Design Patterns

For Loosely Coupled Applications

Barry S. Stahl @bsstahl

http://www.CognitiveInheritance.com

FAVORITE PHYSICISTS & MATHEMATICIANS

FAVORITE PHYSICISTS

1. Harold "Hal" Stahl
2. Carl Sagan
3. Richard Feynman
4. Marie Curie
5. Nikola Tesla
6. Albert Einstein
7. Neil Degrasse Tyson
8. Niels Bohr
9. Galileo Galilei
10. Michael Faraday

Other notables: Stephen Hawking, Edwin Hubble

FAVORITE MATHEMATICIANS

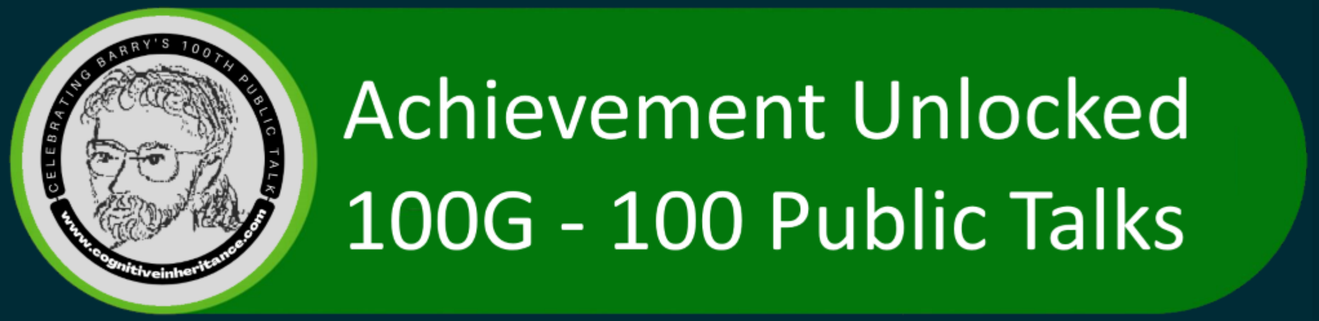
1. Ada Lovelace
2. Alan Turing
3. Johannes Kepler
4. Rene Descartes
5. Isaac Newton
6. Leonardo Fibonacci
7. George Boole
8. Blaise Pascal
9. Johann Gauss
10. Grace Hopper

Other notables: Daphne Koller, Benoit Mandelbrot

SOME OSS PROJECTS I RUN

1. Liquid Victor : Media tracking and aggregation [used to assemble this presentation]
2. Prehensile Pony-Tail : A static site generator built in c#
3. TestHelperExtensions : A set of extension methods helpful when building unit tests
4. Conference Scheduler : A conference schedule optimizer
5. IntentBot - A microservices framework for creating conversational bots on top of Bot Framework
6. LiquidNun : Library of abstractions and implementations for loosely-coupled applications
7. Toastmasters Agenda : A c# library and website for generating agenda's for Toastmasters meetings

HTTP://GIVECAMP.ORG


CELEBRATING BARRY'S 100TH PUBLIC TALK
www.cognitiveinheritance.com
Achievement Unlocked
100G - 100 Public Talks

Come do Amazing things at Carvana!

Thank You to the Code PaLOUsa Sponsors

Progress

WAYSTAR

Bastian SOLUTIONS
a TOYOTA ADVANCED LOGISTICS company

<code> LOUISVILLE

eblu SOLUTIONS

twilio

Scout

aws

cbts

<prosoft>

CGI

Friends of Code PaLOUsa

Asperitas

Couchbase

DATASTAX

mongoDB.

redis

ROCKET Mortgage

WHY LOOSE COUPLING

APPLICATION TCO

- The Total Cost of Ownership of an application is the sum of all the costs of the application throughout its life-cycle.
- These costs include creating, deploying, extending, maintaining and decommissioning the application.

LOOSE COUPLING REDUCES TCO

- Tight coupling makes it more difficult to modify one concern of the application without impacting others.
- Maintenance & Extension : more difficult == more expensive

THE SOLID PRINCIPLES

- *S* - Single Responsibility Principle
- *O* - Open/Closed Principle
- *L* - Liskov Substitution Principle
- *I* - Interface Segregation Principle
- *D* - Dependency Inversion Principle

THE SOLID PRINCIPLES

- *S* - Single Responsibility Principle
- ~~*O* - Open/Closed Principle~~
- ~~*L* - Liskov Substitution Principle~~
- ~~*I* - Interface Segregation Principle~~
- *D* - Dependency Inversion Principle

AGENDA

- Design Principles
  - The Single Responsibility Principle
  - The Dependency Inversion Principle
- A Sample Tightly-Coupled Application
- Design Patterns
  - *Data Abstraction* using the **Repository Pattern**
  - *Dependency Inversion* using the **Dependency Injection Pattern**
  - *Logic Abstraction* using the **Strategy Pattern**

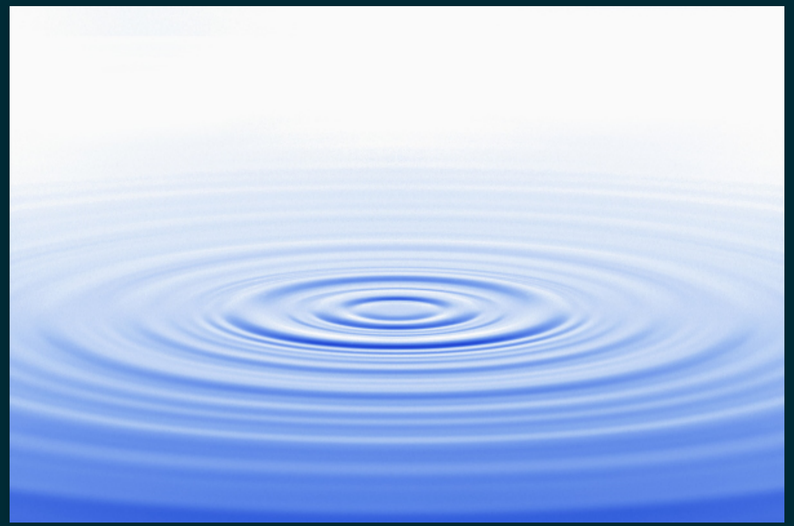
THE SINGLE RESPONSIBILITY PRINCIPLE

*A **class** should have only one reason to change*

- A class should have 1 and only 1 *role*
- A class should represent 1 and only 1 *abstraction*

Also applies to modules and methods

SRP - THE RIPPLE EFFECT



If we don't separate responsibilities, a change to the details of one aspect of an application requires that another area be changed, which requires another...

SRP - CODE SMELLS

BOLO FOR WHEN AN OBJECT'S CODE HAS TO BE CHANGED FOR MORE THAN ONE REASON

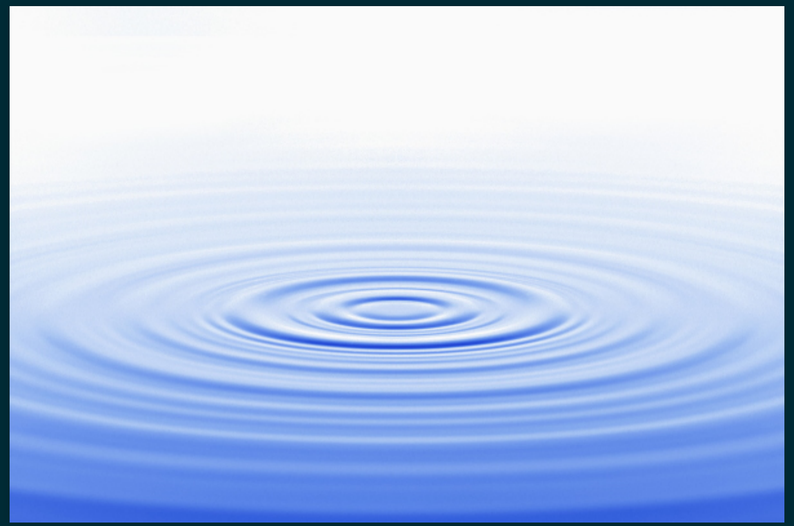
- database field name AND business rule
- business rule AND presentation logic

THE DEPENDENCY INVERSION PRINCIPLE

Entities should depend on abstractions, not on concrete implementations

- A class should **not** have knowledge of its dependencies
- A class should **only** have knowledge of the capabilities its dependencies provide

DIP - THE RIPPLE EFFECT



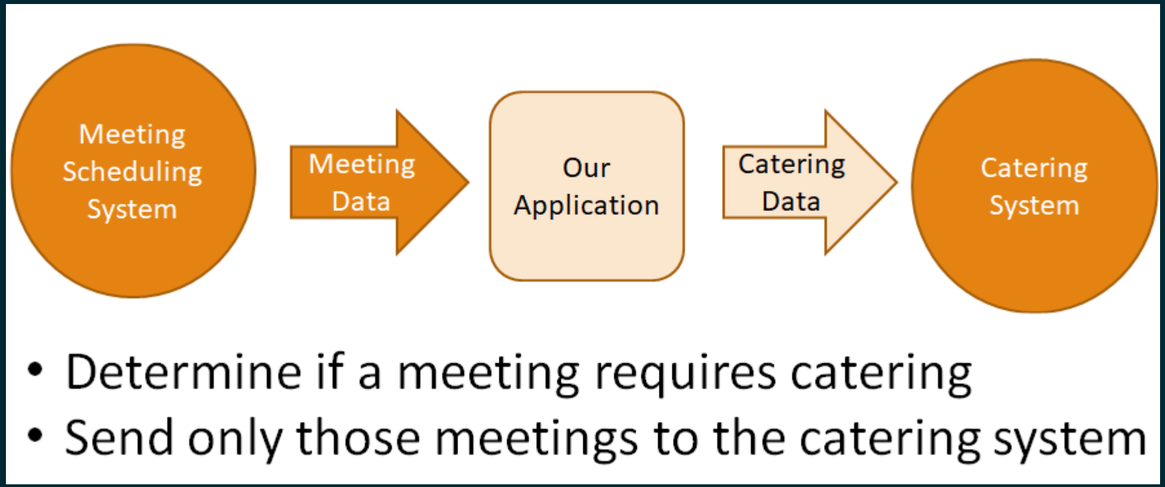
If we allow the details of an implementation to leak out, a change to the implementation of one aspect of an application requires that another area be changed, which requires another...

