

20.01.2022

PATİKA – INNOVA JAVA SPRİNG ODEV – 2

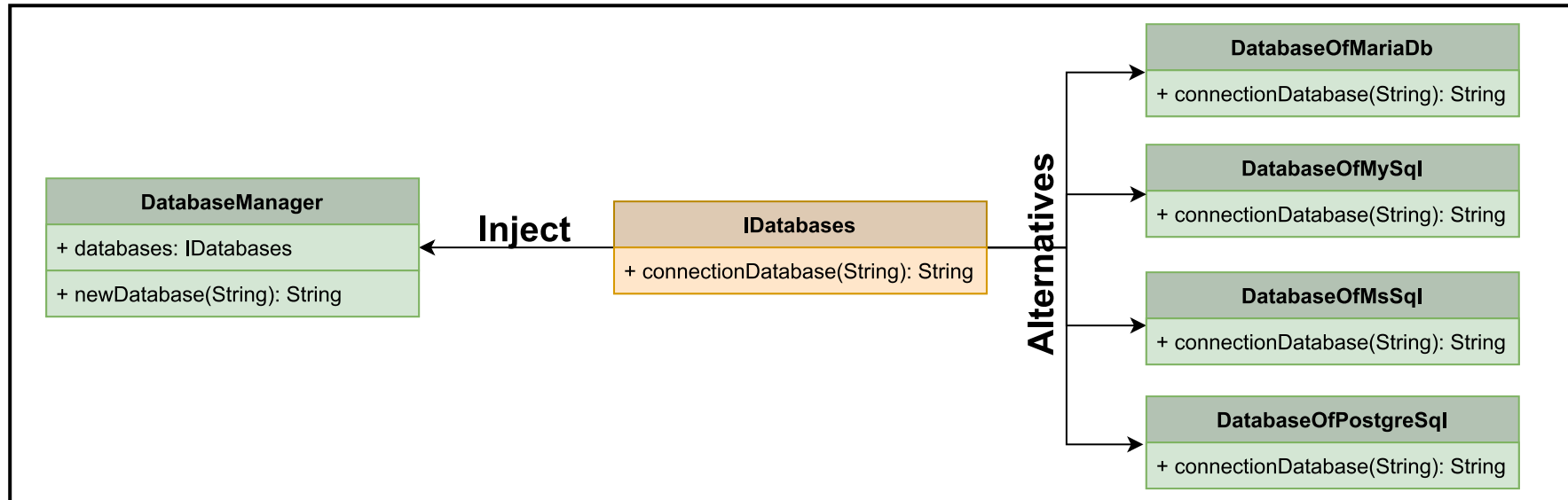
Bekir Gürkan Güldaş

İçindekiler

1. Alternative Annotation	3
2. Qualifer Annotation	5
3. Enumqualifer Annotation	8
4. Interceptor Annotation	11
5. Streotypes Annotation	15

1. Alternative Annotation

Farklı amaçlar için kullanılan bir bean'nin birden fazla versiyonu mevcut ise bir niteleyiciyi ile enjekte ederek bunlar arasında seçim yapılabilir. Kaynak kodunu değiştirmek zorunda kalmak yerine, alternatifleri kullanarak dağıtım zamanında seçim yapmayı mümkün kılar.



1. Alternative Annotation

```
1 package com.innova.CDI;
2
3 import java.io.Serializable;
4
5 @Named(value = "DatabaseManager")
6 @ApplicationScoped
7 public class DatabaseManager implements Serializable {
8
9     private static final long serialVersionUID = 164847823333869916L;
10
11     @Inject
12     private IDatabases databases;
13
14     public String newDatabase()
15     {
16         return databases.connectionDatabase();
17     }
18 }
```

```
1 package com.innova.CDI;
2
3 public interface IDatabases {
4
5     String connectionDatabase();
6 }
```

```
1 package com.innova.CDI;
2
3 import javax.enterprise.inject.Alternative;
4
5 @Alternative
6 public class DatabaseOfPostgreSQL implements IDatabases {
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to PostgreSQL.";
12     }
13 }
```

