Loose Coupling

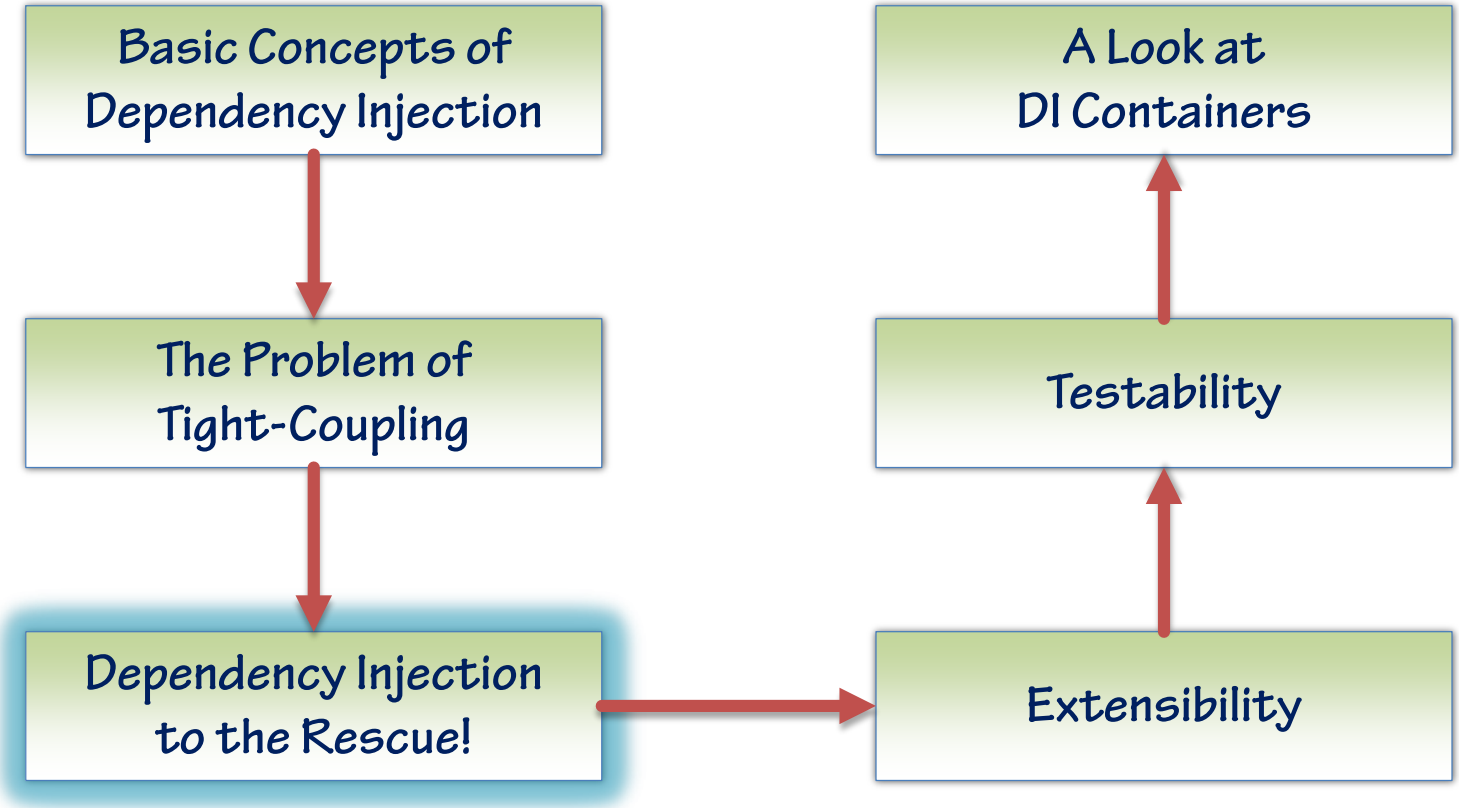
Breaking Tight Coupling with Dependency Injection

