achieving system simplicity

me

I Am:

@jon_fuller

practicing apprentice

SEP

I Am Not:

expert

all-knowing

simplicity

flexible

maintainable

testable

agile

low coupling

DIP

IoC

OMG!

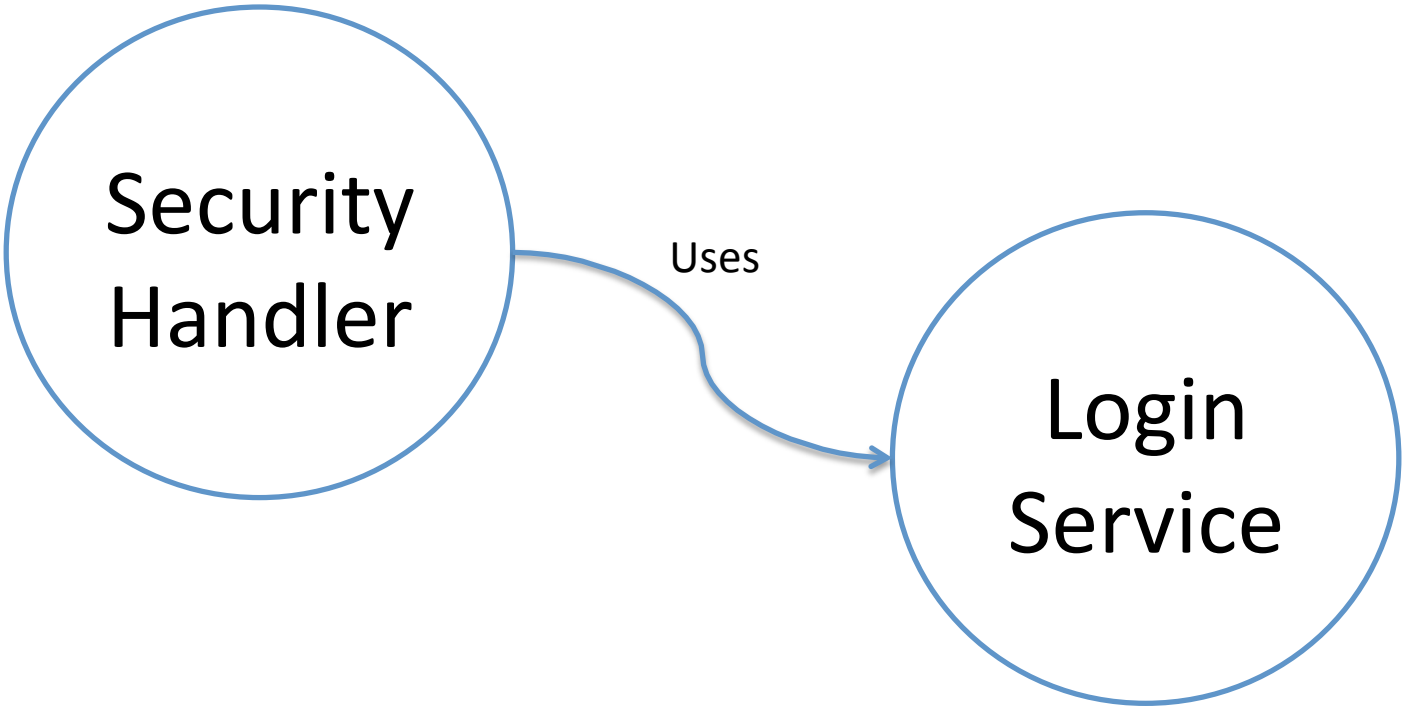
acronym overload

DI

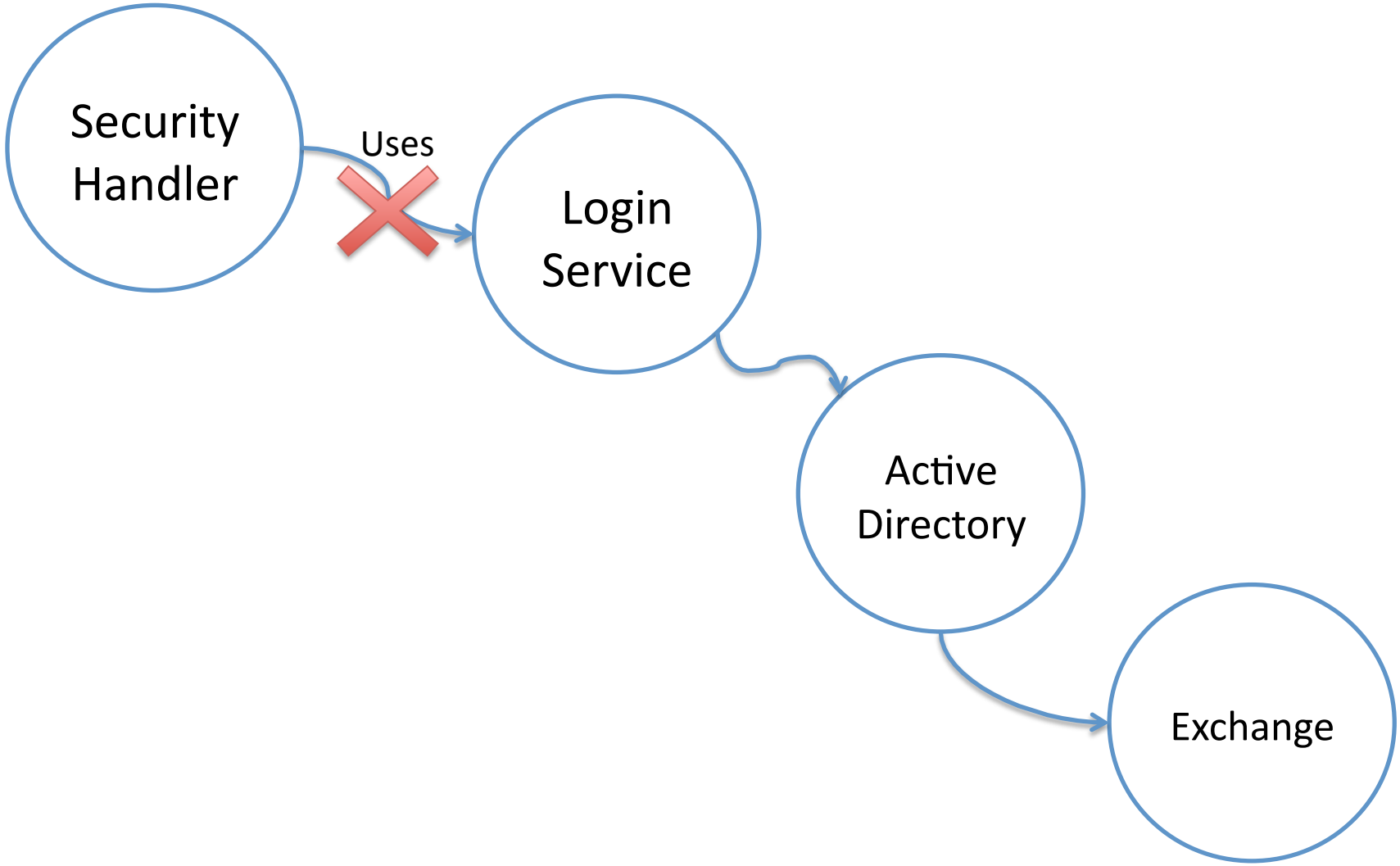
WTF?

Single Responsibility
Open-Closed
Liskov Substitution
Interface Segregation
Dependency Inversion

