Creating Interfaces to Add Extensibility

Jeremy Clark
Developer Betterer

@jeremybytes www.jeremybytes.com

How

Create a repository interface

Implement the interface

- Web service
- Text file
- SQL database

Remove code duplication

Focus on important functionality

Extending Applications with Interfaces

✓ **Rules engine**

✓ **Custom workflow**

✓ **Authorization client**

Different Data Sources

Web service

Text file

SQL database

Document database

Cloud Service

Azure functions

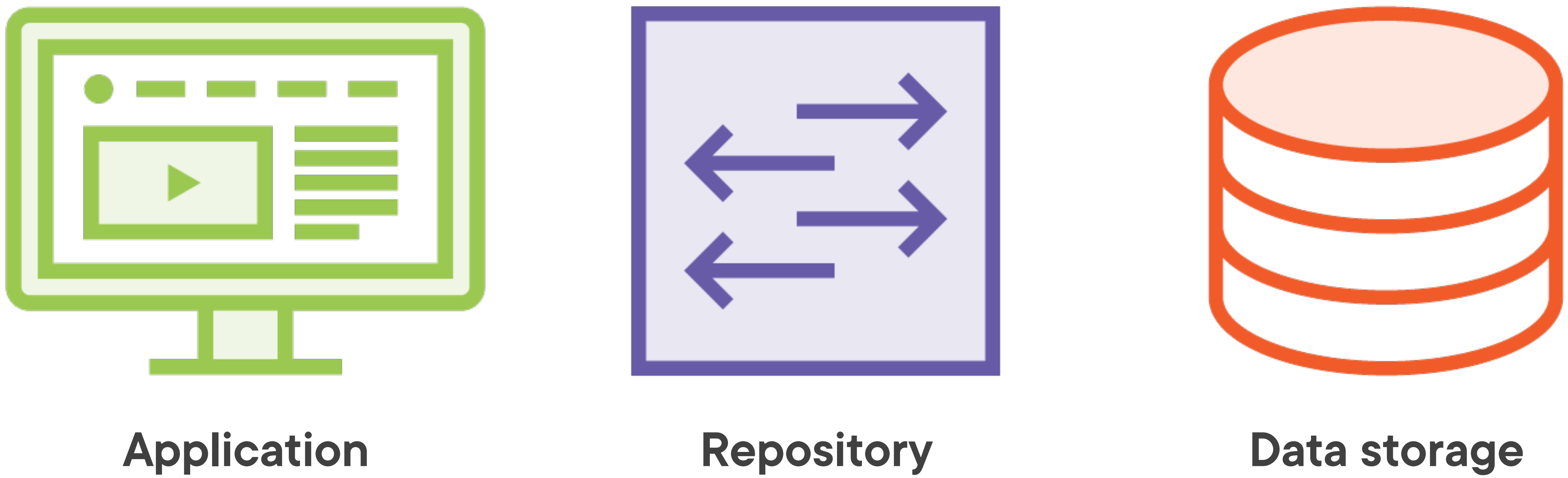
Repository Pattern

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

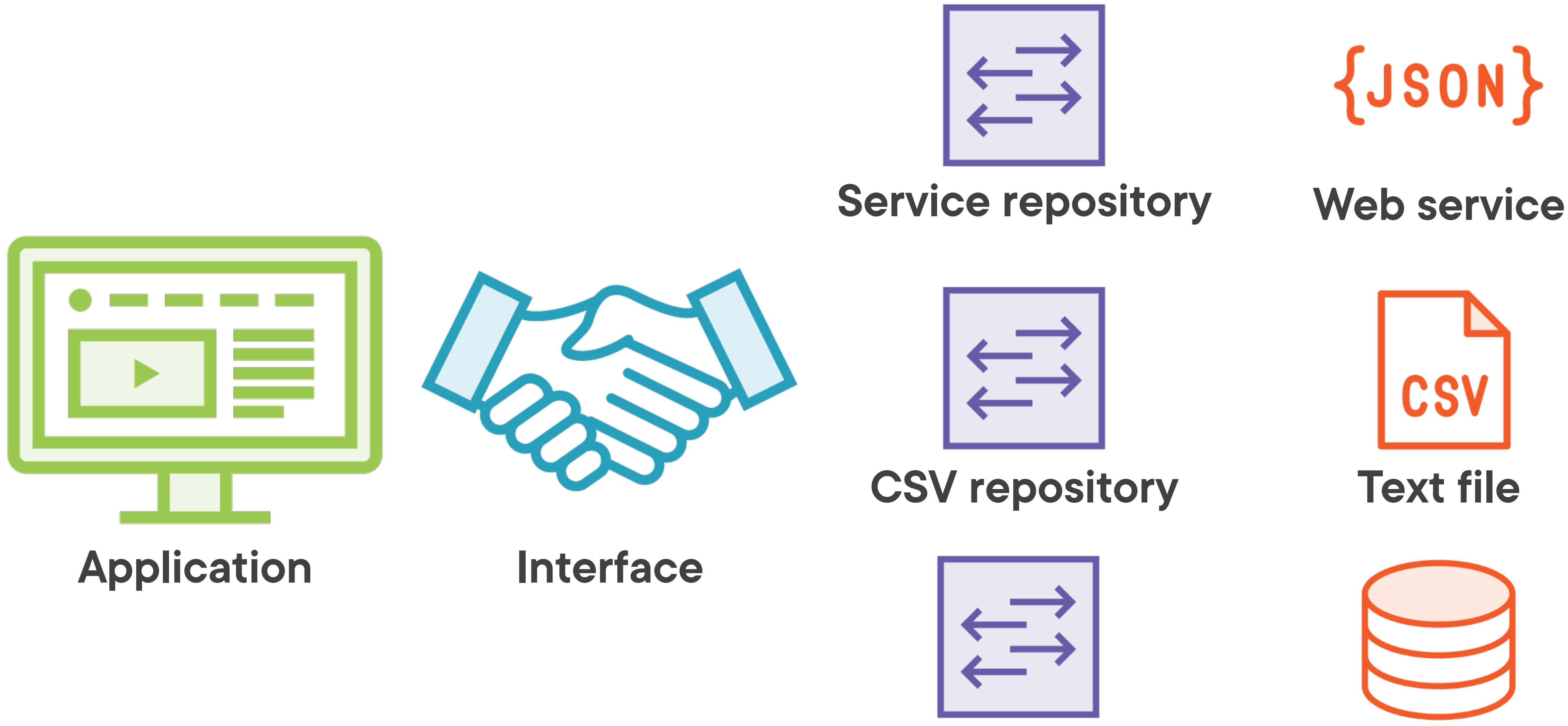
Fowler, et al. *Patterns of Enterprise Application Architecture.* Addison-Wesley, 2002.

Repository Pattern

Separates our application from the data storage technology



Repository Pattern (with an Interface)



Service repository

Web service

Application

Interface

CSV repository

Text file

SQL repository

SQL database

CRUD Repository

- Create
- Read
- Update
- Delete

Read-Only Client

```
private IActionResult PopulatePeopleView(string readerType)
{
  IPersonReader reader = readerFactory.GetReader(readerType);

  IEnumerable<Person> people = reader.GetPeople(); Read-only

  ViewData["ReaderType"] = reader.GetType().ToString();

  return View("Index", people);
}
```

CRUD Repository

