Fundamentals:
The Interface Segregation Principle

Steve Smith
http://pluralsight.com/

Outline

- ISP Defined
- The Problem
- An Example
- Refactoring to Apply ISP
- Related Fundamentals

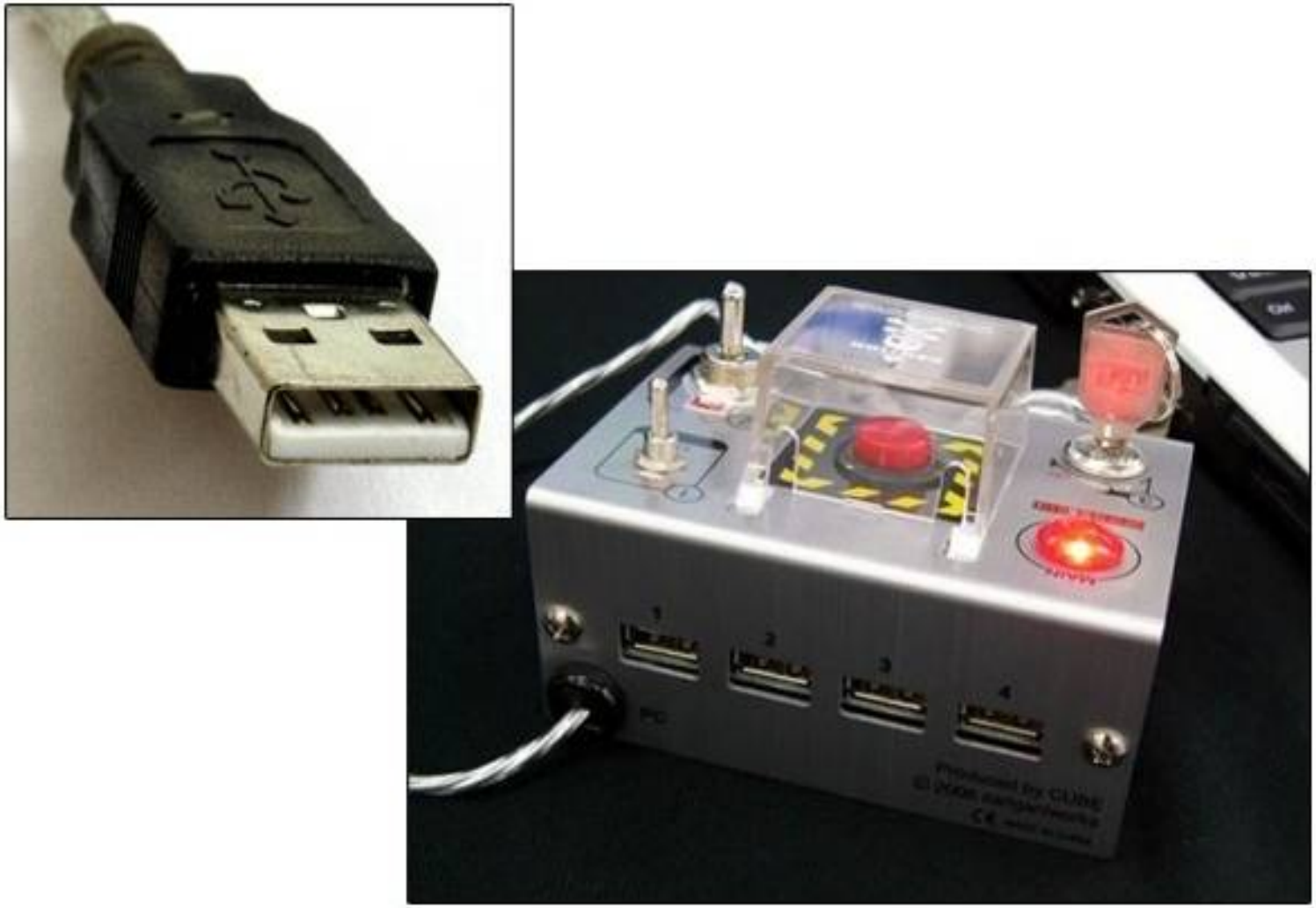
ISP: The Interface Segregation Principle

The Interface Segregation Principle states that Clients should not be forced to depend on methods they do not use.