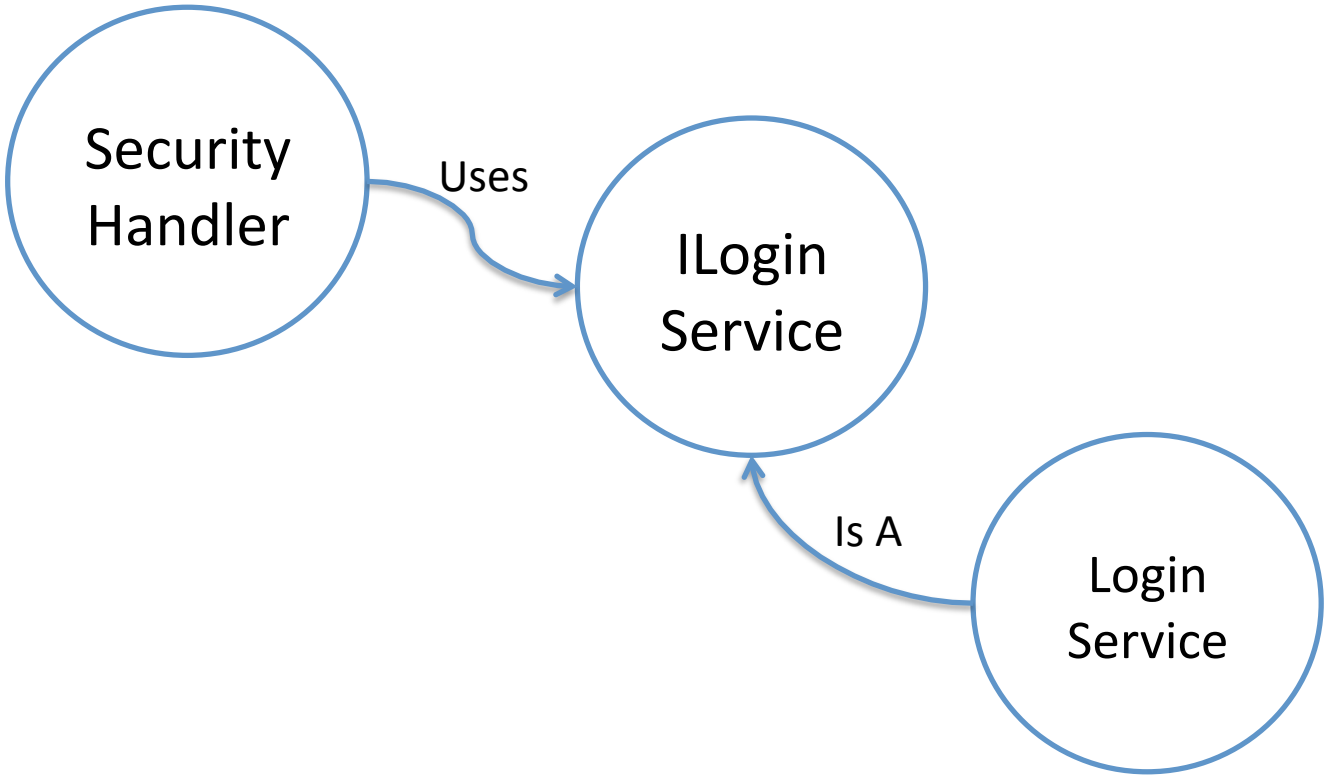
DIP



DIP

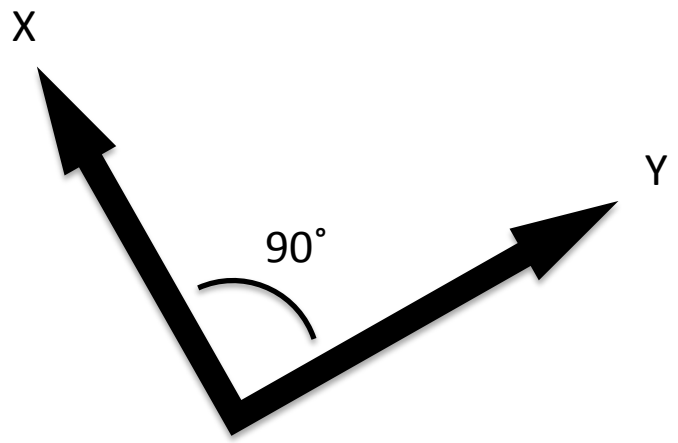

Security Handler
Uses
Login Service
Active Directory
Exchange

DIP



HIGH LEVEL MODULES SHOULD NOT DEPEND UPON LOW LEVEL MODULES . BOTH SHOULD DEPEND UPON ABSTRACTIONS.