~~Create~~

Read

~~Update~~

~~Delete~~

A Data Reader Interface

```
public interface IPersonReader
{
    IEnumerable<Person> GetPeople();
    Person GetPerson(int id);
}
```

Additional Resources

https://bit.ly/3tYeAee

https://github.com/jeremybytes/csharp-interfaces-resources

Includes how to run the samples with Visual Studio Code

Demo

Implementing an interface

Multiple data readers

- Service
- Text file
- SQL database

Remove duplication

Data Reader Factory

```
public IPersonReader GetReader(string readerType)
{
    switch (readerType)
    {
        case "Service": return new ServiceReader();
        case "CSV": return new CSVReader();
        case "SQL": return new SQLReader();
        default: throw new ArgumentException(...);
    };
}
```

Demo

Add factory method

Remove references to specific data readers

Application only knows the interface

No References to Specific Data Readers

```
private IActionResult PopulatePeopleView(string readerType)
{
  IPersonReader reader = readerFactory.GetReader(readerType);

  IEnumerable<Person> people = reader.GetPeople();

  ViewData["ReaderType"] = reader.GetType().ToString();

  return View("Index", people);
}
```

How

Create a repository interface

Implement the interface

- Web service
- Text file
- SQL database

Remove code duplication

Focus on important functionality