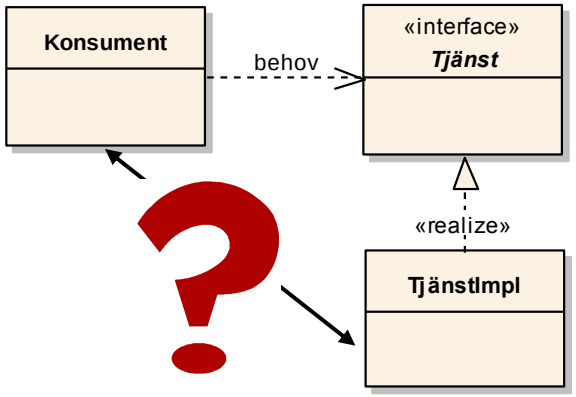
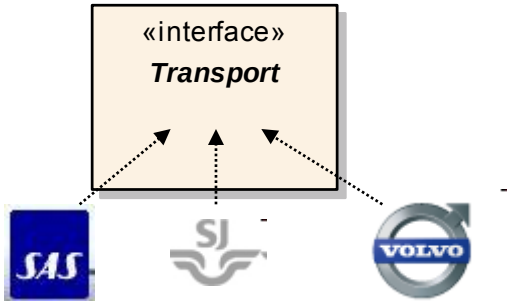
Dependency Injection mönster

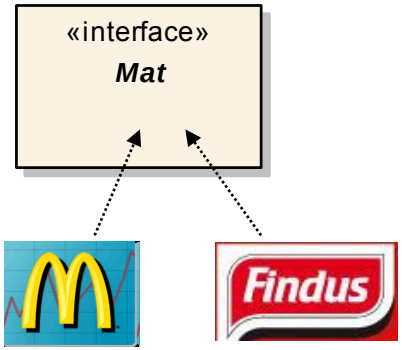
Peter Norrhall

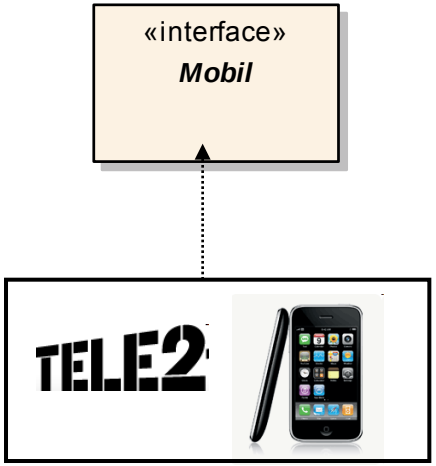
mindedge

Konsumenter & Tjänster









mindedge

Tjänstemäklarna - IoC Ramverk

Injection

- Tre signifikanta

```
public Tonåring(Mobil mobil)

public void setMobil(Mobil mobil)

Mobil mobil;
```

Constructor Injection (Guice)

```
public class Spädbarn extends AbstractHuman
{
   @Inject
   public Spädbarn(Blöja, Välling, Kärlek)
   {
   ...
   }
   ...
}
```

Constructor Injection

```
public class Tonåring extends Spädbarn {
   @Inject
   public Tonåring(Mobil, Sova, Kompisar,
Kärlek) {
   ...
   }
   ...
}
```

Constructor Injection (Spring)

```
<bean id=”tonårsmobil" class="com.tele2.services.Abonnemang">
  <property name=”abonnemang"><ref local=”kontaktKort"/></property>
  <property name=”mobiltelefon"><ref local=”iPhone"/></property>
</bean>

<!- Constructor injection -->
<bean id=”tonåring" class=”god.model.human.Tonåring">
  <constructor-arg ref=”tonårsmobil"></constructor-arg>
  <constructor-arg ref=”säng"></constructor-arg>
...
</bean>
```

Constructor Explosion

```
Vuxen extends Tonåring {
   @Inject
   public Vuxen(MeningenMedLivet, Kärlek,
Familj, Jobb, Träna, Bil, Hus,
FamiljeMiddagar, Sommarsemester,
Vintersemester, ...) {
   ...
   }
   ...
}
```

Constructor vs Setter

- Constructor
  - Tydlighet
  - Inga sidoeffekter
  - Lättmockat
  - Lätt att veta om/när objektet är redo att använda
- Setter/Field
  - Enkelhet
  - Default Constructor
  - Sidoeffekter

Scope

Gifta

Singleton

Liv/
Odödlig

No-Scope

mindedge

Scope

- Singleton (Cache)
  - Per Injector
- No-Scope
- Domain Specific
  - Web - Session, Request, Flash, Conversation
- Custom Scope