Agile Principles, Patterns, and Practices in C#

Corollary:

Prefer small, cohesive interfaces to “fat” interfaces



INTERFACE SEGREGATION PRINCIPLE

You Want Me To Plug This In, Where?

What's an Interface?

Interface keyword/type

```
public interface IDoSomething { … }
```

Public interface of a class

```
public class SomeClass { … }
```

What does the *client* see and use?

Demo

Violating ISP

The Problem

Client Class (Login Control) Needs This:

```
private void AuthenticateUsingMembershipProvider(AuthenticateEventArgs e)
{
    e.Authenticated = LoginUtil.GetProvider(this.MembershipProvider).ValidateUser(this.UserNameInternal, this.PasswordInternal);
}
```

Forced to use this

```
public class CustomMembershipProvider : MembershipProvider
{
    public override MembershipUser CreateUser(string username, string password, string
bershipCreateStatus status)...
    public override bool ChangePasswordQuestionAndAnswer(string username, string passv
    public override string GetPassword(string username, string answer)...
    public override bool ChangePassword(string username, string oldPassword, string ne
    public override string ResetPassword(string username, string answer)...
    public override void UpdateUser(MembershipUser user)...
    public override bool ValidateUser(string username, string password)...
    public override bool UnlockUser(string userName)...
    public override MembershipUser GetUser(object providerUserKey, bool userIsOnline)
    public override MembershipUser GetUser(string username, bool userIsOnline)...
    public override string GetUserNameByEmail(string email)...
    public override bool DeleteUser(string username, bool deleteAllRelatedData)...
    public override MembershipUserCollection GetAllUsers(int pageIndex, int pageSize,
    public override int GetNumberOfUsersOnline()...
    public override MembershipUserCollection FindUsersByName(string usernameToMatch,
    public override MembershipUserCollection FindUsersByEmail(string emailToMatch, int
    public override bool EnablePasswordRetrieval...
    public override bool EnablePasswordReset...
    public override bool RequiresQuestionAndAnswer...
    public override string ApplicationName { get; set; }
    public override int MaxInvalidPasswordAttempts...
    public override int PasswordAttemptWindow...
    public override bool RequiresUniqueEmail...
    public override MembershipPasswordFormat PasswordFormat...
    public override int MinRequiredPasswordLength...
    public override int MinRequiredNonAlphanumericCharacters...
    public override string PasswordStrengthRegularExpression...
}
```

The Problem

- AboutPage simply needs ApplicationName and AuthorName

- Forced to deal with huge ConfigurationSettings class
- Forced to deal with actual configuration files

Interface Segregation violations result in classes that depend on things they do not need, increasing coupling and reducing flexibility and maintainability

Demo

Refactoring to a Better Design

ISP Smells

- Unimplemented interface methods:

```
public override string ResetPassword(
  string username, string answer)
{
      throw new NotImplementedException();
}
```

Remember these violate Liskov Substitution Principle!

ISP Smells

- Client references a class but only uses small portion of it

When do we fix ISP?

- **Once there is pain**
  - If there is no pain, there's no problem to address.
- **If you find yourself depending on a "fat" interface you own**
  - Create a smaller interface with just what you need
  - Have the fat interface implement your new interface
  - Reference the new interface with your code
- **If you find "fat" interfaces are problematic but you do not own them**
  - Create a smaller interface with just what you need
  - Implement this interface using an Adapter that implements the full interface

ISP Tips

- **Keep interfaces small, cohesive, and focused**
- **Whenever possible, let the client define the interface**
- **Whenever possible, package the interface with the client**
  - Alternately, package in a third assembly client and implementation both depend upon
    - Last resort: Package interfaces with their implementation

Summary

- **Don't force client code to depend on things it doesn't need**
- **Keep interfaces lean and focused**
- **Refactor large interfaces so they inherit smaller interfaces**
- **Related Fundamentals:**
  - Polymorphism
  - Inheritance
  - Liskov Substitution Principle
  - Façade Pattern
- **Recommended Reading:**
  - Agile Principles, Patterns, and Practices by Robert C. Martin and Micah Martin [http://amzn.to/agilepppcsharp]

Credits

- **Images Used Under License**
  - http://www.lostechies.com/blogs/derickbailey/archive/2009/02/11/solid-development-principles-in-motivational-pictures.aspx

For more in-depth online developer training visit

pluralsight

see what you can learn

on-demand content from authors you trust