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

pluralsight
hardcore developer training

Goal

- Get Comfortable with Dependency Injection



View Model – Repository Relationship

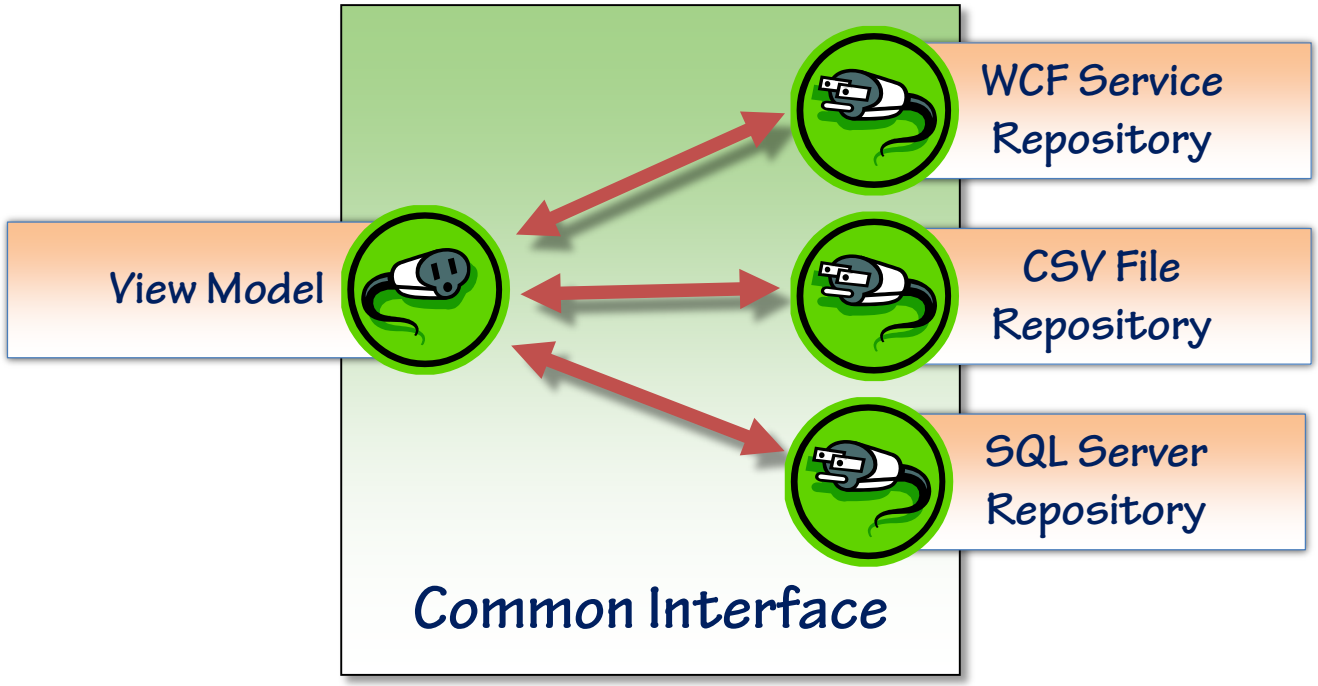
```
public class PeopleViewerViewModel : INotifyPropertyChanged
{
    protected ServiceRepository Repository;

    public PeopleViewerViewModel()
    {
        Repository = new ServiceRepository();
    }
    ...
}
```

- The View Model references a concrete type of Repository
- The View Model takes responsibility for creating and managing the Repository