DIP - CODE SMELLS

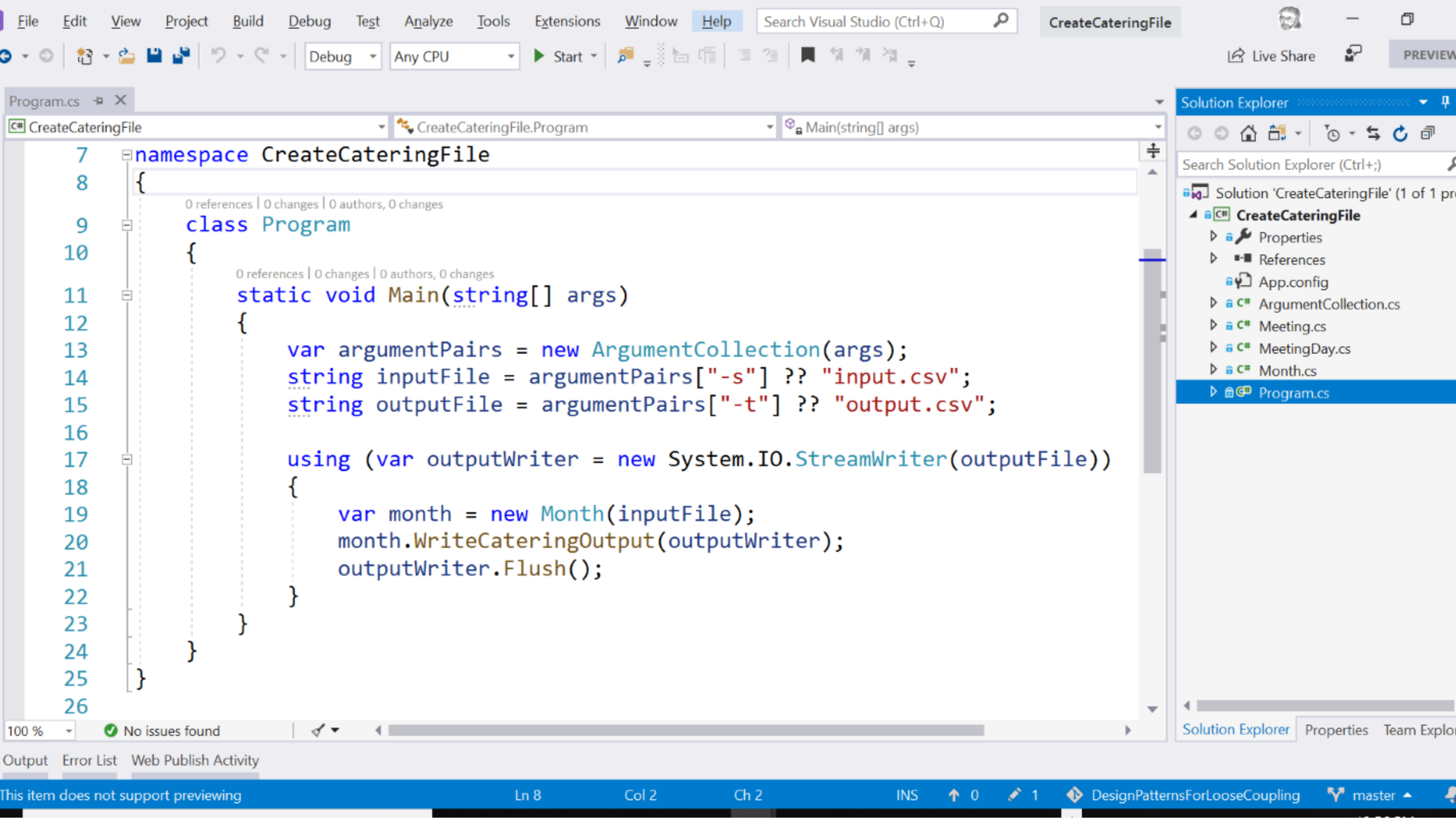
BOLO FOR WHEN AN OBJECT NEEDS TO KNOW DETAILS OF ITS DEPENDENCIES

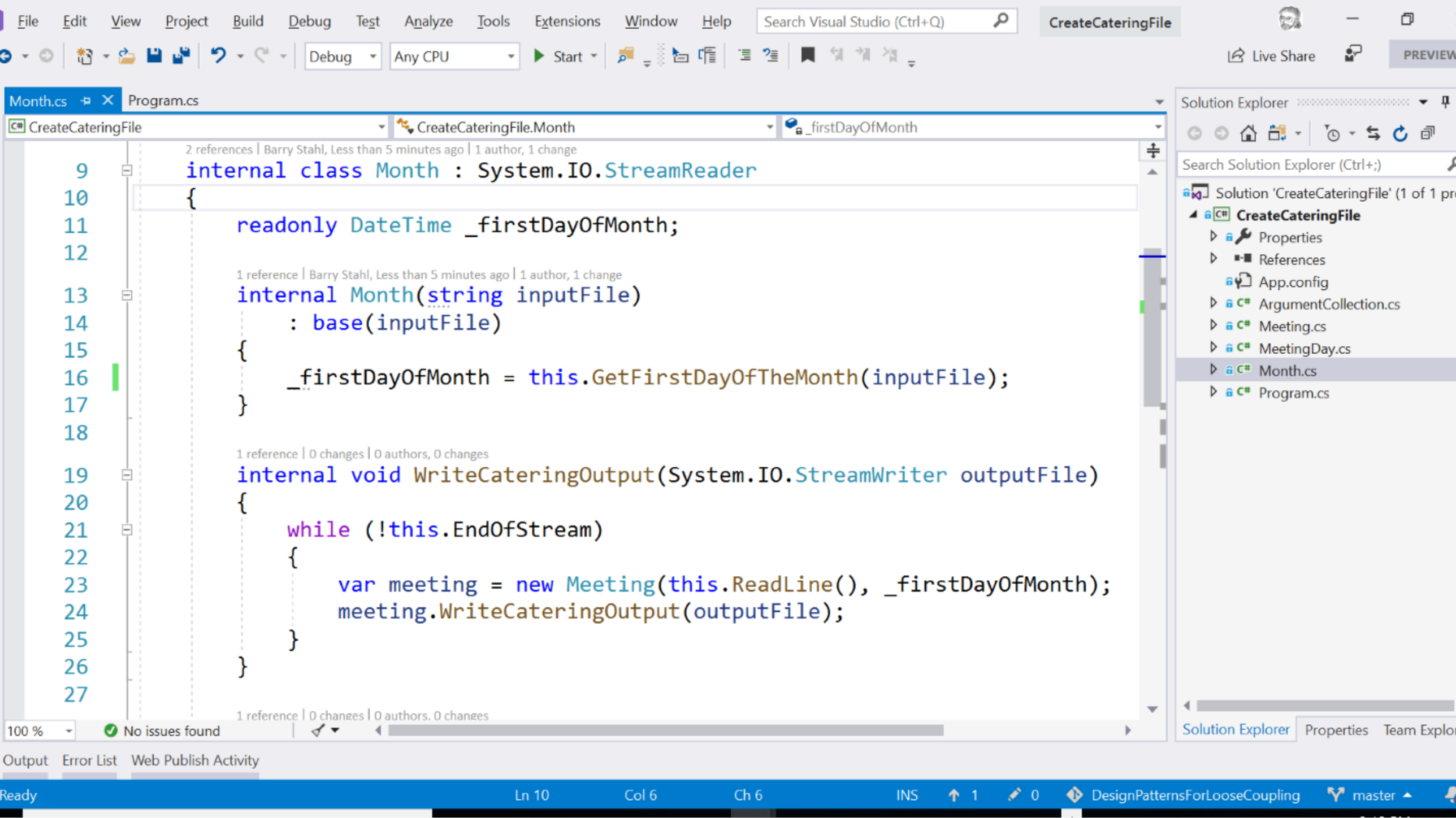
- Business rules are aware of storage details
- UI logic knows specifics of business rules
  - Validation is a common exception

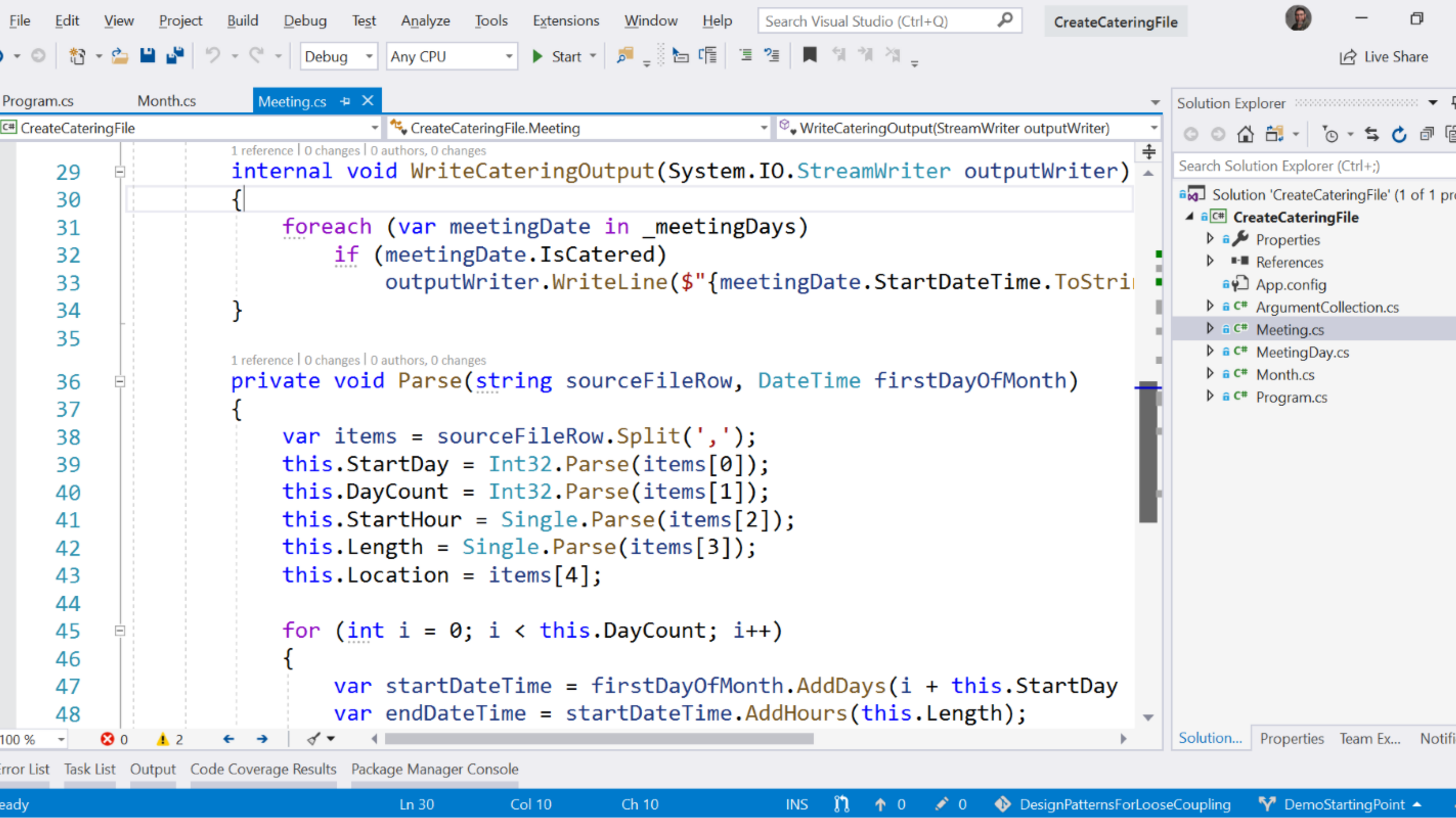
OUR DEMO APPLICATION



- Determine if a meeting requires catering
- Send only those meetings to the catering system


