Castle.Windsor
Castle.MicroKernel

IoC konteineris su konfigūravimo galimybėmis

http://www.castleproject.org/container/index.html

mailto:Darius.Damalakas@gmail.com

Inversion of Control(1)

Techniškai klasė x priklauso nuo Y jeigu:

- X turi Y ir naudoja ją
- X yra Y
- X priklauso nuo Z, kuri priklauso nuo Y (tranzityvumas)

IoC sprendžia klausimą “kaip yra patenkinamos objekto priklausomybės nuo kitų objektų”.

Standartiškai ryšiai yra “įprogramuojami”. Pvz X.Y = new Y().

mailto:Darius.Damalakas@gmail.com

Kaip IoC sprendžia priklausomybių problemą?

Objekto priklausomybės yra patenkinamos:

- Objekto kūrimo metu patenkinami privalomi ryšiai
- Po objekto kūrimo nustatomi neprivalomi ryšiai (per savybes / kintamuosius)

- Inversion of Control principas dar vadinamas Dependency Injection

mailto:Darius.Damalakas@gmail.com

Kodėl IoC?

- OOP stengiasi spręsti plečiamumo problemą.
- Kuo objektai labiau susiję (coupling), tuo mažesnis plečiamumas.
- Taigi tikslas – kuo didesnis plečiamumas, ir kuo “patogesnis” programavimas

mailto:Darius.Damalakas@gmail.com

IoC pavyzdys (1)

http://www.martinfowler.com/articles/injection.html

```
class MovieLister...
    public Movie[] moviesDirectedBy(String arg) {
        List allMovies = finder.findAll();
        for (Iterator it = allMovies.iterator(); it.hasNext();) {
            Movie movie = (Movie) it.next();
            if (!movie.getDirector().equals(arg)) it.remove();
        }
        return (Movie[]) allMovies.toArray(new Movie[allMovies.size()]);
    }
```


mailto:Darius.Damalakas@gmail.com

IoC pavyzdys (2)

Mūsų pavyzdžio tikslas yra padaryti moviesDirectedBy nepriklausoma nuo kur ir kaip yra saugomi filmai.

Tuo tikslu kuriame interfeisą, kuris grąžina sąrašą filmų. Tokiu būdu bandome atsiriboti nuo konkrečios implementacijos

- public interface MovieFinder {
- List findAll();
- }

mailto:Darius.Damalakas@gmail.com

IoC pavyzdys (3)