DIP



DIP

service location

DI

```
public class MovieLister
{
    private readonly IMovieFinder _finder;

    public MovieLister()
    {
        _finder = ServiceLocator.Locate<IMovieFinder>();
    }

    public IEnumerable<Movie> MoviesDirectedBy(string director)
    {
        return _finder
                  .FindAll()
                  .Where(movie => movie.Director == director);
    }
}
```

dependency injection

DI

```
public class MovieLister
{
    private readonly IMovieFinder _finder;

    public MovieLister()
    {
        _finder = new ImdbMovieFinder();
    }

    public IEnumerable<Movie> MoviesDirectedBy(string director)
    {
        return _finder
                .FindAll()
                .Where(movie => movie.Director == director);
    }
}
```

```
public class MovieLister
{
    private readonly IMovieFinder _finder;

    public MovieLister()
    {
    }

    public IMovieFinder Finder
    {
        get;
        set;
    }

    public IEnumerable<Movie> MoviesDirectedBy(string director)
    {
        return _finder
                .FindAll()
                .Where(movie => movie.Director == director);
    }
}
```

DI

```
public class MovieLister
{
    private readonly IMovieFinder _finder;

    public MovieLister(IMovieFinder finder)
    {
        _finder = finder;
    }

    public IEnumerable<Movie> MoviesDirectedBy(string director)
    {
        return _finder
                  .FindAll()
                  .Where(movie => movie.Director == director);
    }
}
```

poor man’s DI

```
public class MovieLister
{
    private readonly IMovieFinder _finder;

    public MovieLister()
      : this(ServiceLocator.Locate<IMovieFinder>())
    {
    }

    public MovieLister(IMovieFinder finder)
    {
       _finder = finder;
    }

    public IEnumerable<Movie> MoviesDirectedBy(string director)
    {
        return _finder
                 .FindAll()
                 .Where(movie => movie.Director == director);
    }
}
```

DI

simplicity

WTF?

inversion of control

IoC

declarative

container

free your mind

```
public class Program
{
    public static void Main()
    {
        var lister = new MovieLister(new ImdbMovieFinder());

        lister.MoviesDirectedBy("Kubrick").Each(
            movie => Console.WriteLine(movie.ToString()));
    }
}
```

IoC

```
public class Program
{
    public static void Main()
    {
        ObjectFactory.Configure(cfg =>
        {
            cfg.For<IMovieFinder>().Use<ImbdMovieFinder>();
        });

        var lister = new MovieLister(ObjectFactory.GetInstance<IMovieFinder>());

        lister.MoviesDirectedBy("Kubrick").Each(
            movie => Console.WriteLine(movie.ToString()));
    }
}
```

IoC

```
public class Program
{
    public static void Main()
    {
        ObjectFactory.Configure(cfg =>
        {
            cfg.For<IMovieFinder>().Use<DatabaseMovieFinder>();
            cfg.For<IDbConnection>().Use<SqlConnection>()
                .Ctor<string>().Is("connection_string");
        });

        var lister = ObjectFactory.GetInstance<MovieLister>());

        lister.MoviesDirectedBy("Kubrick").Each(
            movie => Console.WriteLine(movie.ToString()));
    }
}
```

IoC

conventions

Java .NET

Dynamic

IoC

lifecycle management

```
public class Program
{
    public static void Main()
    {
        ObjectFactory.Configure(cfg =>
        {
            cfg.For<IMovieFinder>().Use<DatabaseMovieFinder>();
            cfg.For<IDbConnection>().Singleton().Use<SqlConnection>()
                .Ctor<string>().Is("connection_string");
        });

        var lister = ObjectFactory.GetInstance<MovieLister>());

        lister.MoviesDirectedBy("Kubrick").Each(
            movie => Console.WriteLine(movie.ToString()));
    }
}
```

IoC

aop/interception

```
public class Program
{
    public static void Main()
    {
        ObjectFactory.Configure(cfg =>
        {
            cfg.For<IMovieFinder>()
               .Use<DatabaseMovieFinder>()
               .InterceptWith(new LoggingInterceptor());

            cfg.For<IDbConnection>().Singleton().Use<SqlConnection>()
                .Ctor<string>().Is("connection_string");
        });

        var lister = ObjectFactory.GetInstance<MovieLister>());

        lister.MoviesDirectedBy("Kubrick").Each(
            movie => Console.WriteLine(movie.ToString()));
    }
}
```

IoC

simplicity

?

@jon_fuller
fullerjc@gmail.com
github.com/jonfuller

Feedback please!