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringFile
Debug
Any CPU
Start
Live Share
Program.cs
CreateCateringFile
CreateCateringFile.Program
Main(string[] args)
namespace CreateCateringFile
{
0 references | 0 changes | 0 authors, 0 changes
class Program
{
0 references | 0 changes | 0 authors, 0 changes
static void Main(string[] args)
{
var argumentPairs = new ArgumentCollection(args);
string inputFile = argumentPairs["-s"] ?? "input.csv";
string outputFile = argumentPairs["-t"] ?? "output.csv";
using (var outputWriter = new System.IO.StreamWriter(outputFile))
{
var month = new Month(inputFile);
month.WriteCateringOutput(outputWriter);
outputWriter.Flush();
}
}
}
}
100 %
No issues found
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringFile' (1 of 1 pr
CreateCateringFile
Properties
References
App.config
ArgumentCollection.cs
Meeting.cs
MeetingDay.cs
Month.cs
Program.cs
Solution Explorer
Properties
Team Explo
Output
Error List
Web Publish Activity
This item does not support previewing
Ln 8
Col 2
Ch 2
INS
0
1
DesignPatternsForLooseCoupling
master


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringFile
Debug
Any CPU
Start
Live Share
Month.cs
Program.cs
CreateCateringFile
CreateCateringFile.Month
_firstDayOfMonth
2 references | Barry Stahl, Less than 5 minutes ago | 1 author, 1 change
internal class Month : System.IO.StreamReader
{
readonly DateTime _firstDayOfMonth;
1 reference | Barry Stahl, Less than 5 minutes ago | 1 author, 1 change
internal Month(string inputFile)
: base(inputFile)
{
_firstDayOfMonth = this.GetFirstDayOfTheMonth(inputFile);
}
1 reference | 0 changes | 0 authors, 0 changes
internal void WriteCateringOutput(System.IO.StreamWriter outputFile)
{
while (!this.EndOfStream)
{
var meeting = new Meeting(this.ReadLine(), _firstDayOfMonth);
meeting.WriteCateringOutput(outputFile);
}
}
1 reference | 0 changes | 0 authors, 0 changes
100 %
No issues found
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringFile' (1 of 1 pr
CreateCateringFile
Properties
References
App.config
ArgumentCollection.cs
Meeting.cs
MeetingDay.cs
Month.cs
Program.cs
Solution Explorer
Properties
Team Explo
Output
Error List
Web Publish Activity
Ready
Ln 10
Col 6
Ch 6
INS
1
0
DesignPatternsForLooseCoupling
master


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringFile
Debug
Any CPU
Start
Live Share
Program.cs
Month.cs
Meeting.cs
CreateCateringFile
CreateCateringFile.Meeting
WriteCateringOutput(StreamWriter outputWriter)
1 reference | 0 changes | 0 authors, 0 changes
internal void WriteCateringOutput(System.IO.StreamWriter outputWriter)
{
foreach (var meetingDate in _meetingDays)
if (meetingDate.IsCatered)
outputWriter.WriteLine($"{meetingDate.StartDateTime.ToStri
}
1 reference | 0 changes | 0 authors, 0 changes
private void Parse(string sourceFileRow, DateTime firstDayOfMonth)
{
var items = sourceFileRow.Split(',');
this.StartDay = Int32.Parse(items[0]);
this.DayCount = Int32.Parse(items[1]);
this.StartHour = Single.Parse(items[2]);
this.Length = Single.Parse(items[3]);
this.Location = items[4];
for (int i = 0; i < this.DayCount; i++)
{
var startDateTime = firstDayOfMonth.AddDays(i + this.StartDay
var endDateTime = startDateTime.AddHours(this.Length);
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringFile' (1 of 1 pr
CreateCateringFile
Properties
References
App.config
ArgumentCollection.cs
Meeting.cs
MeetingDay.cs
Month.cs
Program.cs
Solution...
Properties
Team Ex...
Notifi
100 %
0
2
Error List
Task List
Output
Code Coverage Results
Package Manager Console
eady
Ln 30
Col 10
Ch 10
INS
0
0
DesignPatternsForLooseCoupling
DemoStartingPoint

A TIGHTLY COUPLED APPLICATION

- Our App Violates **SRP**
  - Adding a field in the input file requires changes to the business-rule classes
  - Changing the order of fields in the output file requires changes to the business-rule classes
  - The **Month** and **Meeting** classes have multiple reasons to change
- Our App Violates **DIP**
  - Knowledge of the input stream is required to process the data
  - Knowledge of the output stream is required to produce the results
- The 3 primary aspects of this application, the **input**, **output** & **business-rules** are *tightly coupled*

DESIGN PATTERNS

An abstract solution to a well known problem

- Design Patterns can help implement this architecture
  - Provide a prototype to follow for implementation
  - Provide a common language to understand the solution

THE FACADE PATTERN

Facade - A simplified view of an API that fronts a more complicated interface

- Goals of a Facade
  - Abstract the caller from the underlying implementation
  - Simplify the API for the caller

THE REPOSITORY PATTERN

Repository - A somewhat standardized facade that fronts a data-store

- Goals of a Repository
  - Abstract the caller from the underlying implementation
  - Simplify the API for the caller
- Differences from other Facades
  - Specific to data stores
  - Have a generally accepted basic structure

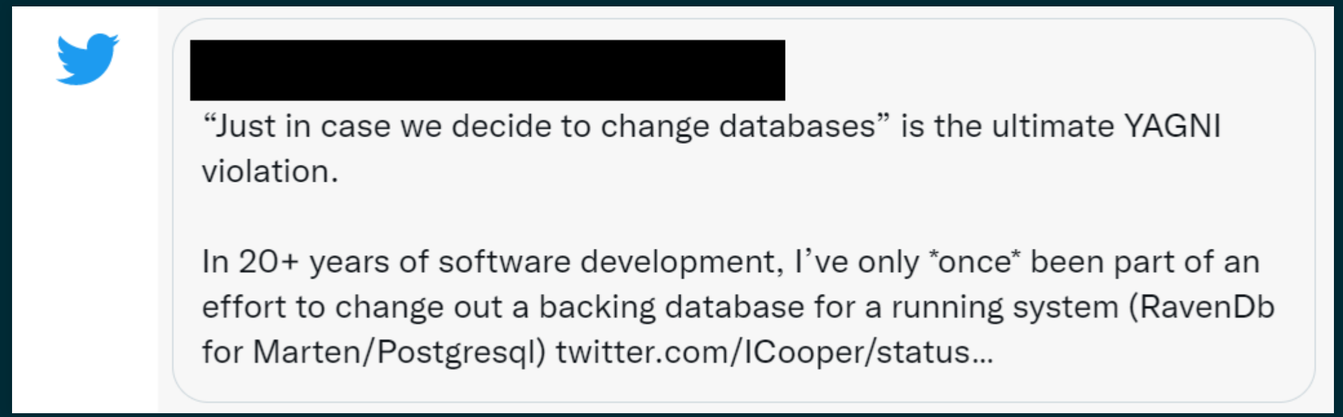
READ REPOSITORY

- IMeetingReadRepository
  - GetAllMeetings()
  - GetMeetingById(Id)
  - GetMeetingsByDateRange(startDate, endDate)

WRITE REPOSITORY

- IMeetingWriteRepository
  - SaveMeeting(meeting)
  - SaveMeeting(Id, property1, property2, ...)