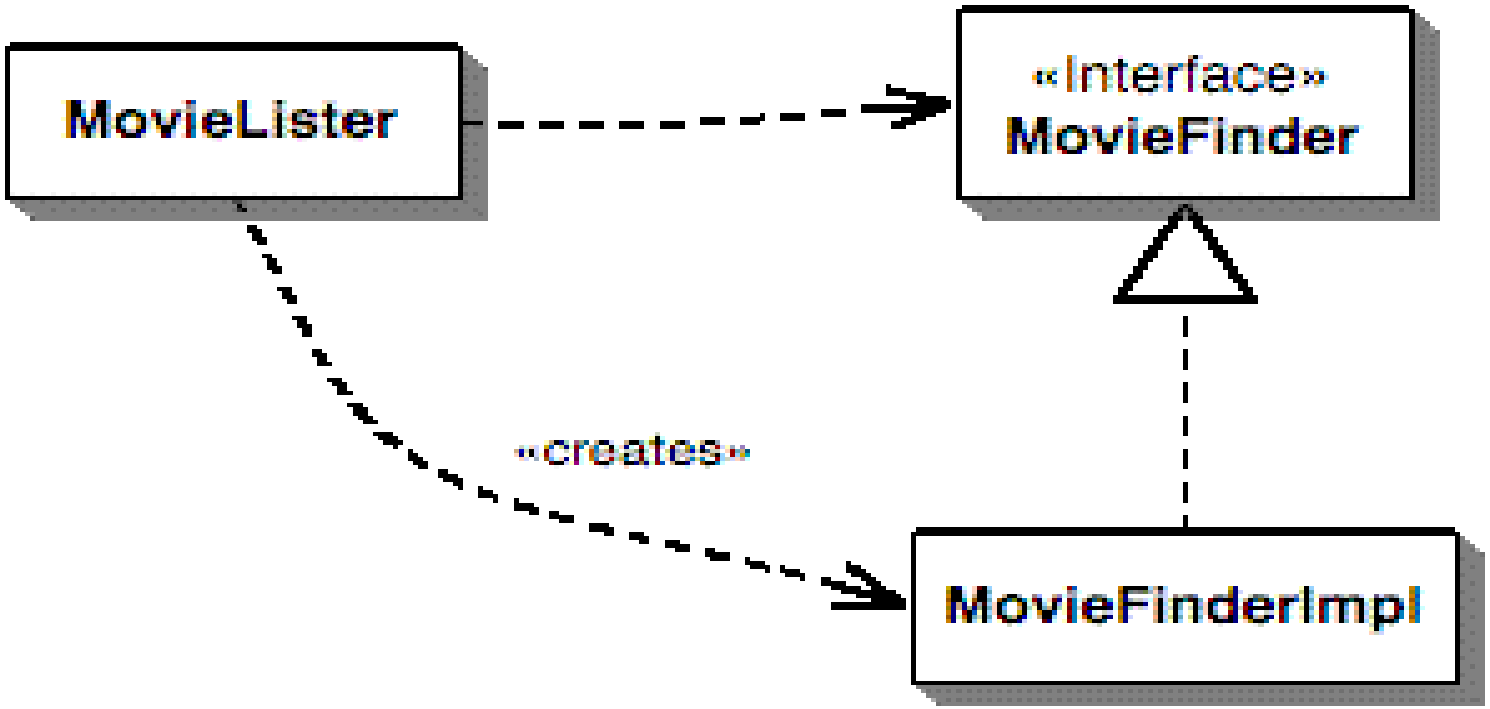
Matytame kode viskas gražiai atskirta (decoupled).

Deja, vienaip ar kitaip MovieLister turi sužinoti apie konkretų MovieFinder. Vienas iš būdų tai nustatyti yra per konstruktorių:

- class MovieLister...
- private MovieFinder finder;
- public MovieLister() {
- finder = new ColonDelimitedMovieFinder("movies1.txt");
- }

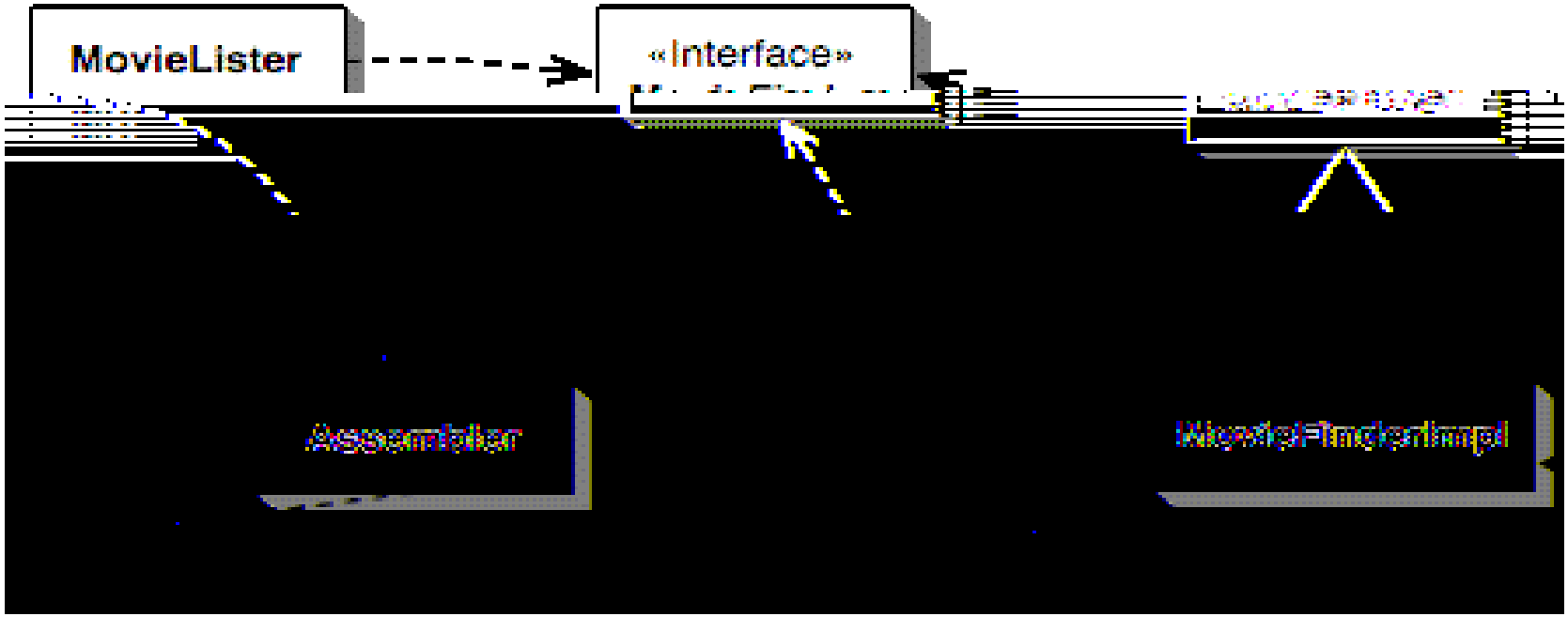
IoC pavyzdys (4)