Lifecycle Management

Start/Stop

Eager

Lazy

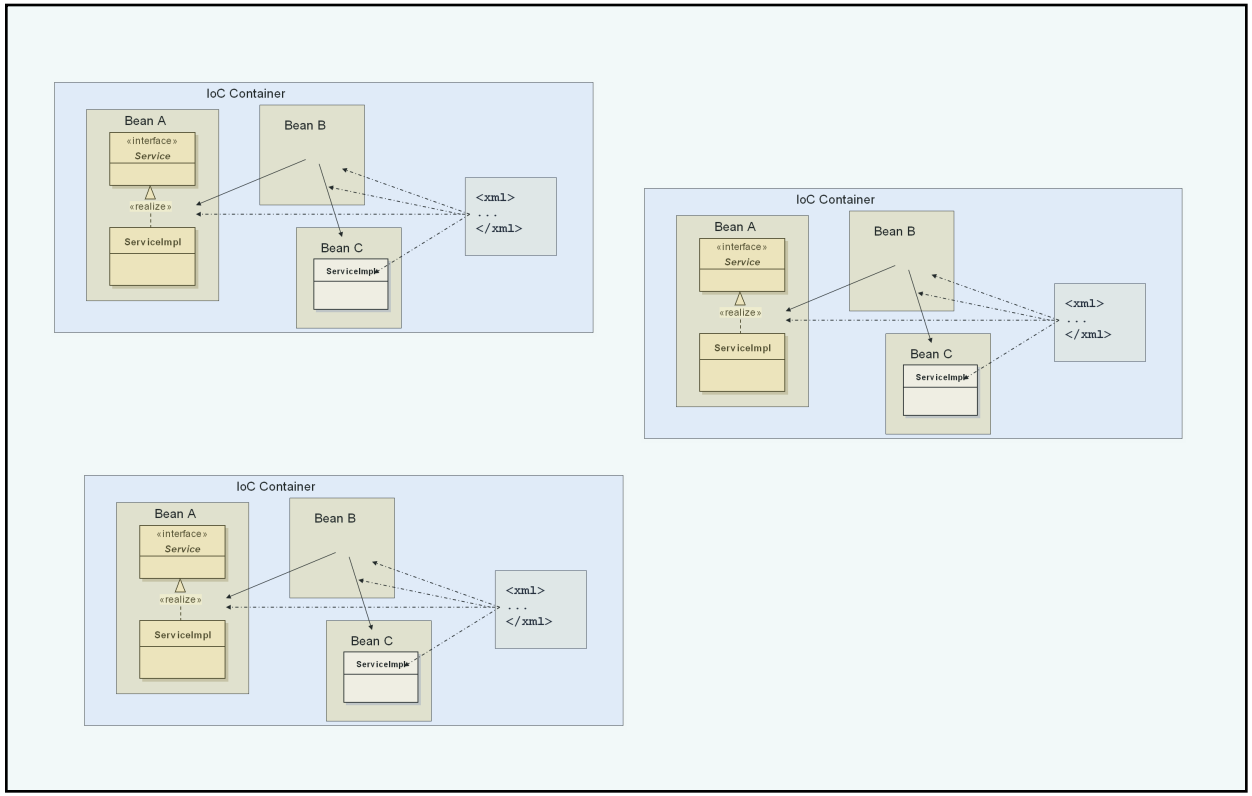
Replace/Unload

mindedge

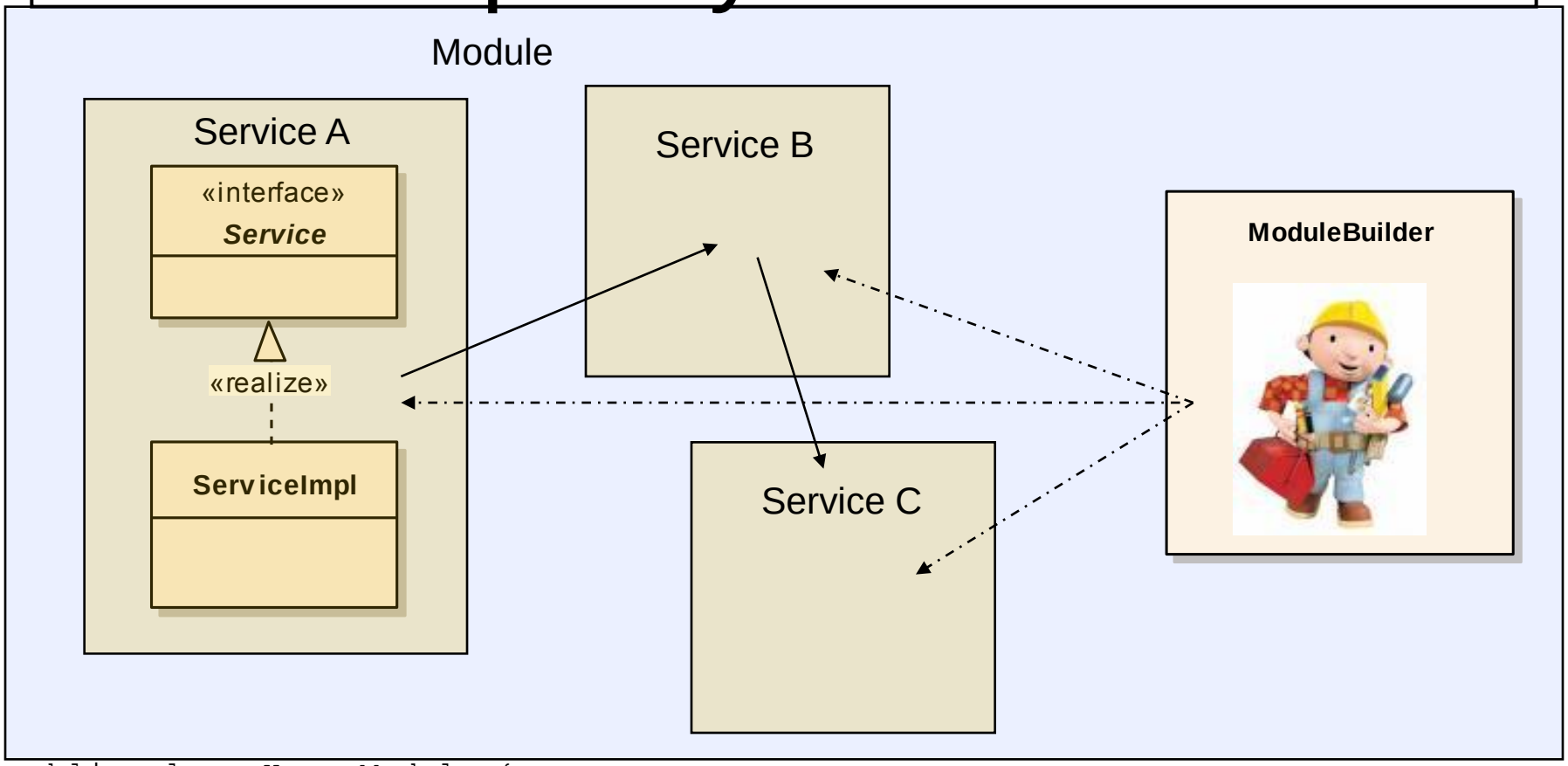
Lifecycle

- Lifecycle management covers
  - Loadtime
    - Eager
    - Lazy
    - Replacing / unloading
  - Execution (of active objects)
    - Starting
    - Stopping