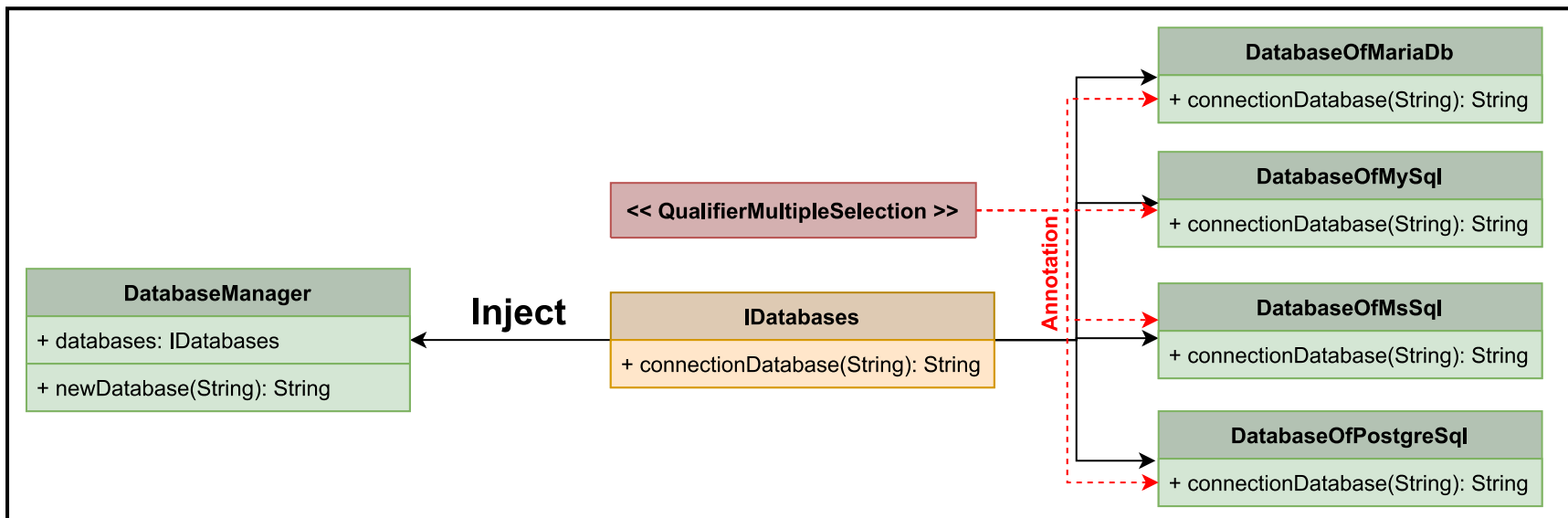
```
1 package com.innova.CDI;
2
3 import javax.enterprise.inject.Alternative;
4
5 @Alternative
6 public class DatabaseOfMySQL implements IDatabases {
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to MySQL.";
12     }
13 }
```

```
1 package com.innova.CDI;
2
3 import javax.enterprise.inject.Alternative;
4
5 @Alternative
6 public class DatabaseOfMsSql implements IDatabases {
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to MsSQL.";
12     }
13 }
```

```
1 package com.innova.CDI;
2
3 import javax.enterprise.inject.Alternative;
4
5 @Alternative
6 public class DatabaseOfMariaDb implements IDatabases {
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to MariaDb.";
12     }
13 }
```

2. Qualifer Annotation

Kullanılan dependency injection konteyner ortamında aynı nesne türünden birden fazla enjekte edilebilir kaynak bulunduğunda, hangisinin seçileceğini belirlemek için kullanılan bir annotationdır.