- Rezultatas tas, kad MovieLister priklauso ir nuo interfeiso, ir nuo implementacijos.
- Kaip būtų galima to atsikratyti?



IoC pavyzdys (5)

- Kaip patenkinti objektų priklausomybes nedidinant surišimo (coupling)?
- Vienas iš būdų – naudoti Dependency Injection principą



Priklausomybių patenkinamas su Dependency Injection

```
private MutablePicoContainer configureContainer() {
    MutablePicoContainer pico = new DefaultPicoContainer();
    Parameter[] finderParams =  {new
                ConstantParameter("movies1.txt")};
    pico.registerComponentImplementation(MovieFinder.class,
                ColonMovieFinder.class, finderParams);
    pico.registerComponentImplementation(MovieLister.class);
    return pico;
}
```


mailto:Darius.Damalakas@gmail.com

Priklausomybių patenkinamas su Dependency Injection (2)

```
public void testWithPico() {
    MutablePicoContainer pico = configureContainer();
    MovieLister lister = (MovieLister)
            pico.getComponentInstance(MovieLister.class);
    Movie[] movies = lister.moviesDirectedBy("Sergio Leone");
    assertEquals("Once Upon a Time in the West",
            movies[0].getTitle());
}
```


mailto:Darius.Damalakas@gmail.com

Kas yra Castle.MicroKernel?

- Castle.MicroKernel yra IoC konteineris
- Sujungia servisus ir objektus, reikalaujančius tų servisų.

Kas yra Castle.Windsor?

- Castle.Windsor išplečia Castle.Microkernelį konfigūravimo galimybėmis
- Prideda galimybę registruoti “interceptors”

(detalizuosime vėliau).

mailto:Darius.Damalakas@gmail.com

Castle.Windsor naudojimo pavyzdys

Paprasta programa gali atrodyti taip:

- Application.Run(new Form1());

mailto:Darius.Damalakas@gmail.com

Castle.Windsor naudojimo pavyzdys (2)

Programa daro tą patį, bet formą gauname iš konteinerio:

- IWindsorContainer container = new WindsorContainer();
- Register the component
- container.AddComponent("form.component", typeof(Form1));
- // Request the component to use it
- Form1 form = (Form1) container[typeof(Form1)];
- // Use the component
- Application.Run(form);
- // Release it
- container.Release(form);

Castle.Windsor naudojimo pavyzdys (3)

Tarkime, reikia programą papildyti nauju funkcionalumu: parašyti HttpServiceWatcher, kuris laikas nuo laiko tikrintų, ar Http servisas yra gyvas.

Kas atsitinka, jei Http servisas neveikia? Gal siųsti e-laišką, Gal pyptelėti aliarmą per garsiakablius?

Svarbiausias klausimas – jei reikia šių abiejų dalykų, tai kaip rašyti kodą?