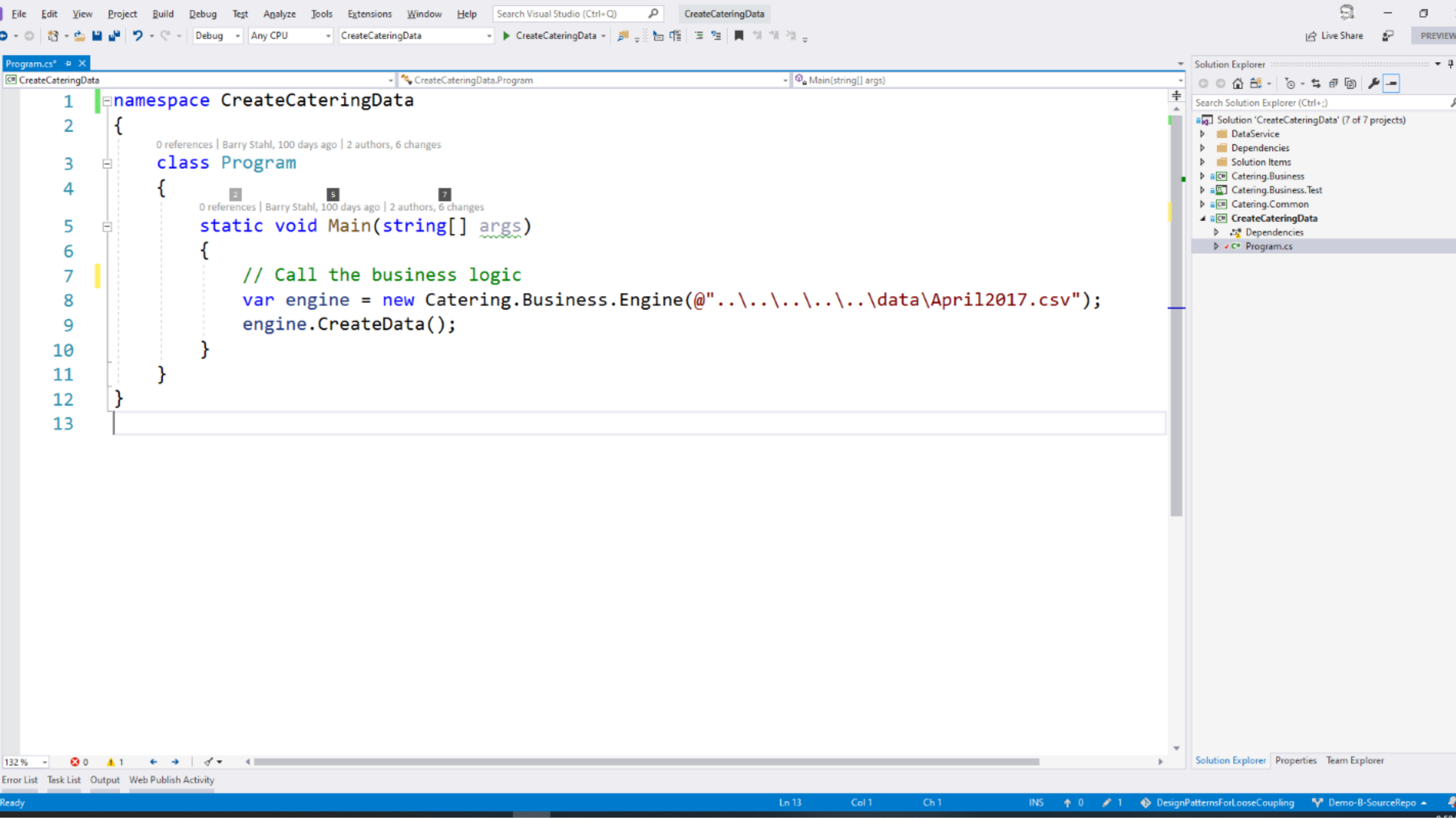
DO WE REALLY CHANGE DATA STORES?