2. Qualifer Annotation

```
1 package com.innova.Qualifier;
2
3+ import java.io.Serializable;
4
5 @Named(value = "QualifierDatabaseManager")
6 @ApplicationScoped
7 public class DatabaseManager implements Serializable{
8
9     private static final long serialVersionUID = 164847823333869916L;
10
11     //Default Inject
12
13     //@Inject
14     //private IDatabases databases;
15
16     //Qualifier Inject
17
18     @Inject
19     @QualifierMultipleSelection
20     private IDatabases databases;
21
22 @ public String newDatabase()
23 {
24     return databases.connectionDatabase();
25 }
26 }
```

```
1 package com.innova.Qualifier;
2
3- import java.lang.annotation.Documented;
4 import java.lang.annotation.Retention;
5 import java.lang.annotation.Target;
6
7 import javax.inject.Qualifier;
8
9 import static java.lang.annotation.ElementType.FIELD;
10 import static java.lang.annotation.ElementType.METHOD;
11 import static java.lang.annotation.ElementType.PARAMETER;
12 import static java.lang.annotation.ElementType.TYPE;
13 import static java.lang.annotation.RetentionPolicy.RUNTIME;
14
15 @Qualifier
16 @Target({ TYPE, METHOD, PARAMETER, FIELD })
17 @Retention(RUNTIME)
18 @Documented
19 public @interface QualifierMultipleSelection {
20
21 }
```

2. Qualifer Annotation

```
1 package com.innova.Qualifier;
2
3 import javax.enterprise.inject.Default;
4
5 @Default
6 public class DatabaseOfMariaDb implements IDatabases{
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to MariaDb.";
12     }
13 }
14 }
```

```
1 package com.innova.Qualifier;
2
3 public class DatabaseOfPostgreSql implements IDatabases{
4
5     @Override
6     public String connectionDatabase()
7     {
8         return "Connected to PostgreSQL.";
9     }
10 }
```

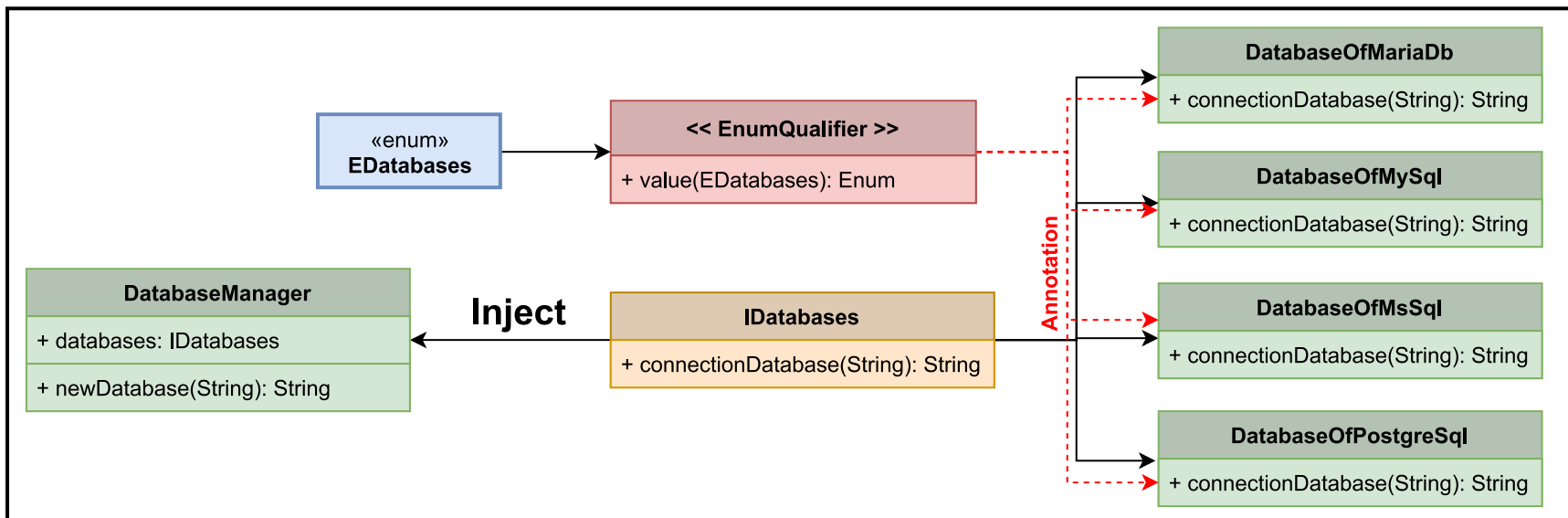
```
1 package com.innova.Qualifier;
2
3 public class DatabaseOfMySql implements IDatabases{
4
5     @Override
6     public String connectionDatabase()
7     {
8         return "Connected to MySQL.";
9     }
10 }
```

```
1 package com.innova.Qualifier;
2
3 public interface IDatabases {
4
5     String connectionDatabase();
6 }
```

```
1 package com.innova.Qualifier;
2
3 @QualifierMultipleSelection
4 public class DatabaseOfMsSql implements IDatabases{
5
6     @Override
7     public String connectionDatabase()
8     {
9         return "Connected to MsSQL.";
10    }
11 }
```