Blogas atsakymas būtų toks – sukurti HttpServiceWatcher kuris daro visus tris dalykus iš karto (watch, email ir alarm)

mailto:Darius.Damalakas@gmail.com

Castle.Windsor naudojimo pavyzdys (4)

- HttpServiceWathcer turi tikrinti Http servisą ir nieko daugiau
- E-laiškų siuntimas ir aliarmai galėtų būti dvi implementacijos IFailureNotifier interfeiso.

```
public interface IFailureNotifier
{
    void Notify();
}
```

```
public class AlarmFailureNotifier :
  IFailureNotifier
{
    public void Notify()
    {
        // Turn on alarm
    }
}
```

Castle.Windsor naudojimo pavyzdys (5)

- IFailuteNotifier yra privalomas HttpServiceWatcher klasei.
- IFailuteNotifier turi būti perduodamas per konstruktorių

```
public class HttpServiceWatcher
{
    private IFailureNotifier notifier;
    public HttpServiceWatcher(IFailureNotifier notifier)
    {
        this.notifier = notifier;
    }
    public void StartWatching()
    {
        // should start a thread to ping the service
        // if (pingresult == Failed)
        // {
            notifier.Notify();
        // }
    }}
```

```
public void StopWatching()
    {
        // stop thread
    }
```


mailto:Darius.Damalakas@gmail.com

Castle.Windsor naudojimo pavyzdys (6)

HttpServiceWatcher surenkame į visumą tokiu būdu:

```
IWindsorContainer container = new WindsorContainer();

// Register the components

container.AddComponent("httpservicewatcher",
  typeof(HttpServiceWatcher));

container.AddComponent("email.notifier", typeof(IFailureNotifier),
  typeof(EmailFailureNotifier));

container.AddComponent("alarm.notifier", typeof(IFailureNotifier),
      typeof(AlarmFailureNotifier));

container.AddComponent("form.component", typeof(Form1));

// Request the component to use it

Form1 form = (Form1) container[typeof(Form1)];
```

Natūralūs klausimai

- Kurį komponentą IFailureNotifier HttpServiceWatcher gavo?
- Standartiškai, Windsor konteineris duoda pirmąjį sutiktą tinkamą komponentą.

- Kaip paduoti sąrašą IfailureNotifier?
- Kaip paduoti parametrą (pvz url) HttpServiceWatcher ?

mailto:Darius.Damalakas@gmail.com

Kokiu būdu galima iškelti konteinerio konfigūraciją?

```
<configuration>
    <component ="httpservicewatcher"
        type="GettingStartedPart1.HttpServiceWatcher, GettingStartedPart1" />
    <component ="email.notifier"
        service="GettingStartedPart1.IFailureNotifier, GettingStartedPart1"
        type="GettingStartedPart1.EmailFailureNotifier, GettingStartedPart1" />
    <component ="alarm.notifier"
        service="GettingStartedPart1.IFailureNotifier, GettingStartedPart1"
        type="GettingStartedPart1.AlarmFailureNotifier, GettingStartedPart1" />
    <component  id="form.component"
        type="GettingStartedPart1.Form1, GettingStartedPart1" /></configuration>
```


mailto:Darius.Damalakas@gmail.com

Formos kūrimas panaudojant išorinę konteinerio konfigūraciją

```
IWindsorContainer container =
        new WindsorContainer(
            new XmlInterpreter(new ConfigResource("castle")));
    // Request the component to use it
    Form1 form = (Form1) container[typeof(Form1)];
```


mailto:Darius.Damalakas@gmail.com

Kaip nurodyti HTTPServiceWatcher URL ir specifinį INotifier

```
<component  id="httpservicewatcher"
    type="GettingStartedPart1.HttpServiceWatcher,
     GettingStartedPart1">
    <parameters>
        <notifier>${alarm.notifier}</notifier>
        <Url>different url</Url>
    </parameters>
</component>
```

- The ${} notation is called service lookup.

mailto:Darius.Damalakas@gmail.com

Kaip gauti IFailureNotifier sąrašą?

```
<component   id="httpservicewatcher"
   type="GettingStartedPart1.HttpServiceWatcher,GettingStartedPart1">
   <parameters>
      <notifiers>
         <array>
            <item>${email.notifier}</item>
            <item>${alarm.notifier}</item>
         </array>
      </notifiers>
      <Url>different url</Url>
   </parameters>
</component>
```


mailto:Darius.Damalakas@gmail.com

IoC konteineris Castle.Windsor

- Castle.Windsor IoC galimybių apžvalga baigta.