Moduler

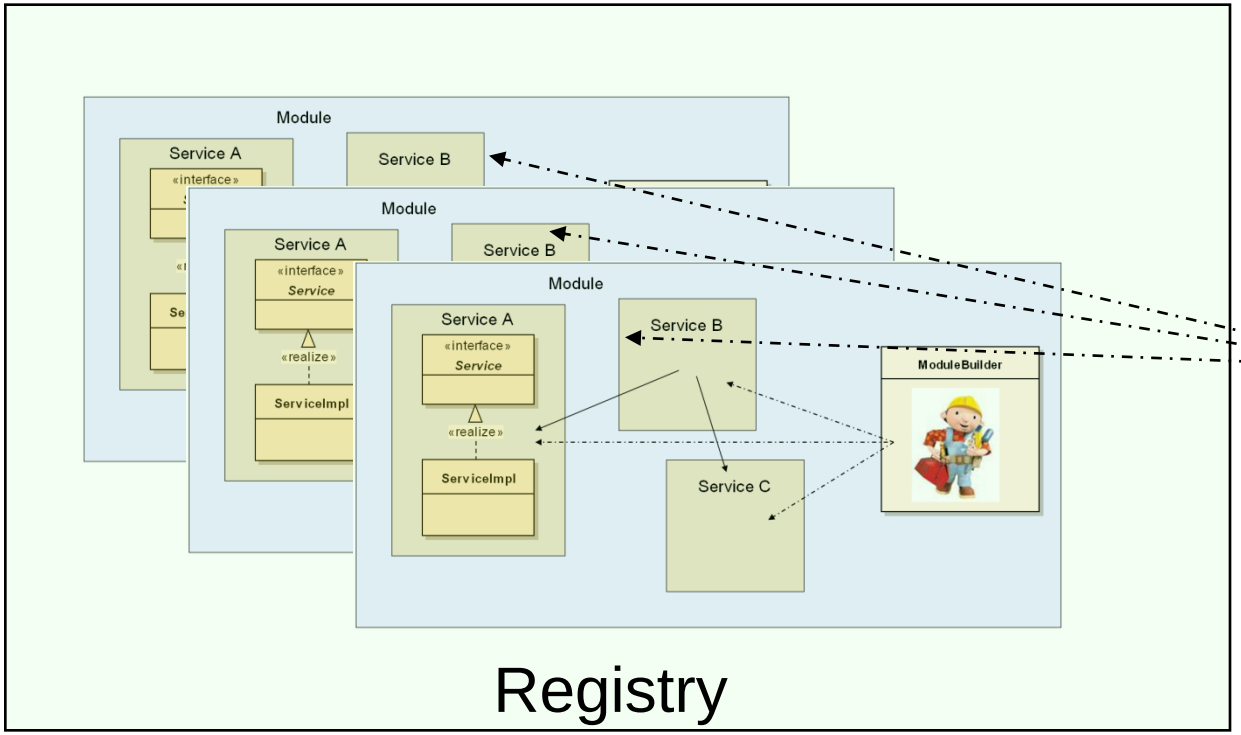


Tapestry 5 Module



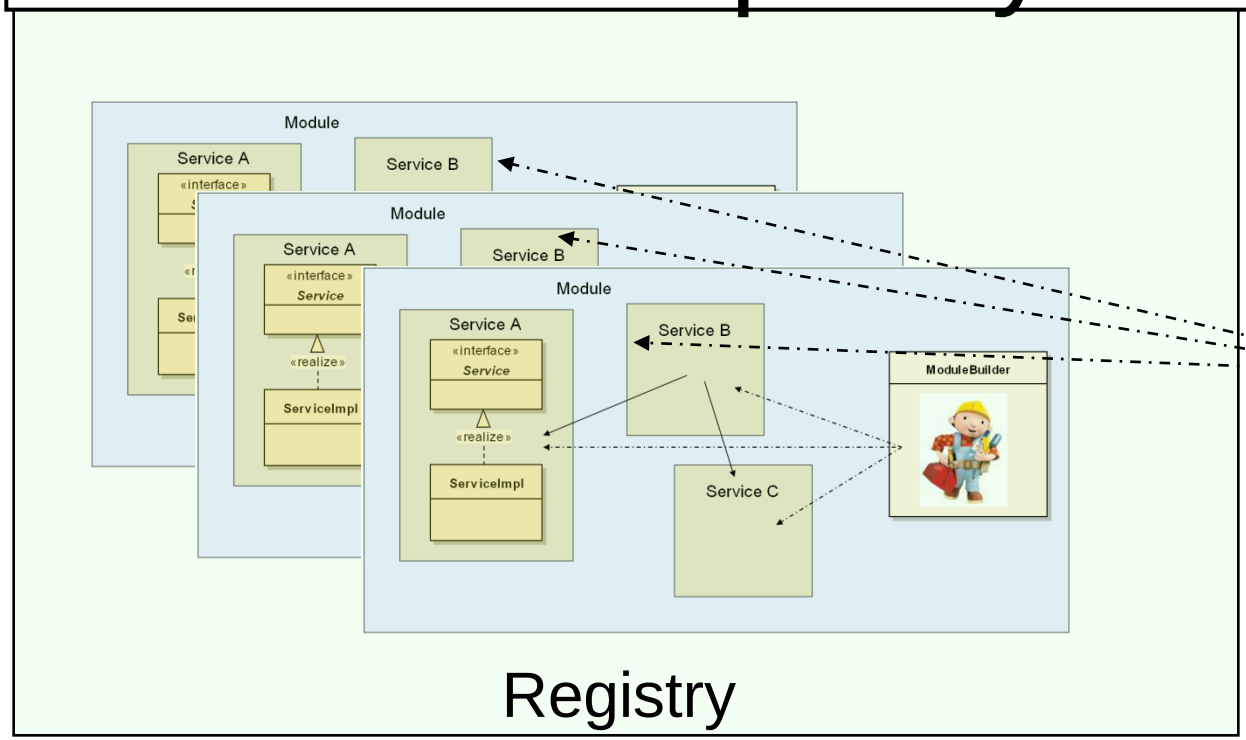

```
public class HumanModule {
  public static void bind(ServiceBinder binder) {
    binder.bind(Mobil.class, IphoneMobil.class).withId("Mobil");
  }
}
```