3. Enumqualifier Annotation

Enjekte edilebilir kaynakların tek bir notasyonla seçilebilir kılmak için aşağıdaki gibi bir enum oluşturulabilir.



3. Enumqualifier Annotation

```
1 package com.innova.EnumQualifier;
2
3 import java.io.Serializable;
4
5 import javax.enterprise.context.ApplicationScoped;
6 import javax.inject.Inject;
7 import javax.inject.Named;
8
9 @Named(value = "EnumQualifierDatabaseManager")
10 @ApplicationScoped
11 public class DatabaseManager implements Serializable{
12
13     private static final long serialVersionUID = 1648478233333869916L;
14
15     //Default Inject
16
17     //@Inject
18     //private IDatabases databases;
19
20     //Enum Qualifier Inject
21
22     @Inject
23     @EnumQualifier(EDatabases.MSSQL)
24     private IDatabases databases;
25
26     public String newDatabase()
27     {
28         return databases.connectionDatabase();
29     }
30
31 }
```

```
1 package com.innova.EnumQualifier;
2
3 public enum EDatabases
4 {
5     MariaDB, MSSQL, MySQL, PostgreSQL
6 }
```

```
1 package com.innova.EnumQualifier;
2
3 import static java.lang.annotation.ElementType.FIELD;
4 import static java.lang.annotation.ElementType.METHOD;
5 import static java.lang.annotation.ElementType.PARAMETER;
6 import static java.lang.annotation.ElementType.TYPE;
7 import static java.lang.annotation.RetentionPolicy.RUNTIME;
8
9 import java.lang.annotation.Documented;
10 import java.lang.annotation.Retention;
11 import java.lang.annotation.Target;
12
13 import javax.inject.Qualifier;
14
15 @Qualifier
16 @Target({ TYPE, METHOD, PARAMETER, FIELD })
17 @Retention(RUNTIME)
18 @Documented
19 public @interface EnumQualifier {
20     EDatabases value();
21 }
```

3. Enumqualifer Annotation

```
1 package com.innova.EnumQualifier;
2
3 import javax.enterprise.inject.Default;
4
5 @Default
6 public class DatabaseOfPostgreSql implements IDatabases{
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to PostgreSQL.";
12     }
13 }
```