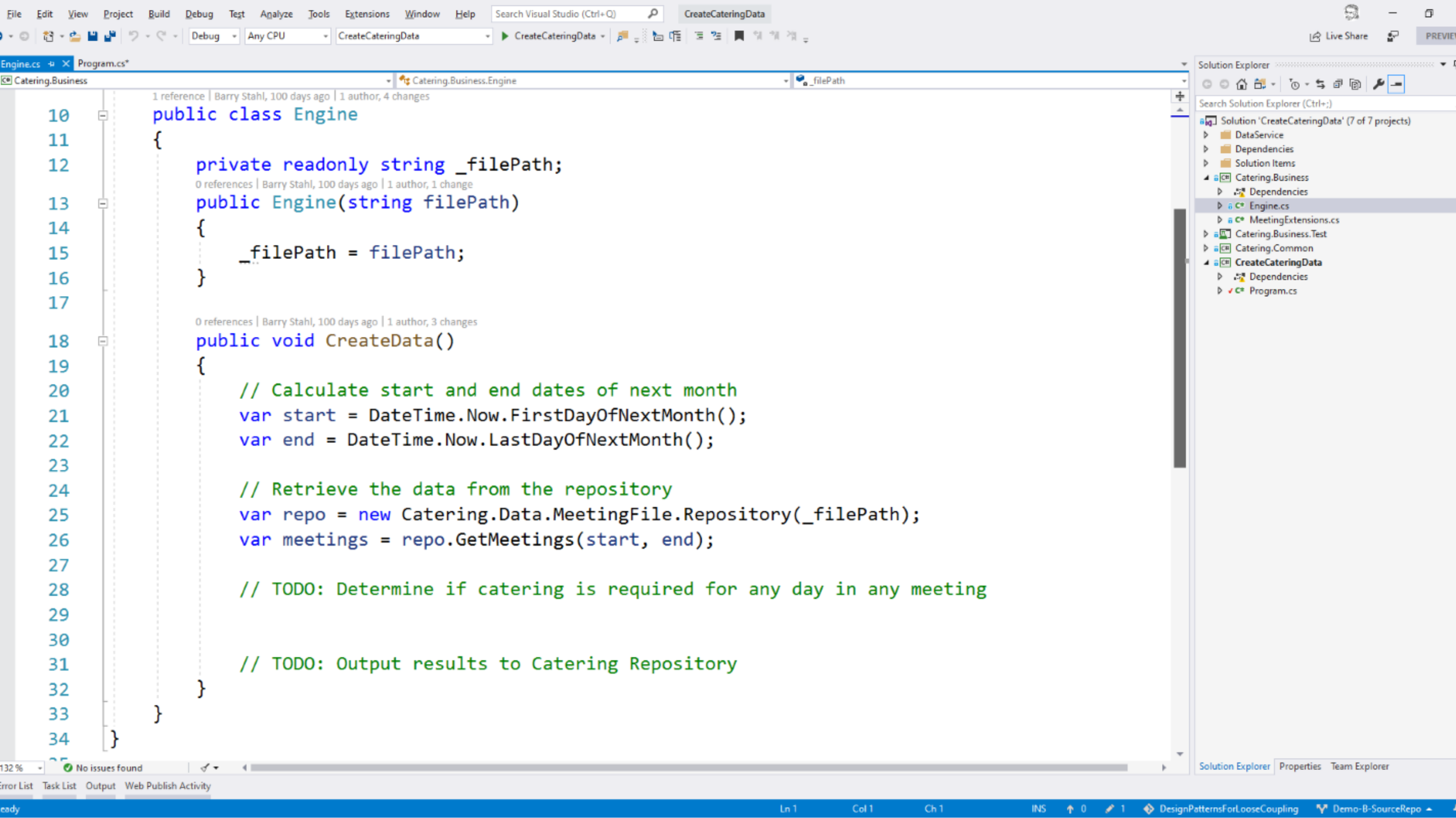


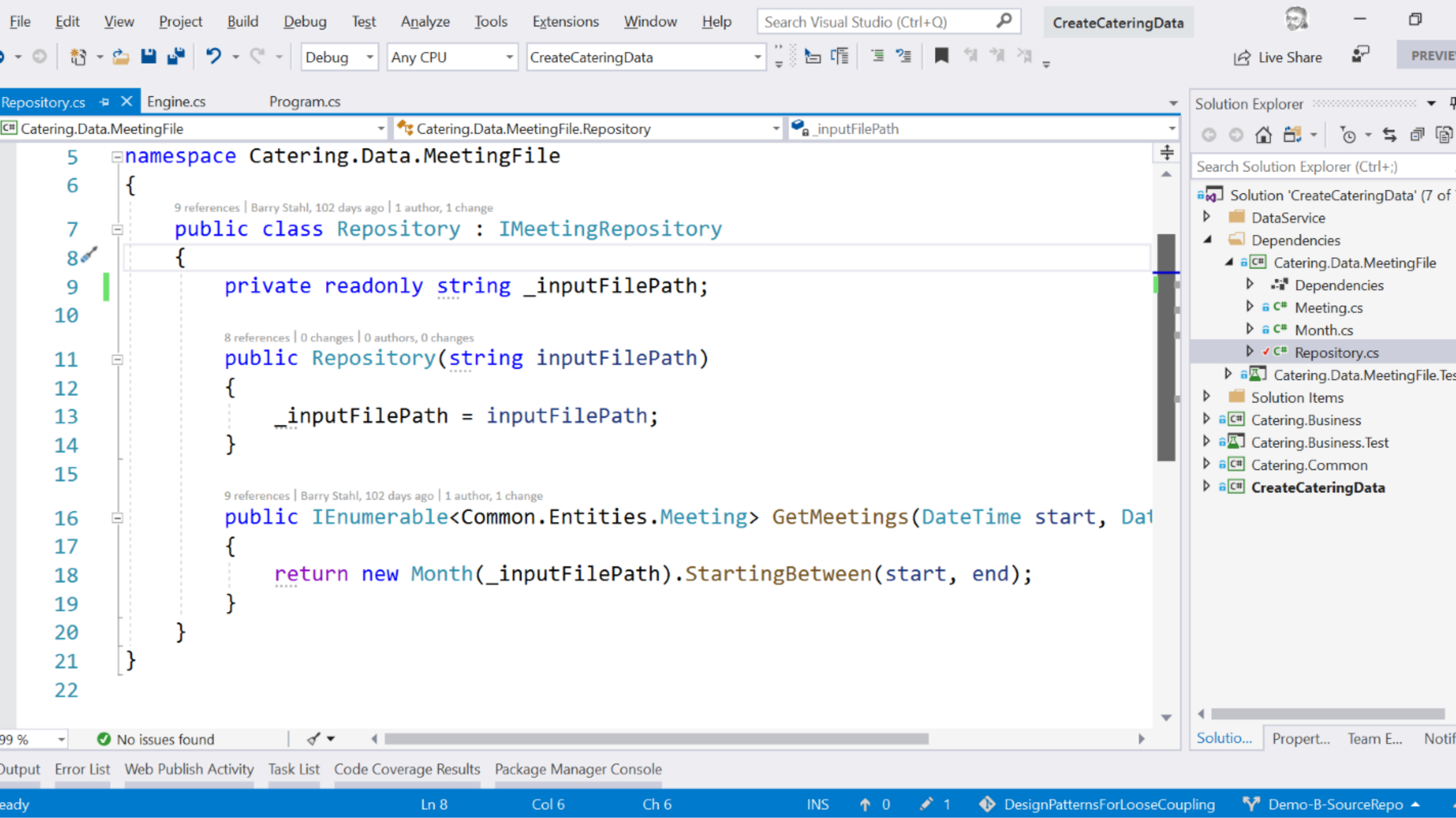
WE REALLY DO CHANGE DATA STORES

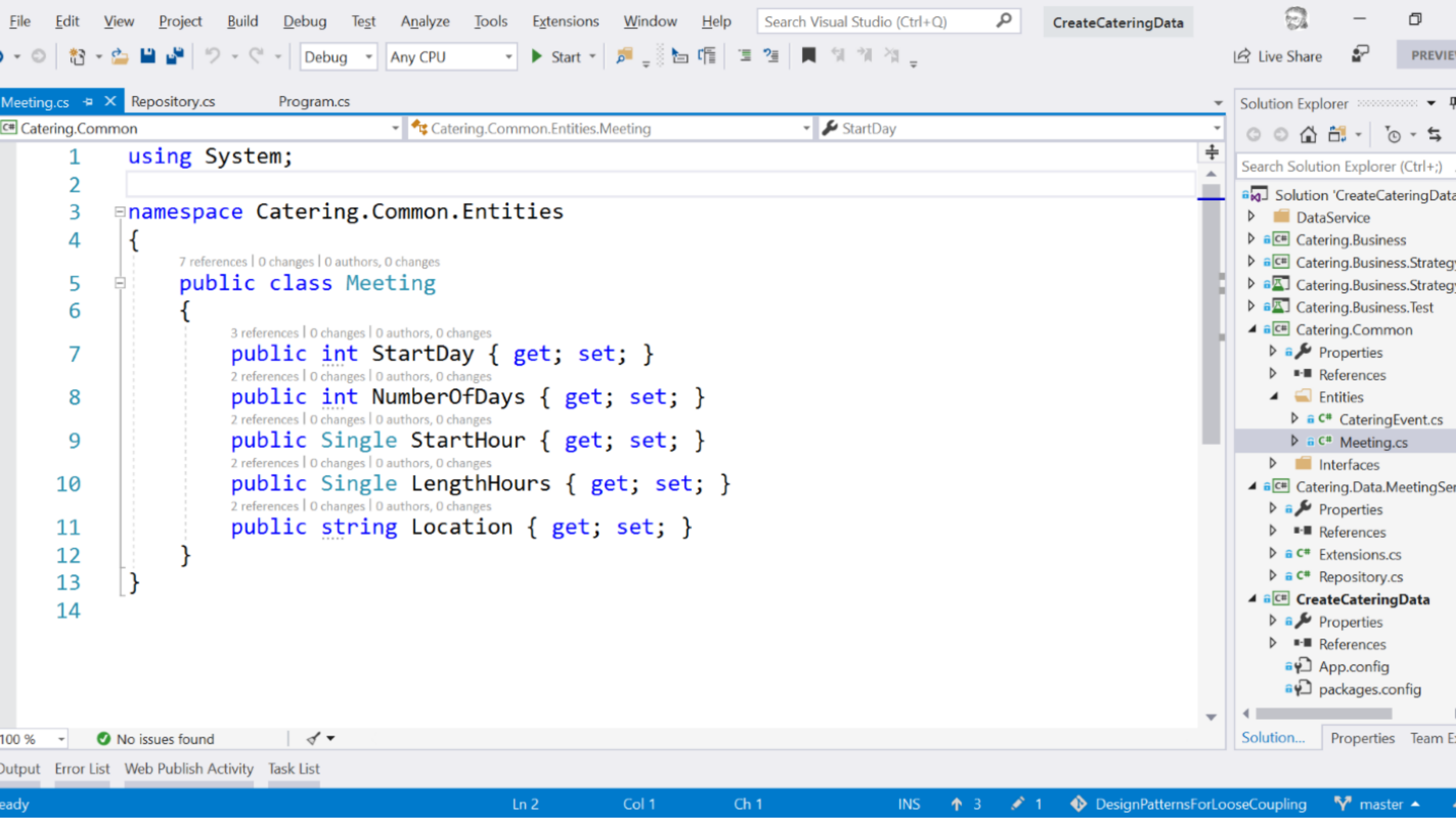
- You have changed data stores if:
  - You tested middle-tier logic in isolation (without testing the DB)
  - You changed from a file-source to a relational store or web-service
  - You added an API layer in front of an existing DB
  - You added a caching layer in front of a data source
  - You moved from a local DB to a cloud implementation

```
1  public interface IMeetingReadRepository
2  {
3      IEnumerable<Meeting> GetAllMeetings();
4  }
5
6  public class FileMeetingReadRepository : IMeetingReadRepository
7  {
8      private string _sourceFilePath;
9
10     public FileMeetingReadRepository(string sourceFilePath)
11     {
12         _sourceFilePath = sourceFilePath;
13     }
14
15     public IEnumerable<Meeting> GetAllMeetings()
16     {
17         // TODO: Load the meetings from the file
18         throw new NotImplementedException();
19     }
20 }
```


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringData
Debug
Any CPU
CreateCateringData
CreateCateringData
Live Share
Program.cs*
CreateCateringData
CreateCateringData.Program
Main(string[] args)
namespace CreateCateringData
{
0 references | Barry Stahl, 100 days ago | 2 authors, 6 changes
class Program
{
0 references | Barry Stahl, 100 days ago | 2 authors, 6 changes
static void Main(string[] args)
{
// Call the business logic
var engine = new Catering.Business.Engine(@"..\..\..\..\..\data\April2017.csv");
engine.CreateData();
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringData' (7 of 7 projects)
DataService
Dependencies
Solution Items
Catering.Business
Catering.Business.Test
Catering.Common
CreateCateringData
Dependencies
Program.cs
132 %
0
1
Solution Explorer
Properties
Team Explorer
Error List
Task List
Output
Web Publish Activity
Ready
Ln 13
Col 1
Ch 1
INS
0
1
DesignPatternsForLooseCoupling
Demo-B-SourceRepo


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringData
Debug
Any CPU
CreateCateringData
Live Share
Engine.cs
Program.cs*
Catering.Business
Catering.Business.Engine
_filePath
1 reference | Barry Stahl, 100 days ago | 1 author, 4 changes
public class Engine
{
private readonly string _filePath;
0 references | Barry Stahl, 100 days ago | 1 author, 1 change
public Engine(string filePath)
{
_filePath = filePath;
}
0 references | Barry Stahl, 100 days ago | 1 author, 3 changes
public void CreateData()
{
// Calculate start and end dates of next month
var start = DateTime.Now.FirstDayOfNextMonth();
var end = DateTime.Now.LastDayOfNextMonth();
// Retrieve the data from the repository
var repo = new Catering.Data.MeetingFile.Repository(_filePath);
var meetings = repo.GetMeetings(start, end);
// TODO: Determine if catering is required for any day in any meeting
// TODO: Output results to Catering Repository
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringData' (7 of 7 projects)
DataService
Dependencies
Solution Items
Catering.Business
Dependencies
Engine.cs
MeetingExtensions.cs
Catering.Business.Test
Catering.Common
CreateCateringData
Dependencies
Program.cs
132 %
No issues found
Solution Explorer
Properties
Team Explorer
Error List
Task List
Output
Web Publish Activity
Ready
Ln 1
Col 1
Ch 1
INS
DesignPatternsForLooseCoupling
Demo-B-SourceRepo


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringData
Debug
Any CPU
CreateCateringData
Live Share
Repository.cs
Engine.cs
Program.cs
Catering.Data.MeetingFile
Catering.Data.MeetingFile.Repository
_inputFilePath
namespace Catering.Data.MeetingFile
{
9 references | Barry Stahl, 102 days ago | 1 author, 1 change
public class Repository : IMeetingRepository
{
private readonly string _inputFilePath;
8 references | 0 changes | 0 authors, 0 changes
public Repository(string inputFilePath)
{
_inputFilePath = inputFilePath;
}
9 references | Barry Stahl, 102 days ago | 1 author, 1 change
public IEnumerable<Common.Entities.Meeting> GetMeetings(DateTime start, Dat
{
return new Month(_inputFilePath).StartingBetween(start, end);
}
}
}
99 %
No issues found
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringData' (7 of
DataService
Dependencies
Catering.Data.MeetingFile
Dependencies
Meeting.cs
Month.cs
Repository.cs
Catering.Data.MeetingFile.Tes
Solution Items
Catering.Business
Catering.Business.Test
Catering.Common
CreateCateringData
Solutio...
Propert...
Team E...
Notif
Output
Error List
Web Publish Activity
Task List
Code Coverage Results
Package Manager Console
eady
Ln 8
Col 6
Ch 6
INS
0
1
DesignPatternsForLooseCoupling
Demo-B-SourceRepo


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringData
Debug
Any CPU
Start
Live Share
Meeting.cs
Repository.cs
Program.cs
Catering.Common
Catering.Common.Entities.Meeting
StartDay
using System;
namespace Catering.Common.Entities
{
7 references | 0 changes | 0 authors, 0 changes
public class Meeting
{
3 references | 0 changes | 0 authors, 0 changes
public int StartDay { get; set; }
2 references | 0 changes | 0 authors, 0 changes
public int NumberOfDays { get; set; }
2 references | 0 changes | 0 authors, 0 changes
public Single StartHour { get; set; }
2 references | 0 changes | 0 authors, 0 changes
public Single LengthHours { get; set; }
2 references | 0 changes | 0 authors, 0 changes
public string Location { get; set; }
}
}
100 %
No issues found
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringData
DataService
Catering.Business
Catering.Business.Strateg
Catering.Business.Strateg
Catering.Business.Test
Catering.Common
Properties
References
Entities
CateringEvent.cs
Meeting.cs
Interfaces
Catering.Data.MeetingSer
Properties
References
Extensions.cs
Repository.cs
CreateCateringData
Properties
References
App.config
packages.config
Solution...
Properties
Team E
Output
Error List
Web Publish Activity
Task List
eady
Ln 2
Col 1
Ch 1
INS
3
1
DesignPatternsForLooseCoupling
master

A LESS TIGHTLY COUPLED APPLICATION

- ~~Our App Violates **SRP**~~
  - Business logic and data storage can change details independently
- Our App *Still* Violates **DIP**
  - Knowledge of the data source is required to access the input data
  - Knowledge of the input implementation is required to create the output

DEPENDENCY INJECTION

An implementation of the Dependency Inversion Principle that provides the concrete implementations of abstract dependencies to client objects