Guice



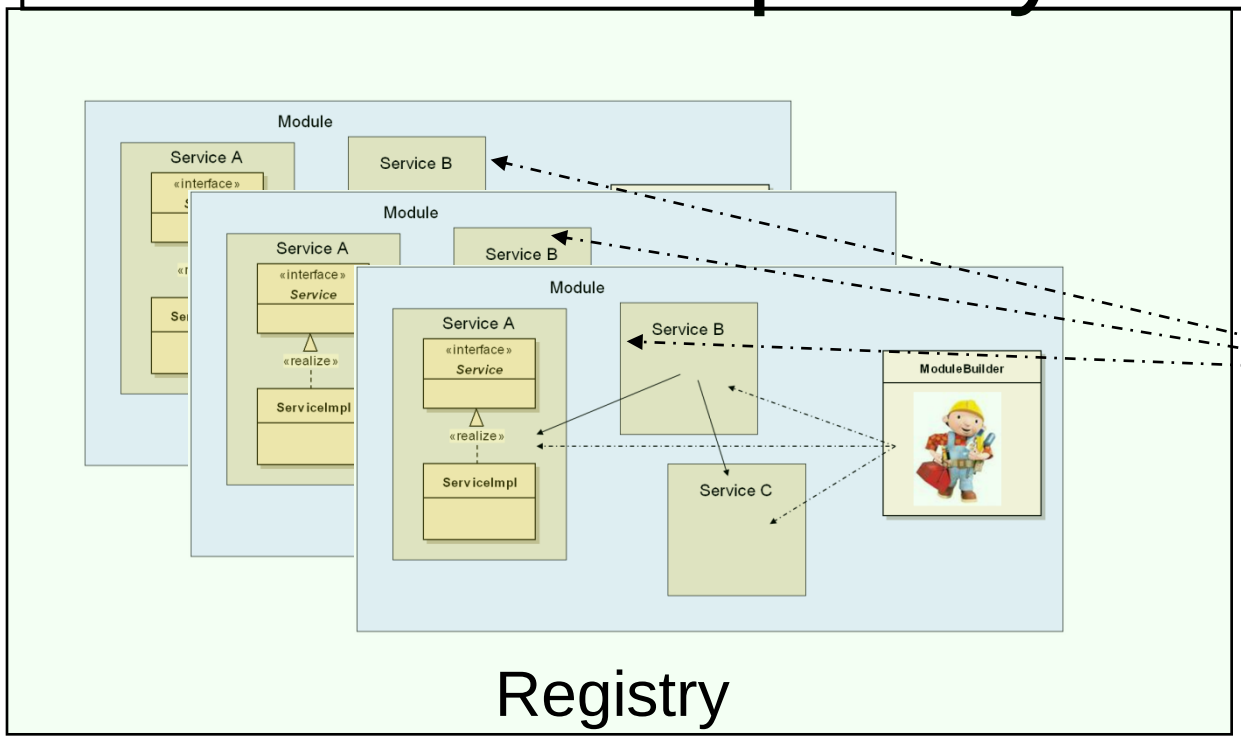

```
Module[] modules = {new GuiceModule()};
Injector injector = Guice.createInjector(modules);
MailService service = injector.getInstance(MailService.class);
```