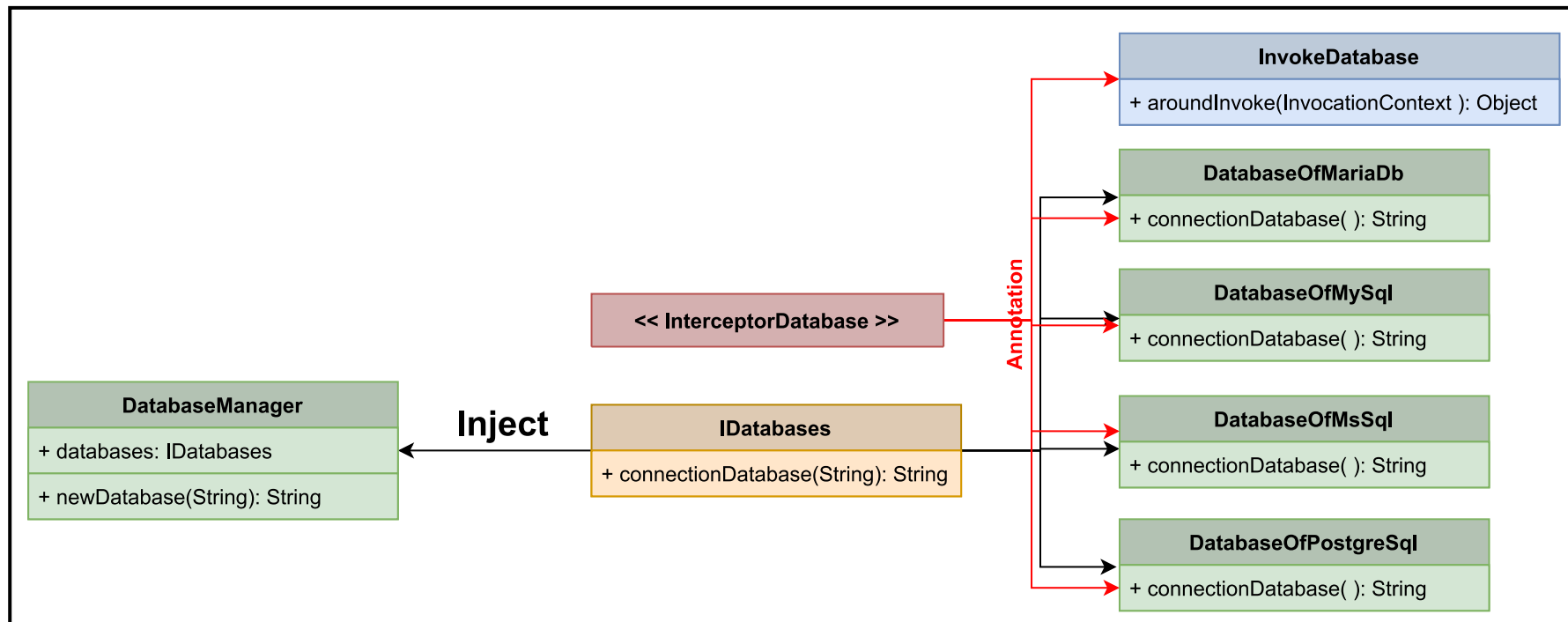
```
1 package com.innova.EnumQualifier;
2
3 @EnumQualifier(EDatabases.MariaDB)
4 public class DatabaseOfMariaDb implements IDatabases{
5
6     @Override
7     public String connectionDatabase()
8     {
9         return "Connected to MariaDb.";
10     }
11 }
```

```
1 package com.innova.EnumQualifier;
2
3 @EnumQualifier(EDatabases.MySQL)
4 public class DatabaseOfMySql implements IDatabases{
5
6     @Override
7     public String connectionDatabase()
8     {
9         return "Connected to MySQL.";
10     }
11 }
```

```
1 package com.innova.EnumQualifier;
2
3 public interface IDatabases
4 {
5     String connectionDatabase();
6 }
7
8 package com.innova.EnumQualifier;
9
10 @EnumQualifier(EDatabases.MSSQL)
11 public class DatabaseOfMsSql implements IDatabases{
12
13     @Override
14     public String connectionDatabase()
15     {
16         return "Connected to MsSQL.";
17     }
18 }
```

4. Interceptor Annotation

Yazılım geliştirme alanında bir önleyici desendir. Yazılım sistemleri veya çerçeveleri normal işlem döngülerini değiştirmek veya artırmak için bir yol sunmak istediğinde kullanılan bir yazılım tasarım modelidir.



