATES

STEP 1: DEPLOY NEW FUNCTIONALITY DISABLED

Orders

<table>
<thead>
<tr>
<th>Id</th>
<th>Number</th>
<th>Urgent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10001</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>10002</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>10003</td>
<td>F</td>
</tr>
</tbody>
</table>

Web Service
No Urgent Code
Uses Urgent Code
### 3 STAGE UPDATES

**STEP 2: TOGGLE NEW SERVICE CODE**

<table>
<thead>
<tr>
<th>Id</th>
<th>Number</th>
<th>Urgent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10001</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>10002</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>10003</td>
<td>F</td>
</tr>
</tbody>
</table>

**Web Service**

- No Urgent Code
- Uses Urgent Code

@dpiessens
3 STAGE UPDATES

STEP 3: REMOVE OLD FUNCTIONALITY

<table>
<thead>
<tr>
<th>Orders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Web Service
Uses Urgent Code
TESTING

Really...!??
Simple, component testing

Ensure toggle framework is an interface

Don’t get fancy here!
This example looks pretty good right?

```
Feature: Add a product from the list to the cart

 Scenario: Tom adds a single B-10 model airplane to an empty cart
     Given Tom is on the model airplanes webpage
     And Tom selects view plane on the B-10 model airplane
     When Tom presses Add to Cart
     Then Tom should be on the My Cart page
     And The cart should contain B-10 bomber
```
**EXAMPLE: STORE PURCHASE**

Let’s look at that scenario again . . .

---

```
Feature: Add a product from the list to the cart

Scenario: Tom adds a single B-10 model airplane to an empty cart
  Given Tom is on the model airplanes webpage
  And Tom selects view plane on the B-10 model airplane
  When Tom presses Add to Cart
  Then Tom should be on the My Cart page
  And The cart should contain B-10 bomber
```

- Doesn't this mean the same thing?
- And this?
- And this?
- And this?
EXAMPLE: STORE PURCHASE

Your developer would have to wire up this!

```csharp
[Binding]
public class AddToCartSteps
{
    [Given("Tom is on the add_to_cart webpage")]
    void GivenTomIsOnTheModelAirplanePage()
    {
    }

    [Given("Tom selects view plane on the add to model airplane")]
    public void ThenTomSelectsViewPlaneOnTheModelAirplane()
    {
    }

    [When("Tom presses Add to Cart")]
    public void WhenTomPressesAddToCart()
    {
    }

    [Then("Tom should be on the My Cart page")]
}
EXAMPLE: STORE PURCHASE

But if we didn’t care... how would it look?

```
[Binding]
public class AddToCartSteps
{
    [Given("Tom is on the model airplanes webpage")]
    public void GivenTomIsOnTheModelAirplanesWebpage()
    {
        // What the heck is this???
        var map = new UIMap();
        map.AddBomberToCart();
    }
```
ENTER: SPECBIND

SpecFlow  →  SpecBind  →  Web Automation

Common Steps  ←  SpecBind  ➔  Page Model

@dpiessens
SPECBIND STEPS

- Creates a common vocabulary
- Uses correct language tenses
- **Given** I chose Add To Cart
- **When** I choose Add To Cart
SPECBIND STEP EXAMPLES

- Select an item on a page

- Enter Data

- Validating Data

  When I choose **Add To Cart**

  When I enter data

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Bob</td>
</tr>
</tbody>
</table>

  Then I see

<table>
<thead>
<tr>
<th>Field</th>
<th>Rule</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Equals</td>
<td>Bob</td>
</tr>
</tbody>
</table>
Feature: Add a product from the list to the cart

Scenario: Tom adds a single B-10 model airplane to an empty cart

Given I navigated to the home page
And I chose model airplanes
And I was on the product list page
And I was on list product items item 1
And I chose view
And I was on the product details page
When I choose Add To Cart
Then I am on the shopping cart page
And I see the products list starts with

<table>
<thead>
<tr>
<th>Field</th>
<th>Rule</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Name</td>
<td>Equals</td>
<td>B-10 Bomber</td>
</tr>
<tr>
<td>Quantity</td>
<td>Equals</td>
<td>1</td>
</tr>
</tbody>
</table>
SIMP liFY DEVELOP ER WORK

- The Page Model
  - Simplifies Development Work
  - Reusable
  - Refactorable

- Can be developed asynchronously!
```csharp
[PageNavigation("/")]
public class HomePage : HtmlDocument
{
    public HomePage(UITestControl parent) : base(parent)
    {
    }

    [ElementLocator(Class = "checkout", Title = "Proceed to checkout")]
    public HtmlHyperlink Checkout { get; set; }

    [ElementLocator(Id = "modellink")]
    public HtmlHyperlink ModelAirplanes { get; set; }
}
```
PAGE MODEL: HOME PAGE

```csharp
[PageNavigation("/")]
public class HomePage : HtmlDocument
{
    public HomePage UITestControl parent) : base(parent)
    {
    }

    [ElementLocator(Class = "checkout", Title = "Proceed to checkout")]
    public HtmlHyperlink Checkout { get; set; }

    [ElementLocator(Id = "modellink")]
    public HtmlHyperlink ModelAirplanes { get; set; }
}
```
Use same toggle mechanism

Testing On/Off Scenarios
  - Multiple Environments
  - Build toggle flips into test framework
GETTING FEEDBACK
RUNTIME FEEDBACK

- Instrument your applications at runtime!
- Many tools available
  - New Relic
  - Application Insights
  - Splunk
  - Raygun.io
- Include User Analytics
  - Google
  - All above tools
DIG DEEPER FOR DATA

- Monitoring tools have APIs for tracing / logging
- Instrument key transactions in your system
- Track performance end-over-end for deployments
- Choose a logging framework and leverage it!
  - Log to a common location
Find errors during development

Make performance reviews part of your DoD

Example: Glimpse

- Trace your application
- Many plugins available
- http://www.getglimpse.com
TAKE AWAY CONCEPTS

- Codify Feature Toggles
- Toggles are technical debt!
- Go deeper than the UI if needed
- Automate the testing of toggles
- Separate deployments from releases

"Deployments are like exercise, the more you do them the less it hurts"

-Dan Piessens
RESOURCES

- Moving Applications to the Cloud 3rd Edition

- Building Hybrid Applications in the Cloud on Microsoft Azure

- CQRS Journey

- Octopus Deploy
  - http://www.octopusdeploy.com
QUESTIONS?

Thank You!

Email: dan.piessens@centare.com  Blog: www.danpiessens.com  Twitter: @dpiessens