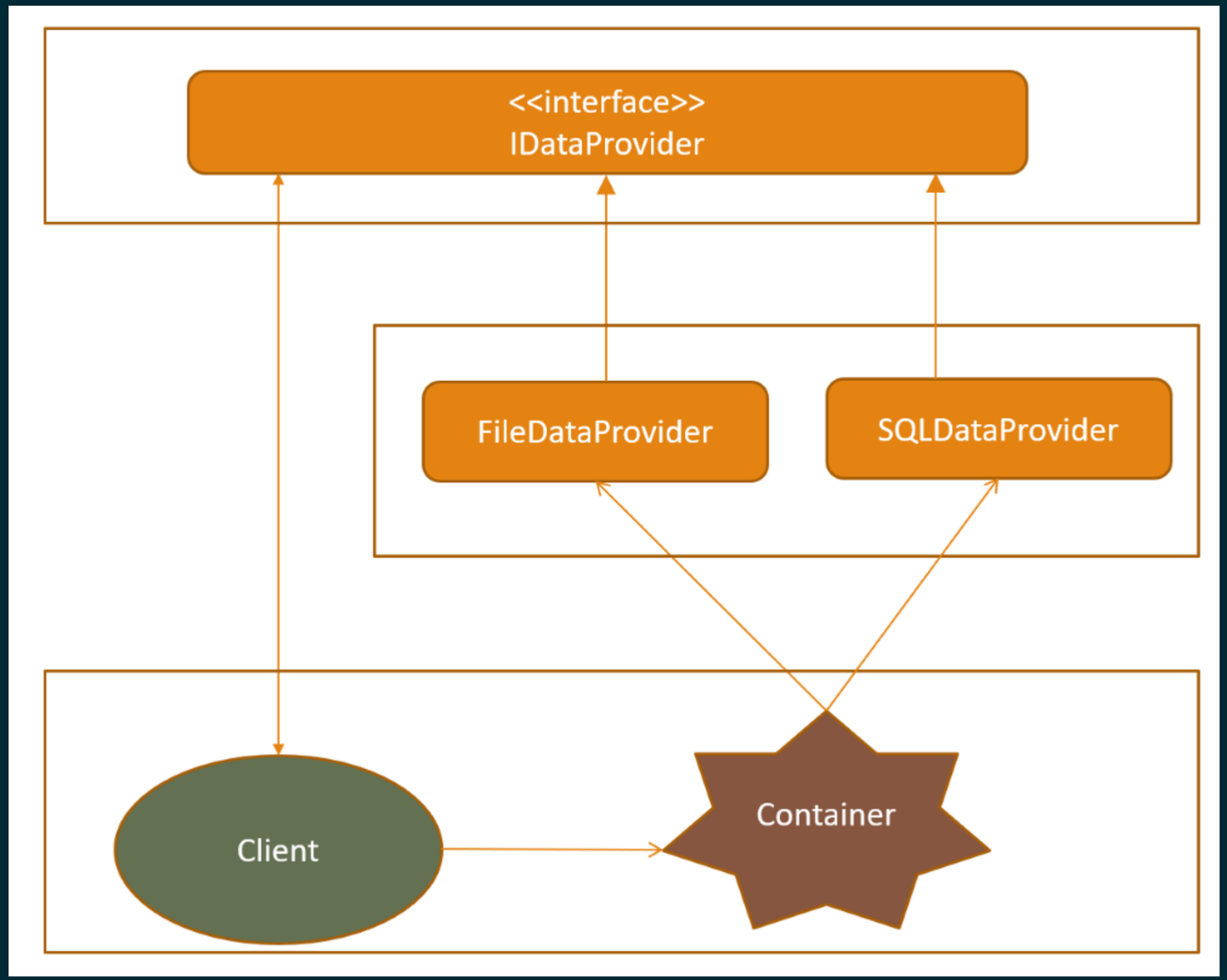
- Goals of Dependency Injection
  - Abstract the caller from knowledge of implementation details

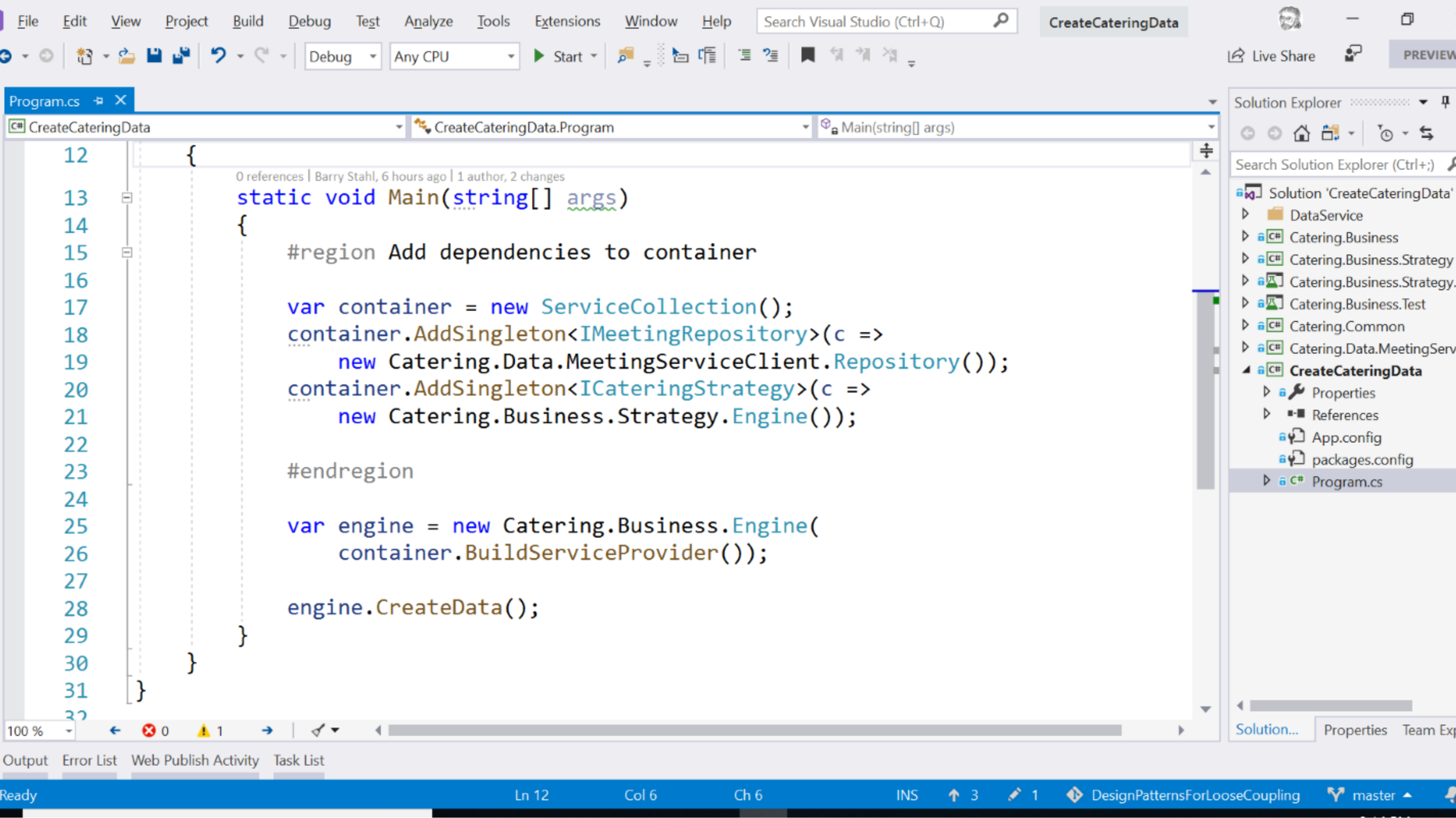
DEPENDENCY INJECTION FRAMEWORKS

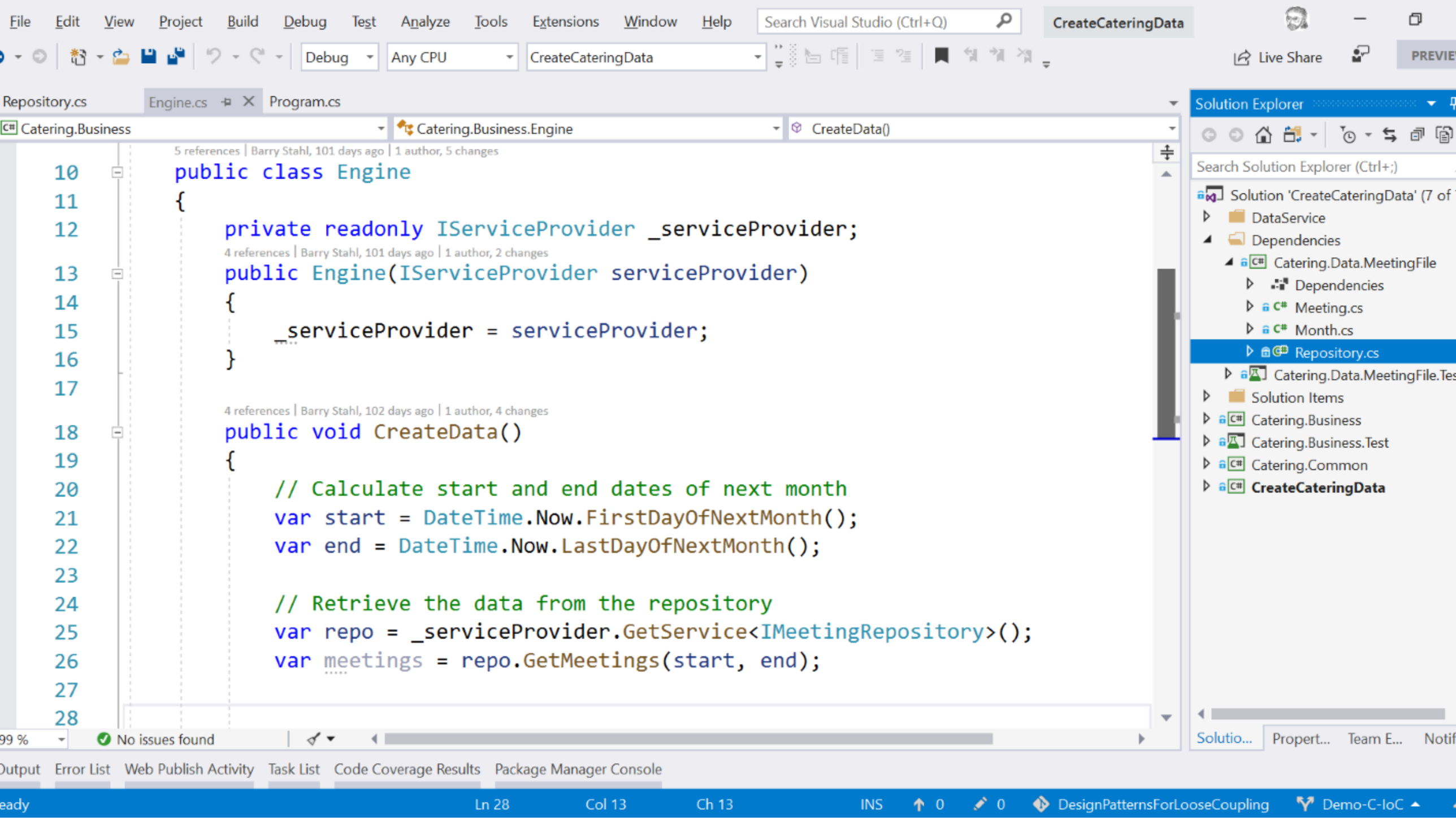
Tools for implementing Dependency Injection - Configured to hold the concrete implementations of dependencies

