# Creating Interfaces to Add Extensibility

The “How”

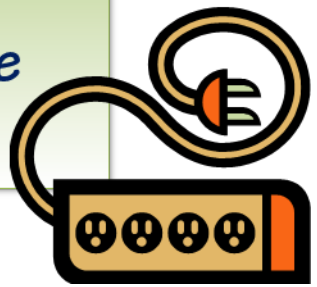
Jeremy Clark
www.jeremybytes.com
jeremy@jeremybytes.com

pluralsight
hardcore developer training

# Why Interfaces?

Maintainable

Extensible



Easily Testable


Interfaces help us get there

# Different Data Sources

- **Relational Databases**
  - Microsoft SQL Server, Oracle, MySQL, etc.
- **Document / Object Databases (NoSQL)**
  - MongoDB, Hadoop, RavenDB, etc.
- **Text Files**
  - CSV, XML, JSON, etc.
- **SOAP Services**
  - WCF, ASMX Web Service, Apache CXF, etc.
- **REST Services**
  - WebAPI, WCF, Apache CXF, JAX-RS, etc.
- **Cloud Storage**
  - Microsoft Azure, Amazon AWS, Google Cloud SQL

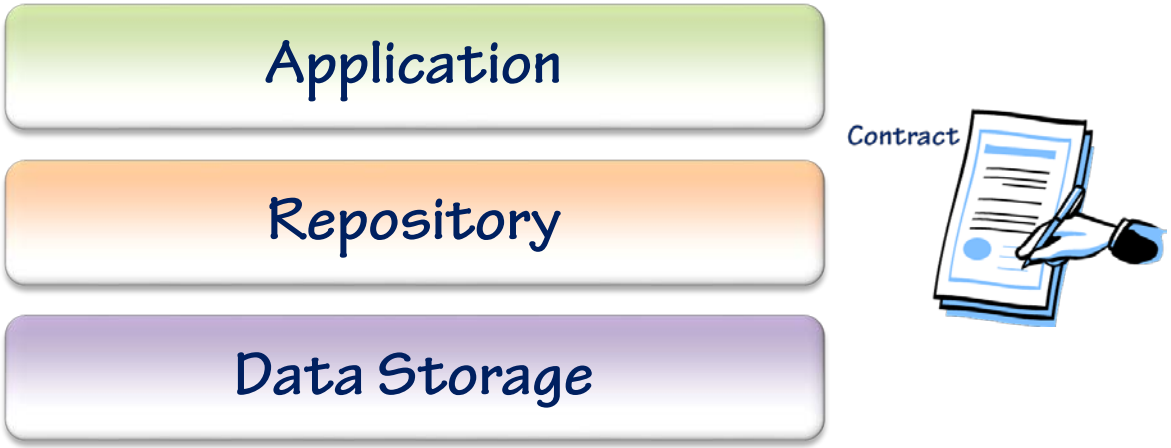
# Repository Pattern

## Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects*

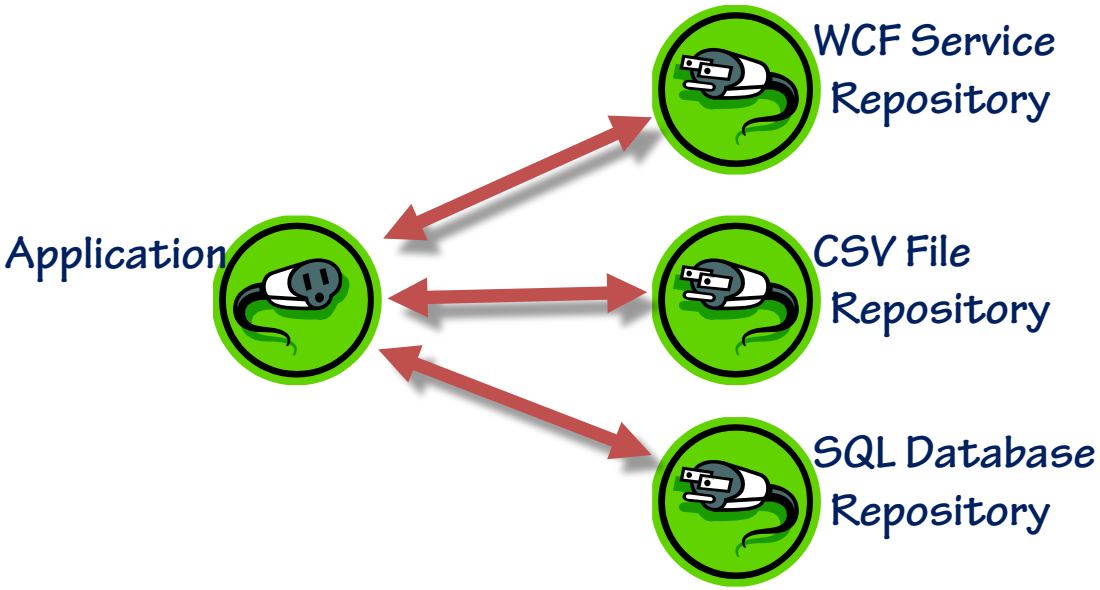
*Fowler, et al. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003.

# Repository Pattern

## Layer to separate our application from the data storage technology



# Pluggable Repositories



# Simple Repository

- **Data Access Operations**

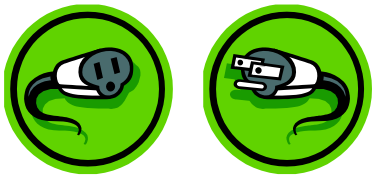
# Creating a Repository Interface

```
public interface IPersonRepository
{
C    void AddPerson(Person newPerson);

     IEnumerable<Person> GetPeople();
R
     Person GetPerson(string lastName);

     void UpdatePerson(string lastName,
          Person updatedPerson);
U
     void UpdatePeople(IEnumerable<Person>
          updatedPeople);

D    void DeletePerson(string lastName);
}
```

# Summary

- **Repository Pattern**


- **How to Create and Implement a Custom Interface**
  - IPersonRepository
- **Easy Extensibility**



- **Next up: Explicit Interface Implementation**