4. Interceptor Annotation

```
1 package com.innova.Interceptor;
2
3 import java.io.Serializable;
4
5 import javax.enterprise.context.ApplicationScoped;
6 import javax.inject.Inject;
7 import javax.inject.Named;
8
9 @Named(value = "InterceptorDatabaseManager")
10 @ApplicationScoped
11 public class DatabaseManager implements Serializable{
12
13     private static final long serialVersionUID = 1648478233333869916L;
14
15     @Inject
16     private IDatabases databases;
17
18     public String newDatabase()
19     {
20         return databases.connectionDatabase();
21     }
22
23 }
```

```
1 package com.innova.Interceptor;
2
3 import java.lang.annotation.Documented;
4 import java.lang.annotation.Inherited;
5 import java.lang.annotation.Retention;
6 import java.lang.annotation.Target;
7
8 import javax.interceptor.InterceptorBinding;
9
10 import static java.lang.annotation.ElementType.METHOD;
11 import static java.lang.annotation.ElementType.TYPE;
12 import static java.lang.annotation.RetentionPolicy.RUNTIME;
13
14 @InterceptorBinding
15 @Inherited
16 @Target({ TYPE, METHOD })
17 @Retention(RUNTIME)
18 @Documented
19 public @interface InterceptorDatabase {
20
21 }
```

```
1 package com.innova.Interceptor;
2
3 import javax.interceptor.AroundInvoke;
4 import javax.interceptor.Interceptor;
5 import javax.interceptor.InvocationContext;
6
7 @Interceptor
8 @InterceptorDatabase
9 public class InvokeDatabase {
10
11     @AroundInvoke
12     public Object aroundInvoke(InvocationContext context) {
13
14         boolean isLogin = false;
15         Object isContinue = null;
16         if (isLogin)
17         {
18             System.out.println("Connection Failed.");
19             return null;
20         }
21         else
22         {
23             try
24             {
25                 isContinue = context.proceed();
26                 System.out.println("Connection Succeeded.");
27             }
28             catch (Exception e)
29             {
30                 e.printStackTrace();
31             }
32         }
33         return isContinue;
34     }
35 }
```

4. Interceptor Annotation

```
1 package com.innova.Interceptor;
2
3 import javax.enterprise.inject.Alternative;
4
5 @InterceptorDatabase
6 @Alternative
7 public class DatabaseOfPostgreSql implements IDatabases{
8
9     @Override
10    public String connectionDatabase()
11    {
12        return "Connected to PostgreSQL.";
13    }
14 }
```

```
1 package com.innova.Interceptor;
2
3 import javax.enterprise.inject.Alternative;
4
5 @InterceptorDatabase
6 @Alternative
7 public class DatabaseOfMsSql implements IDatabases{
8
9     @Override
10    public String connectionDatabase()
11    {
12        return "Connected to MsSQL.";
13    }
14 }
```

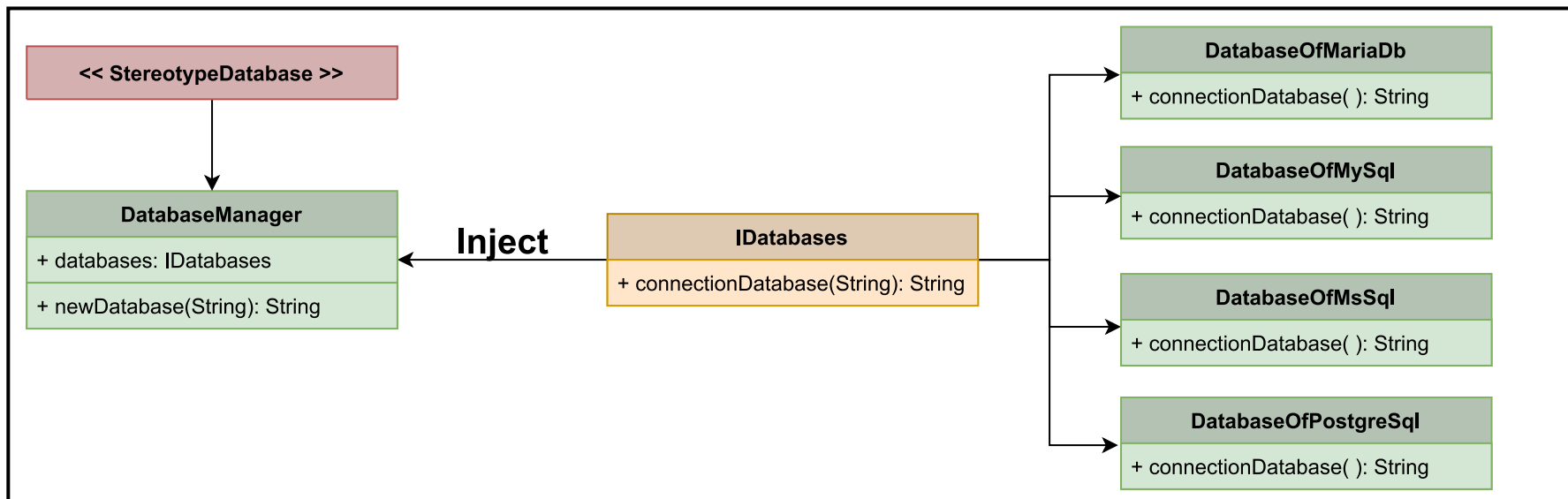
```
1 package com.innova.Interceptor;
2
3 import javax.enterprise.inject.Alternative;
4
5 @InterceptorDatabase
6 @Alternative
7 public class DatabaseOfMySQL implements IDatabases{
8
9     @Override
10    public String connectionDatabase()
11    {
12        return "Connected to MySQL.";
13    }
14 }
```

```
1 package com.innova.Interceptor;
2
3 import javax.enterprise.inject.Alternative;
4
5 @InterceptorDatabase
6 @Alternative
7 public class DatabaseOfMariaDb implements IDatabases{
8
9     @Override
10    public String connectionDatabase()
11    {
12        return "Connected to MariaDb.";
13    }
14 }
15 }
```

```
1 package com.innova.Interceptor;
2
3 public interface IDatabases
4 {
5     String connectionDatabase();
6 }
```