- Goals of DI Frameworks
  - Abstract the caller from knowledge of implementation details
  - Provide a well-known interface for *Service Location*
- Usage
  - Can be "wired-up" to automate dependency resolution
  - Can often be queried for a dependency by interface
  - Some containers will also manage the life-cycle of the dependencies

LOOSE COUPLING WITH DI



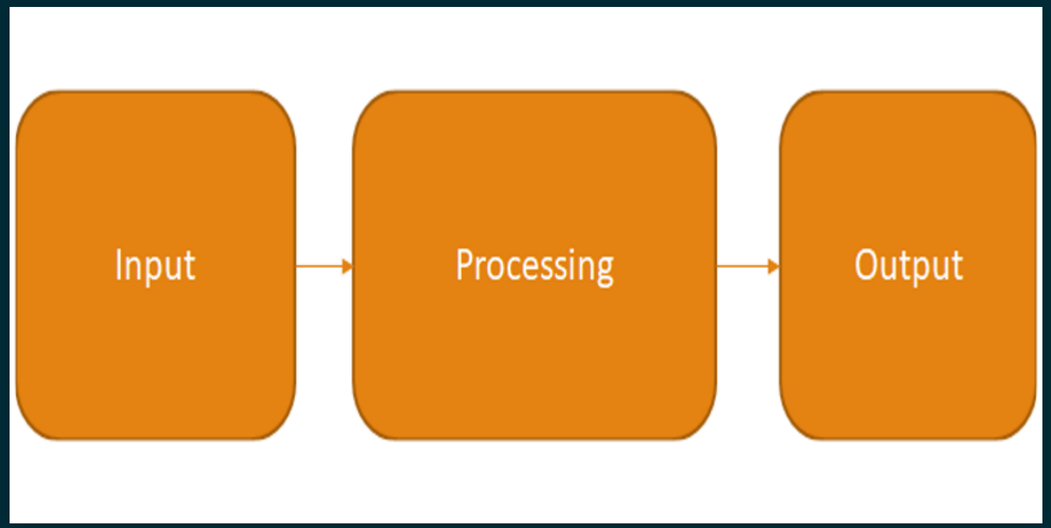

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringData
Debug
Any CPU
Start
Live Share
Program.cs
CreateCateringData
CreateCateringData.Program
Main(string[] args)
0 references | Barry Stahl, 6 hours ago | 1 author, 2 changes
static void Main(string[] args)
{
#region Add dependencies to container
var container = new ServiceCollection();
container.AddSingleton<IMeetingRepository>(c =>
new Catering.Data.MeetingServiceClient.Repository());
container.AddSingleton<ICateringStrategy>(c =>
new Catering.Business.Strategy.Engine());
#endregion
var engine = new Catering.Business.Engine(
container.BuildServiceProvider());
engine.CreateData();
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringData'
DataService
Catering.Business
Catering.Business.Strategy
Catering.Business.Strategy.
Catering.Business.Test
Catering.Common
Catering.Data.MeetingServ
CreateCateringData
Properties
References
App.config
packages.config
Program.cs
100 %
0
1
Output
Error List
Web Publish Activity
Task List
Ready
Ln 12
Col 6
Ch 6
INS
3
1
DesignPatternsForLooseCoupling
master


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringData
Debug
Any CPU
CreateCateringData
Live Share
Repository.cs
Engine.cs
Program.cs
Catering.Business
Catering.Business.Engine
CreateData()
5 references | Barry Stahl, 101 days ago | 1 author, 5 changes
public class Engine
{
private readonly IServiceProvider _serviceProvider;
4 references | Barry Stahl, 101 days ago | 1 author, 2 changes
public Engine(IServiceProvider serviceProvider)
{
_serviceProvider = serviceProvider;
}
4 references | Barry Stahl, 102 days ago | 1 author, 4 changes
public void CreateData()
{
// Calculate start and end dates of next month
var start = DateTime.Now.FirstDayOfNextMonth();
var end = DateTime.Now.LastDayOfNextMonth();
// Retrieve the data from the repository
var repo = _serviceProvider.GetService<IMeetingRepository>();
var meetings = repo.GetMeetings(start, end);
99 %
No issues found
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringData' (7 of
DataService
Dependencies
Catering.Data.MeetingFile
Dependencies
Meeting.cs
Month.cs
Repository.cs
Catering.Data.MeetingFile.Tes
Solution Items
Catering.Business
Catering.Business.Test
Catering.Common
CreateCateringData
Solutio...
Propert...
Team E...
Notif
Output
Error List
Web Publish Activity
Task List
Code Coverage Results
Package Manager Console
eady
Ln 28
Col 13
Ch 13
INS
0
0
DesignPatternsForLooseCoupling
Demo-C-IoC

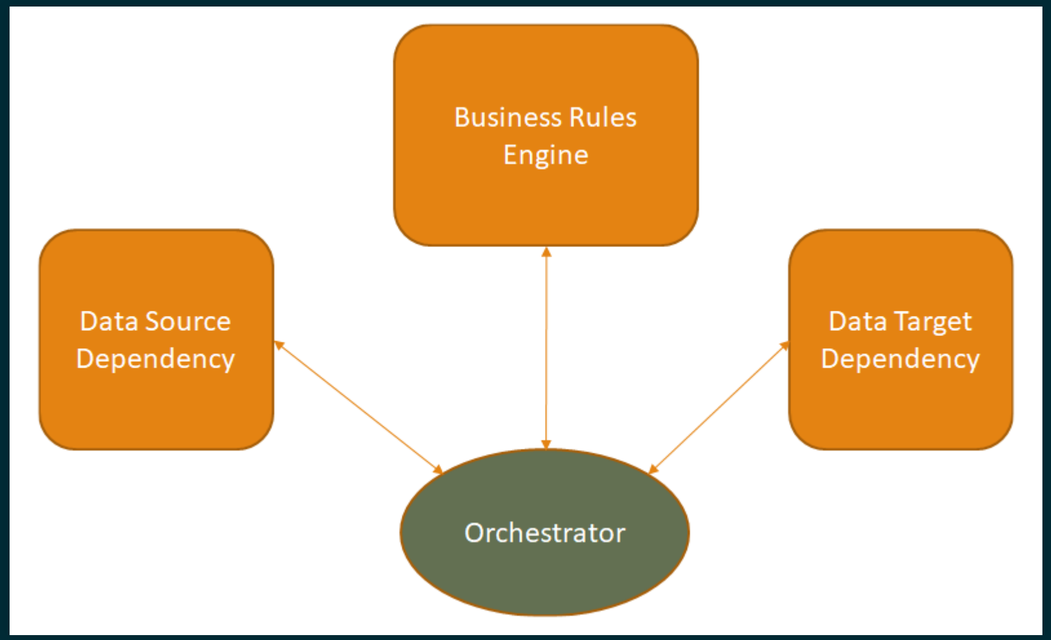
A FAIRLY LOOSELY COUPLED APPLICATION

- ~~Our App Violates **SRP**~~
  - Business logic and data storage can change independently
- ~~Our App Violates **DIP**~~
  - No knowledge of how data is stored is required to access the data
- We now have a new set of mixed concerns (SRP violation)
  - Our business logic is mixed-in with our orchestration logic

A TRADITIONAL ARCHITECTURE



BUSINESS VS ORCHESTRATION LOGIC



THE STRATEGY PATTERN

An implementation of a facade that fronts an algorithm or other logic

- Goals of a Strategy
  - Abstract the caller from the underlying implementation
  - Simplify the API for the caller
- Differences from other Facades
  - Specific to logic
  - Are less likely to contain side-effects (more functional)