Tapestry 5



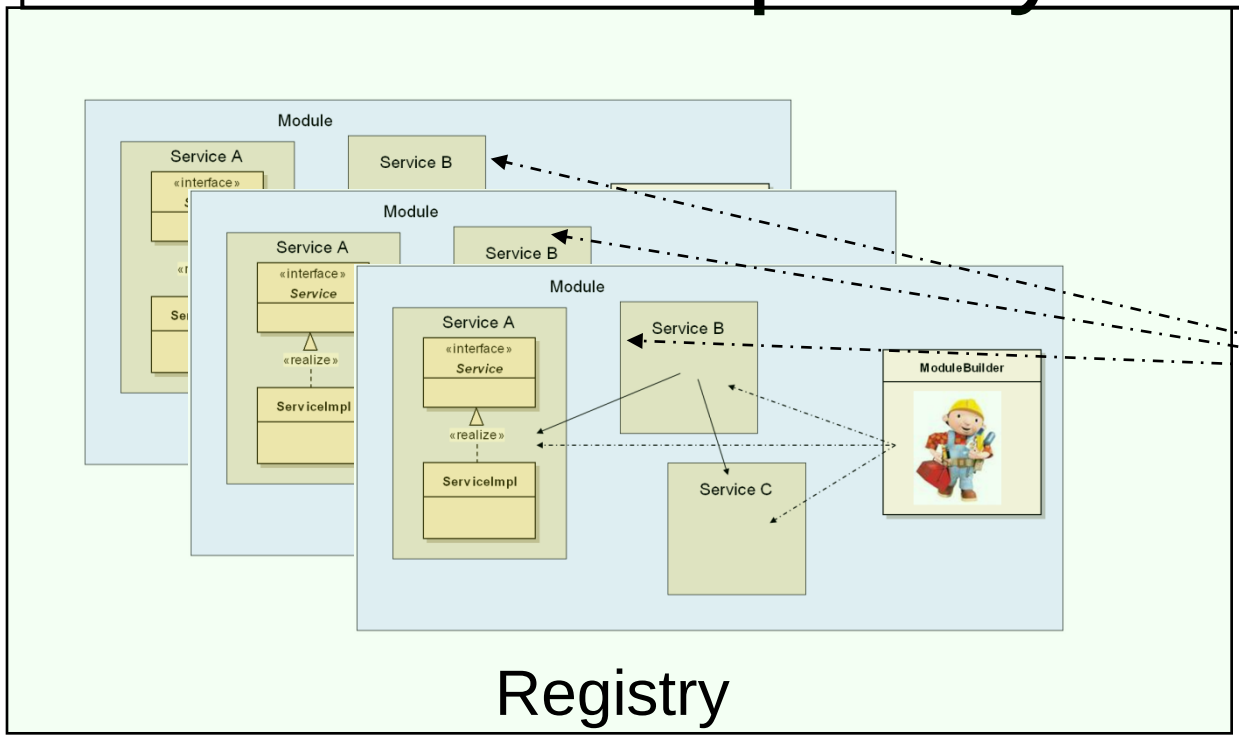

```
RegistryBuilder builder = new RegistryBuilder();
builder.add(TapestryModule.class);
Registry registry = builder.build();
registry.performRegistryStartup();
OrderManager orderManager = registry.getService(OrderManager.class);
```

Tapestry 5



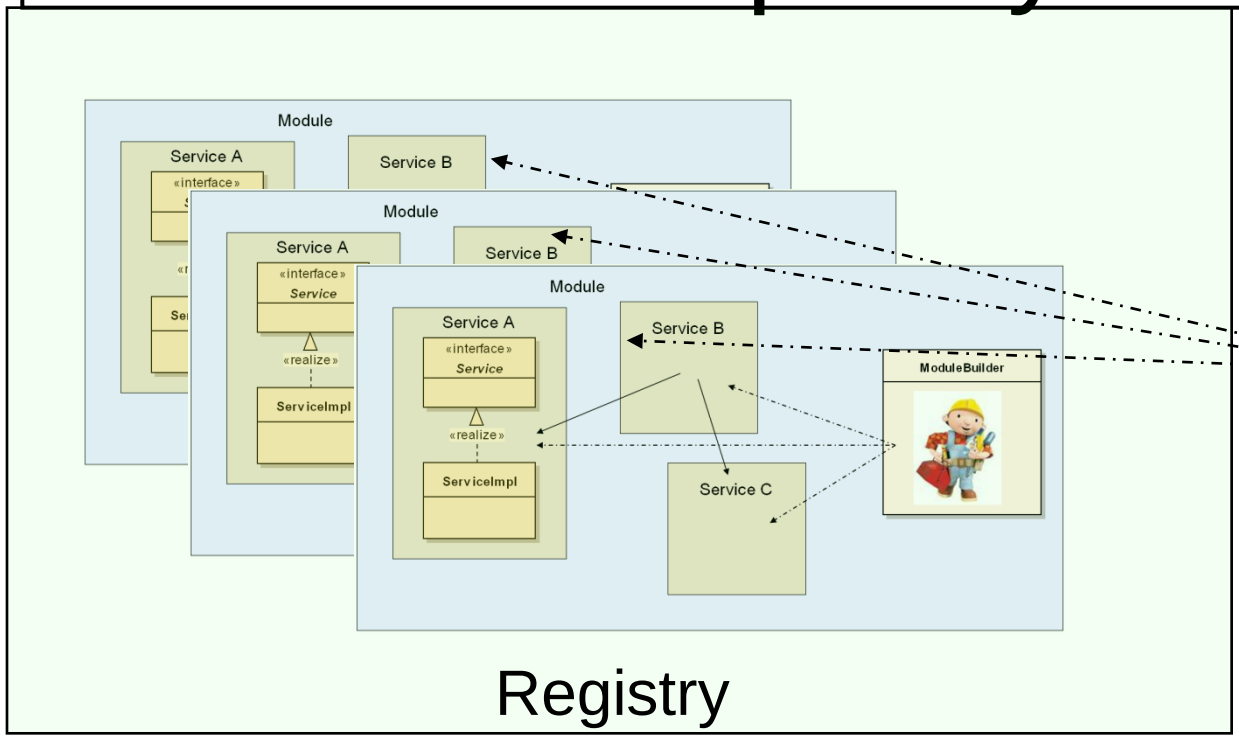

```
RegistryBuilder builder = new RegistryBuilder();
builder.add(TapestryModule.class);
Registry registry = builder.build();
registry.performRegistryStartup();
OrderManager orderManager = registry.getService(OrderManager.class);
```