Pluggable Repositories

- Accessing Different Data Sources



Creating a Repository Interface

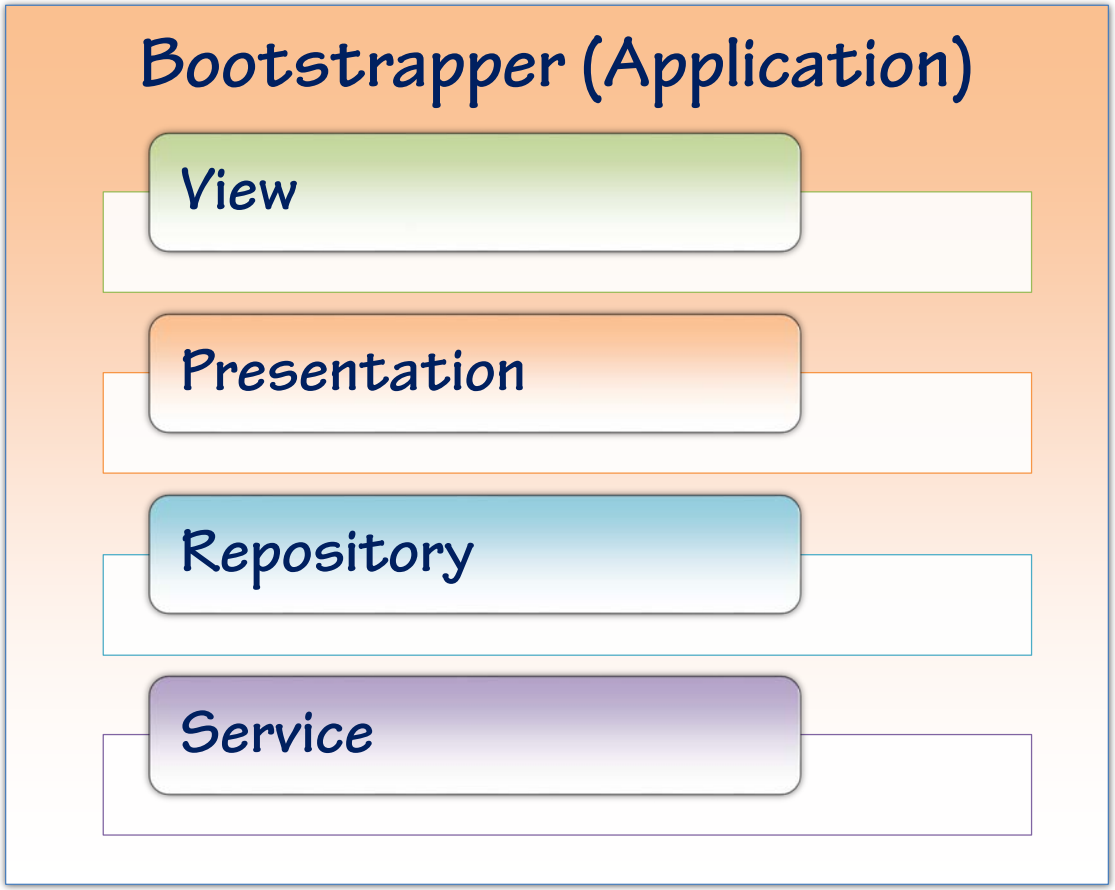
```
public interface IPersonRepository
{
C    void AddPerson(Person newPerson);

     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);
}
```

Composing the Application



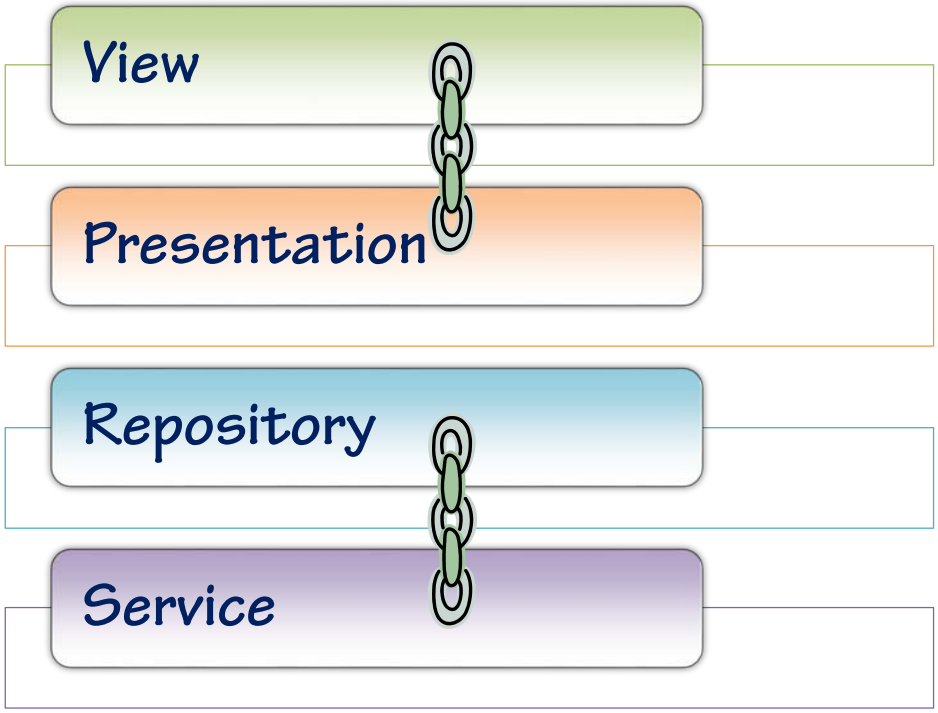
Constructor Injection

```
public class PeopleViewerViewModel : INotifyPropertyChanged
{
    protected IPersonRepository Repository;

    public PeopleViewerViewModel(IPersonRepository repository)
    {
        Repository = repository;
    }
    ...
}
```

- The View Model expects “Someone Else” to provide the Repository
- The View Model is no longer tied to a particular Repository

Loose(r) Coupling



Summary

- **Repository Interface**
- **Constructor Injection**
  - "Someone Else's Problem"
- **Composition Root**
  - Snapping things together
- **Bootstrapping the Application**

- **Next Up: Taking Advantage of Loose Coupling**
  Adding Repositories and Client-Side Caching