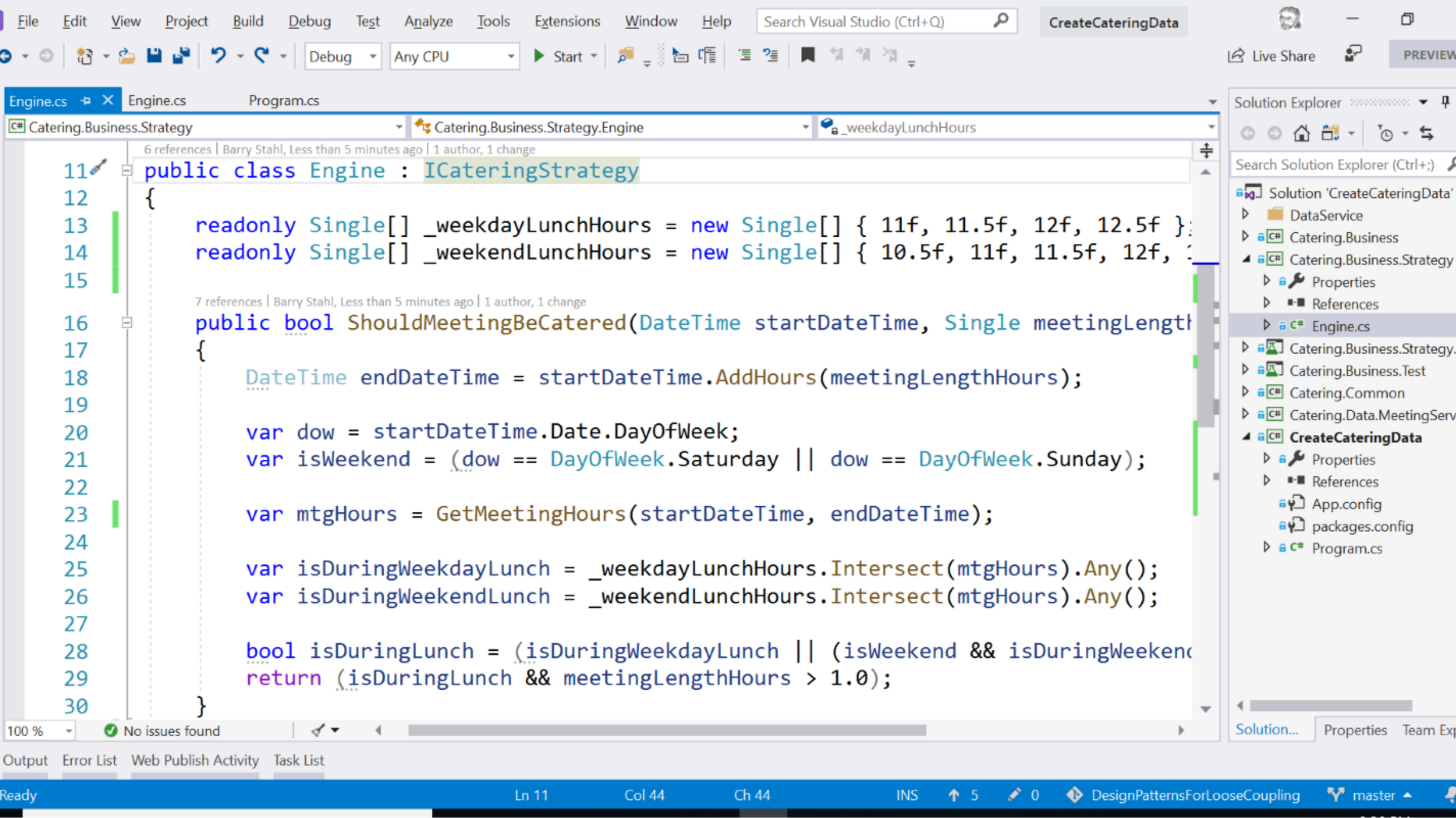
THE STRATEGY PATTERN ANALOGY

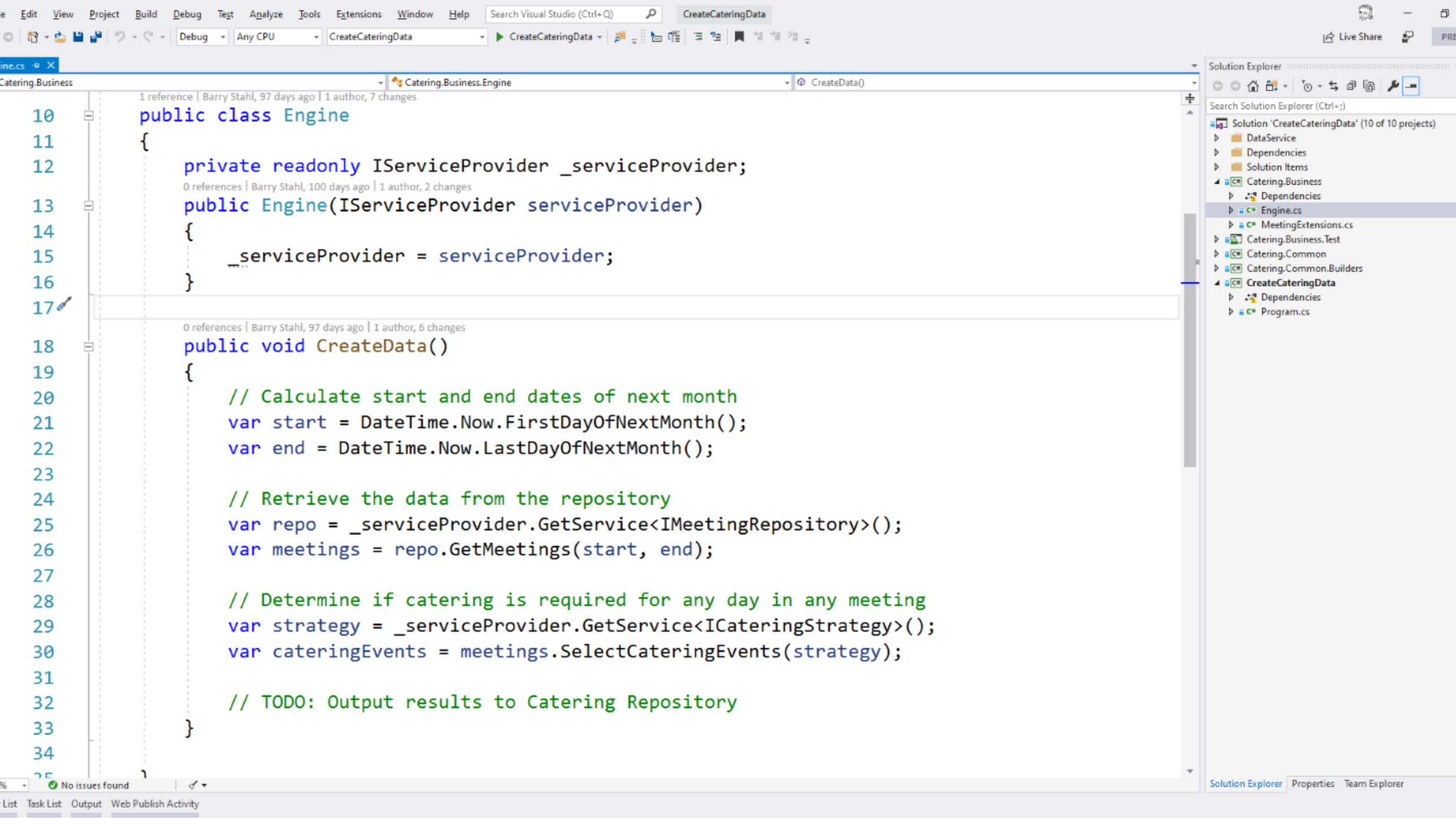
Barry Stahl - Co-Founder of AZGiveC...
@bsstahl

The Strategy Pattern is to algorithms what the Repository Pattern is to data stores, a useful and well-known abstraction for loose-coupling.

8:52 AM - 6 Jan 2017 from Phoenix, AZ

```
1  public interface ICateringStrategy
2  {
3      bool ShouldMeetingBeCatered(DateTimeOffset start, TimeSpan length);
4  }
5
6  public class AlwaysCaterStrategy : ICateringStrategy
7  {
8      public bool ShouldMeetingBeCatered(DateTimeOffset start, TimeSpan length)
9      {
0          return true;
1      }
2  }
```


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringData
Debug
Any CPU
Start
Live Share
Engine.cs
Engine.cs
Program.cs
Catering.Business.Strategy
Catering.Business.Strategy.Engine
_weekdayLunchHours
6 references | Barry Stahl, Less than 5 minutes ago | 1 author, 1 change
public class Engine : ICateringStrategy
{
readonly Single[] _weekdayLunchHours = new Single[] { 11f, 11.5f, 12f, 12.5f };
readonly Single[] _weekendLunchHours = new Single[] { 10.5f, 11f, 11.5f, 12f,
7 references | Barry Stahl, Less than 5 minutes ago | 1 author, 1 change
public bool ShouldMeetingBeCatered(DateTime startDateTime, Single meetingLengt
{
DateTime endDateTime = startDateTime.AddHours(meetingLengthHours);
var dow = startDateTime.Date.DayOfWeek;
var isWeekend = (dow == DayOfWeek.Saturday || dow == DayOfWeek.Sunday);
var mtgHours = GetMeetingHours(startDateTime, endDateTime);
var isDuringWeekdayLunch = _weekdayLunchHours.Intersect(mtgHours).Any();
var isDuringWeekendLunch = _weekendLunchHours.Intersect(mtgHours).Any();
bool isDuringLunch = (isDuringWeekdayLunch || (isWeekend && isDuringWeeken
return (isDuringLunch && meetingLengthHours > 1.0);
}
100 %
No issues found
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringData'
DataService
Catering.Business
Catering.Business.Strategy
Properties
References
Engine.cs
Catering.Business.Strategy.
Catering.Business.Test
Catering.Common
Catering.Data.MeetingServ
CreateCateringData
Properties
References
App.config
packages.config
Program.cs
Solution...
Properties
Team Ex
Output
Error List
Web Publish Activity
Task List
Ready
Ln 11
Col 44
Ch 44
INS
5
0
DesignPatternsForLooseCoupling
master


Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
CreateCateringData
Debug
Any CPU
CreateCateringData
Live Share
ine.cs
Catering.Business
Catering.Business.Engine
CreateData()
1 reference | Barry Stahl, 97 days ago | 1 author, 7 changes
public class Engine
{
private readonly IServiceProvider _serviceProvider;
0 references | Barry Stahl, 100 days ago | 1 author, 2 changes
public Engine(IServiceProvider serviceProvider)
{
_serviceProvider = serviceProvider;
}
0 references | Barry Stahl, 97 days ago | 1 author, 6 changes
public void CreateData()
{
// Calculate start and end dates of next month
var start = DateTime.Now.FirstDayOfNextMonth();
var end = DateTime.Now.LastDayOfNextMonth();
// Retrieve the data from the repository
var repo = _serviceProvider.GetService<IMeetingRepository>();
var meetings = repo.GetMeetings(start, end);
// Determine if catering is required for any day in any meeting
var strategy = _serviceProvider.GetService<ICateringStrategy>();
var cateringEvents = meetings.SelectCateringEvents(strategy);
// TODO: Output results to Catering Repository
}
No issues found
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CreateCateringData' (10 of 10 projects)
DataService
Dependencies
Solution Items
Catering.Business
Dependencies
Engine.cs
MeetingExtensions.cs
Catering.Business.Test
Catering.Common
Catering.Common.Builders
CreateCateringData
Dependencies
Program.cs
Solution Explorer
Properties
Team Explorer
Task List
Output
Web Publish Activity

WHAT HAVE WE DONE?

- Data Source => Repository Pattern
- Business Rules => Strategy Pattern
- Abstractions => Dependency Injection
- What about the Data Target?

YOUR MISSION

> *Implement a target data store using the Repository and Dependency Injection patterns*

- You can implement using:
  - File-system output
  - A relational database
  - A web-service

RESOURCES

- Demos & Articles
  - This Slide Deck: https://DesignPatternsForLooseCoupling.AzureWebsites.net
  - My Blog: https://www.CognitiveInheritance.com
  - Code Samples: https://www.github.com/bsstahl/DesignPatternsForLooseCoupling
- Patterns
  - DI: https://msdn.microsoft.com/en-us/library/hh323705(v=vs.100).aspx
  - Repository: https://msdn.microsoft.com/en-us/library/ff649690.aspx
- Tooling
  - MS DI: https://www.nuget.org/packages/Microsoft.Framework.DependencyInjection/
  - MOQ: https://github.com/Moq/moq4/wiki/Quickstart