Tapestry 5



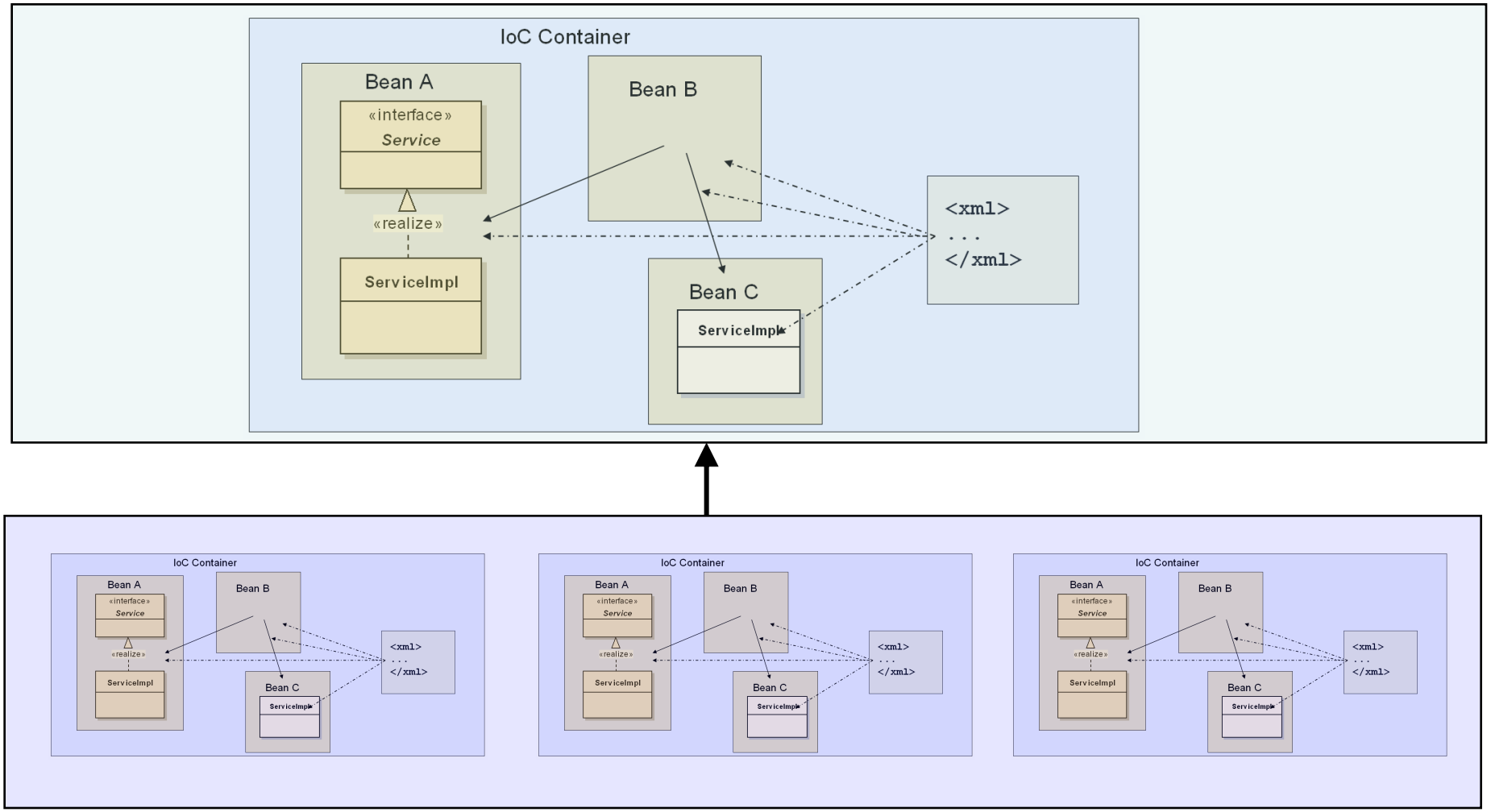

```
RegistryBuilder builder = new RegistryBuilder();
builder.add(TapestryModule.class);
Registry registry = builder.build();
registry.performRegistryStartup();
OrderManager orderManager = registry.getService(OrderManager.class);
```