5. Stereotypes Annotation

Stereotype anotasyonu, bir uygulama içinde bir rolü yerine getiren herhangi bir sınıf için belirteçlerdir. Bileşenler için gereken yapılandırmaların kaldırılmasına veya en azından büyük ölçüde azaltılmasına yardımcı olur.



5. Stereotypes Annotation

```
1 package com.innova.Stereotype;
2
3 import java.io.Serializable;
4 import javax.inject.Inject;
5
6 @StereotypeDatabase
7 public class DatabaseManager implements Serializable{
8
9     private static final long serialVersionUID = 164847823333869916L;
10
11     @Inject
12     private IDatabases databases;
13
14     public String newDatabase()
15     {
16         return databases.connectionDatabase();
17     }
18 }
```

```
1 package com.innova.Stereotype;
2
3 import static java.lang.annotation.ElementType.FIELD;
4 import static java.lang.annotation.ElementType.METHOD;
5 import static java.lang.annotation.ElementType.TYPE;
6 import static java.lang.annotation.RetentionPolicy.RUNTIME;
7
8 import java.lang.annotation.Documented;
9 import java.lang.annotation.Retention;
10 import java.lang.annotation.Target;
11
12 import javax.enterprise.context.ApplicationScoped;
13 import javax.enterprise.inject.Stereotype;
14 import javax.inject.Named;
15
16 @Stereotype
17 @Target({ TYPE, METHOD, FIELD })
18 @Retention(RUNTIME)
19 @Documented
20
21 @Named
22 @ApplicationScoped
23 public @interface StereotypeDatabase {
24
25 }
```

5. Stereotypes Annotation

```
1 package com.innova.Stereotype;
2
3 import javax.enterprise.inject.Alternative;
4
5 @Alternative
6 public class DatabaseOfMySQL implements IDatabases{
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to MySQL.";
12     }
13 }
```

```
1 package com.innova.Stereotype;
2
3 import javax.enterprise.inject.Alternative;
4
5 @Alternative
6 public class DatabaseOfMsSql implements IDatabases{
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to MsSQL.";
12     }
13 }
```

```
1 package com.innova.Stereotype;
2
3 import javax.enterprise.inject.Alternative;
4
5 @Alternative
6 public class DatabaseOfPostgreSql implements IDatabases{
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to PostgreSQL.";
12     }
13 }
```

```
1 package com.innova.Stereotype;
2
3 import javax.enterprise.inject.Alternative;
4
5 @Alternative
6 public class DatabaseOfMariaDb implements IDatabases{
7
8     @Override
9     public String connectionDatabase()
10     {
11         return "Connected to MariaDb.";
12     }
13
14 }
```

```
1 package com.innova.Stereotype;
2
3 public interface IDatabases
4 {
5     String connectionDatabase();
6 }
```


Kaynak Kod : <https://github.com/gurkanguldas/InnovaCdiProject>