***Castle.Windsor ir** Interceptors*

mailto:Darius.Damalakas@gmail.com

Interceptors – trumpai ir greitai

- Interceptoriai padeda pasiekti AOP (Aspect Oriented Programmimg) galimybes
- Kiekviena programavimo idėja (procedūrinis programavimas/ objektiškai orientuotas) vienaip ar kitaip leidžia atskirti interesus vieną nuo kito
- Procedūros, paketai, klasės ir metodai padeda grupuoti interesus ir taip skaidyti juos į atskiras esybes
- Bet yra ir tokių interesų, kurie perkerta visas esybes. Tokie interesai vadinami “cross-cutting” concerns.

mailto:Darius.Damalakas@gmail.com

Cross-cutting concern - logging

Įsivaizduokime situaciją – programos kūrimo viduryje atsiranda poreikis registruoti visus pirkimo įvykius specialiame log faile.

Problema ta, kad norint tokį funkcionalumą pridėti, reikia keisti visus programos modulius, kurie susiję su pirkimais.

AOP leidžia pridėti / keisti “aspektus”.

mailto:Darius.Damalakas@gmail.com

Rezultatas turi atrodyti taip:

Su kiekvienu operacijos return iškvietimu, žurnale turi atsirasti nauajs įrašas.

- order.Return(...)

Kodas turi išlikti toks pat, kaip ir ankščiau, t.y. Jokio naujo kodo metode Return neturi būti.

mailto:Darius.Damalakas@gmail.com

Kaip pasiekti logginimą su Windsor

Esmė – sukurti objektus, kurie dinamiškai perimtų reikiamų metodų kvietimus (šiuo atveju Return) ir leistų iškviesti kitus, logginimo metodus.

Reikiamas funkciolumas gali būti įvykdytas prieš metodo kvietimą ir po kvietimo (pre ir post-call).

Rezultatas su Windsor atrodo taip:

- IWindsorContainer container = new WindsorContainer("windsor.config");
- IOrder order = container.Resolve<IOrder>();
- order.Return(...);

mailto:Darius.Damalakas@gmail.com

Kaip pasiekti logginimą su Windsor

```
<facility id="policy-injection"   type="Windsor.Interception.PolicyFacility, Windsor.Interception">
    <policy name="logMethodsWhoseNameIsReturn">
        <rules>
            <rule type="Windsor.Interception.NameMatchingMethodRule, Windsor.Interception"
                  matching-name="Return"/>
        </rules>
        <handlers>
            <handler  type="Windsor.Interception.LoggingMethodCallHanlder,
                Windsor.Interception"/>
        </handlers>
    </policy>
</facility>
```


mailto:Darius.Damalakas@gmail.com

Kaip pasiekti logginimą su Windsor

- Detalės -

http://www.ayende.com/Blog/archive/2007/03/07/Building-the-Policy-Injection-in-40-Minutes-with-Windsor.asp

http://ayende.com/Blog/archive/2007/03/13/Recommended-approaches-to-AOP-with-Windsor.aspx

mailto:Darius.Damalakas@gmail.com

Kas slepiasi už Castle? - CastleStrongHold!

- Castle Stronghold specializes in delivering premium software. We are internationally known for the quality of our work, for pushing cutting-edge technology and for our involvement with successful open-source projects
- Development - You can hire us to plan, design and implement your custom software applications
- Consultation - Our consultation services keep our professionals closely involved with your project
- Support - We offer subscription-based support to companies that use the Castle Project to build their enterprise software applications
- http://www.castlestronghold.com/

mailto:Darius.Damalakas@gmail.com

Daugiau informacijos

- Castle.Windsor - http://www.castleproject.org/container/index.html
- Castle.MonoRail - http://www.castleproject.org/monorail/index.html
- Castle.ActiveRecord - http://www.castleproject.org/activerecord/index.html

Skaidrės bus čia:
http://griuvesiai.blogspot.com/2007/03/castlewindsor-ioc-konteineris.html

mailto:Darius.Damalakas@gmail.com