Tapestry 5

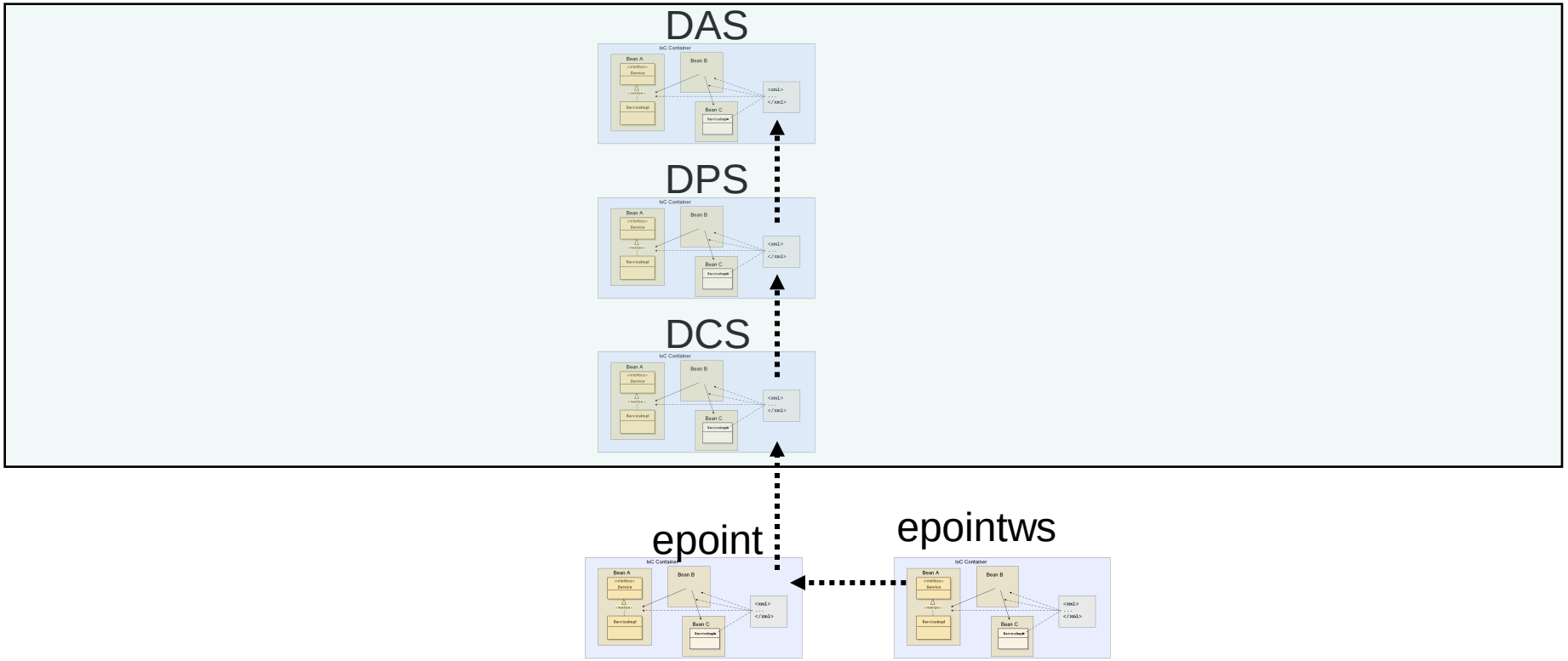



```
RegistryBuilder builder = new RegistryBuilder();
builder.add(TapestryModule.class);
Registry registry = builder.build();
registry.performRegistryStartup();
OrderManager orderManager = registry.getService(OrderManager.class);
```

Lager


IoC Container
Bean A
«interface»
Service
«realize»
ServiceImpl
Bean B
Bean C
ServiceImpl
<xml>
...
</xml>
IoC Container
Bean A
«interface»
Service
«realize»
ServiceImpl
Bean B
Bean C
ServiceImpl
<xml>
...
</xml>
IoC Container
Bean A
«interface»
Service
«realize»
ServiceImpl
Bean B
Bean C
ServiceImpl
<xml>
...
</xml>
IoC Container
Bean A
«interface»
Service
«realize»
ServiceImpl
Bean B
Bean C
ServiceImpl
<xml>
...
</xml>

ATG Dynamo



Manifest-Version: 1.0
ATG-Required: epoint

```
>startdynamo –m epointws
```


mindedge

Guice 2 Lagerkonfiguration

```
Module combinedModule
= Guice.overrideModule(new DCSModule(), new EpointModule());
```

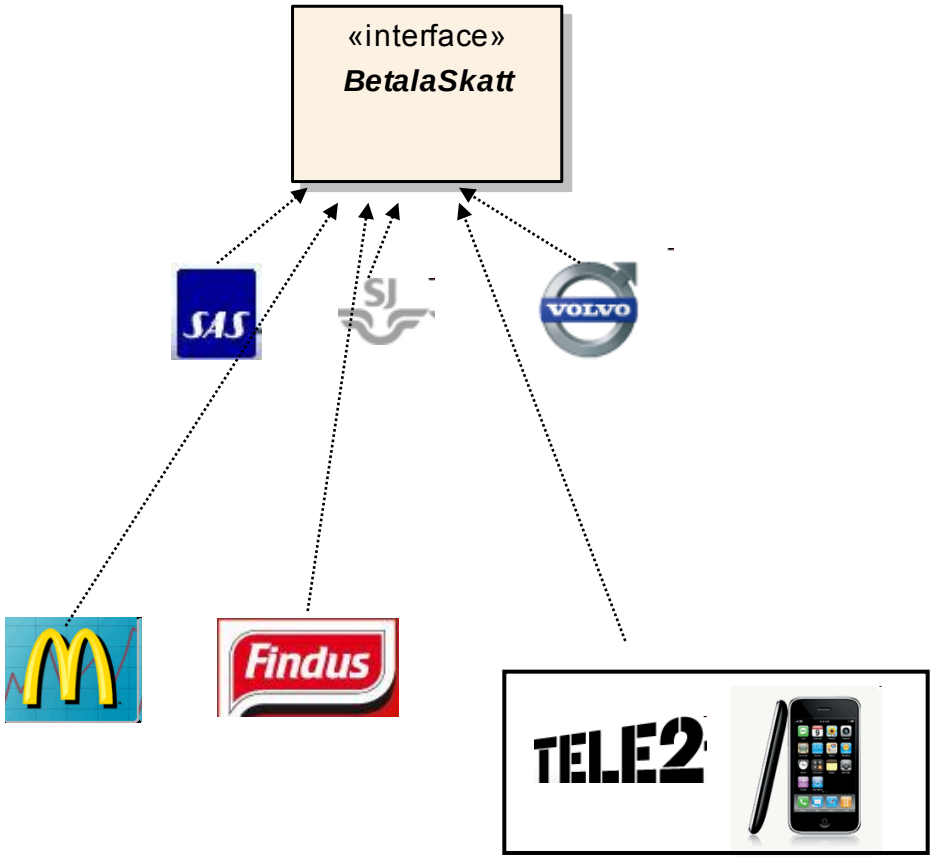
Assembler

- Guice – Injector
- Tapestry – Registry
- Spring – ApplicationContext/BeanFactory

Konfiguration

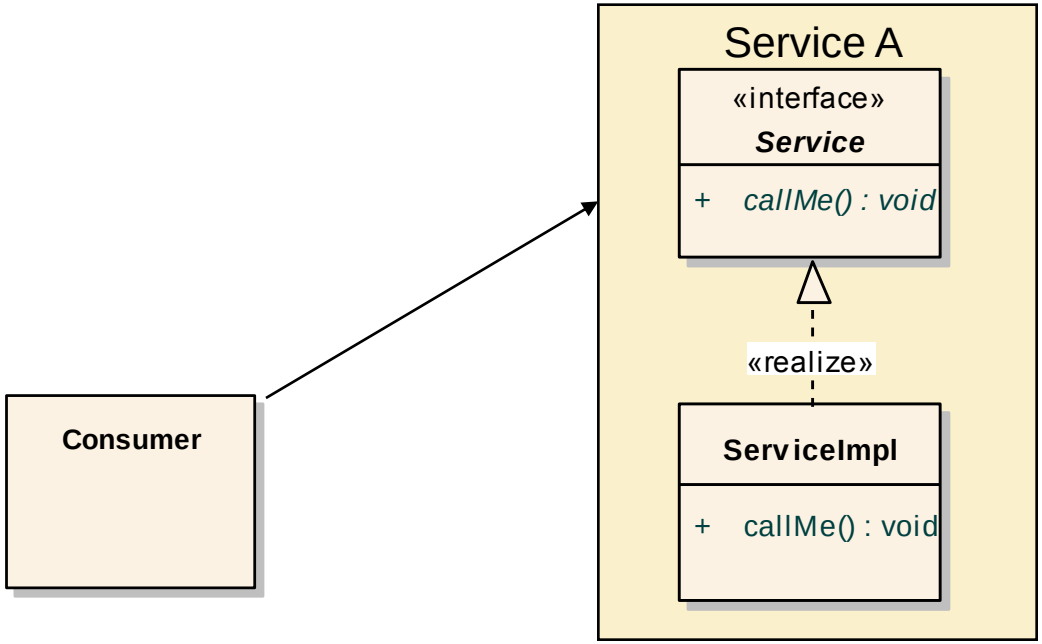
- Externt (XML, .properties)
- Kod (POJO, Annoteringar)

Aspekter



mindedge

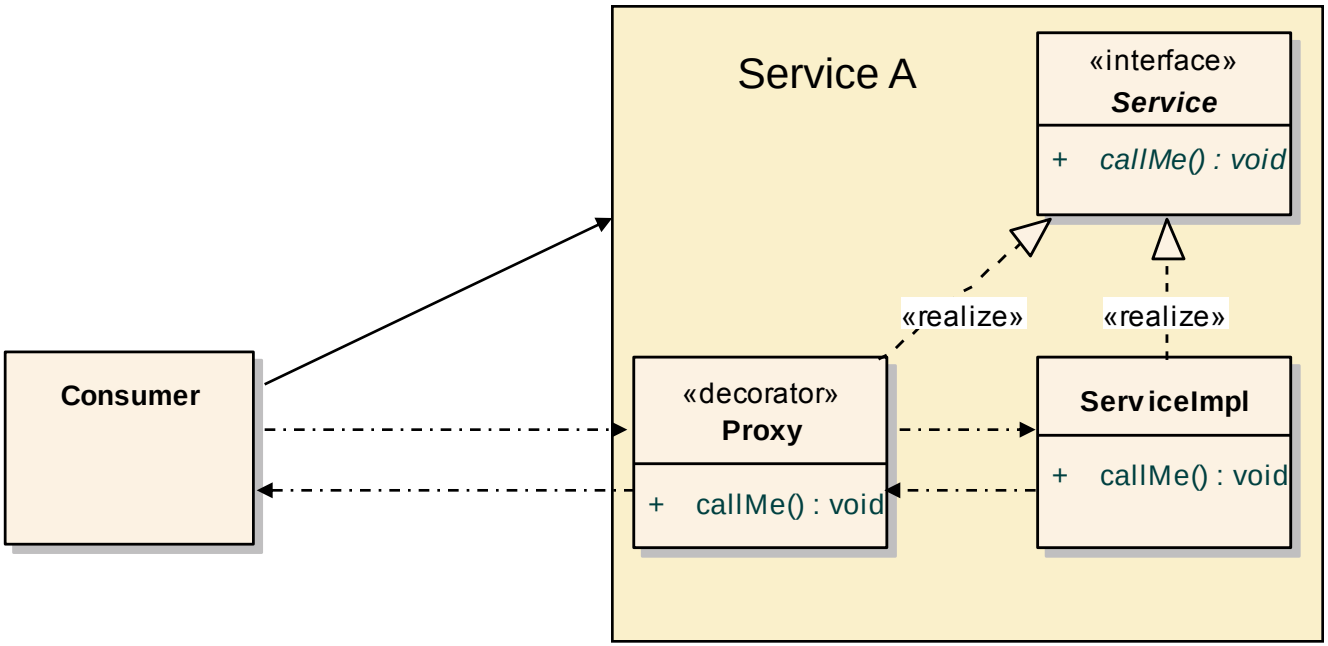
Tapestry IoC Decorators (”AOP”)




```
public class Consumer {
  Service service;
...
    ServiceImpl serviceImpl = (ServiceImpl)service;

}
```

Tapestry IoC Decorators ("AOP")



AOP

- Guice/Tapestry - Proxy/Interceptor
- Spring – AOP Proxy/Advice
- Qi4j – Concerns
  - SideEffects

Advice

”Learn at least one new programming language every year”
Andrew Hunt & David Thomas

”Learn at least one new IoC framework every year”
Peter Norrhall

Avslutande Frågor

peter.norrhall@mindedge.se