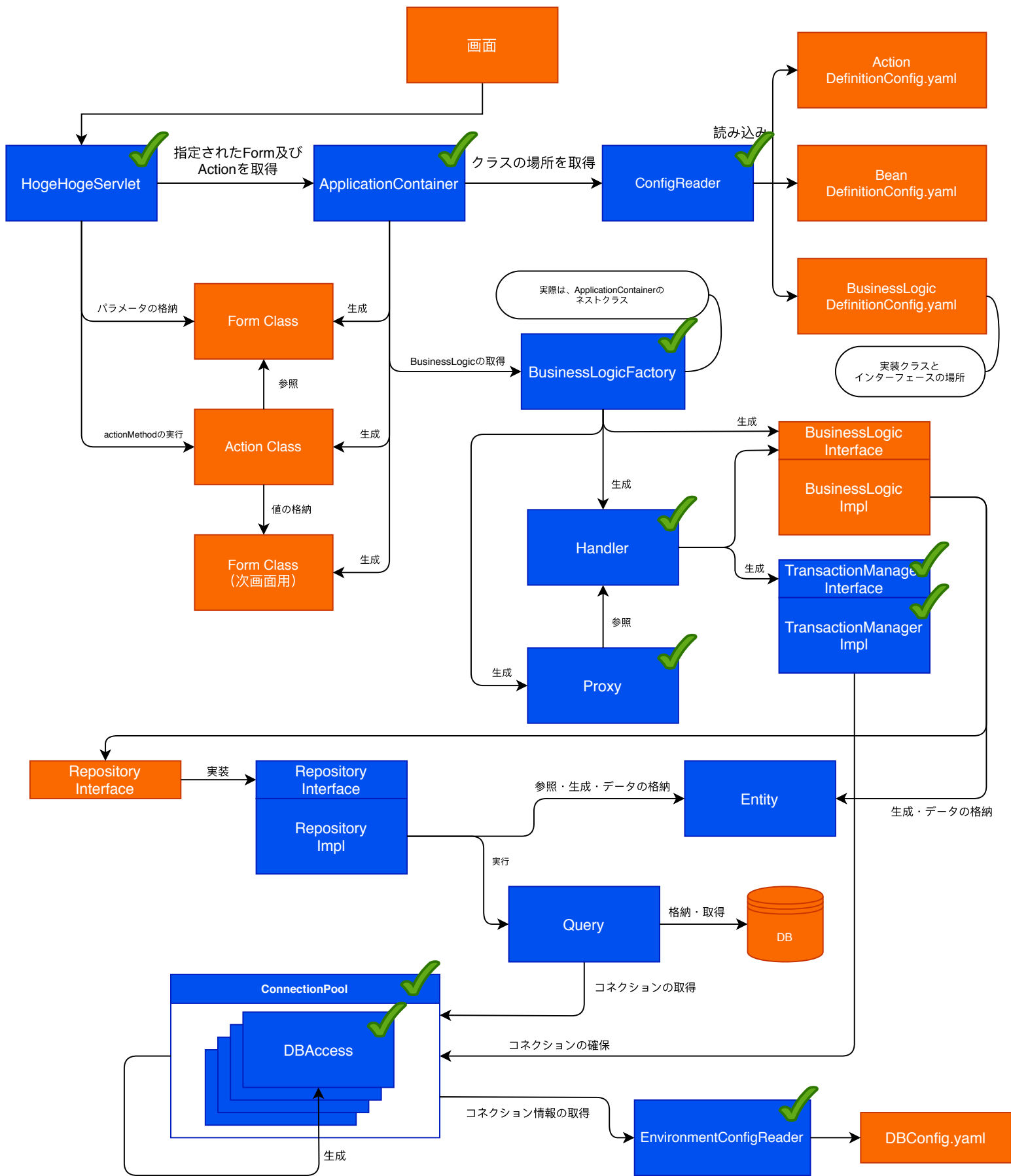


FrameWorkDesign(Overall view)



FrameWorkDesign(ユーザ開発分)①

画面

```
<!-- ServletはHogeHogeServletを指定 -->
<form action="HogeHogeServlet" method="post">

<!-- name属性に、バインドしたいFormの変数名を記載する-->
<input type="text" size="20" value="" name="i"></td>
<input type="text" size="5" value="" name="s"></td>

<!--
  formNameに、バインドしたいBeanの名前(設定ファイルに記載したもの)
  actionNameに、使用するActionクラスの名前を、
  actionMethodNameに、上記で指定したActionClassの実行したいメソッドを指定
-->
<input type="hidden" name="formName" value="TestC">
<input type="hidden" name="actionName" value="actionC">
<input type="hidden" name="actionMethodName" value="hogeHoge">

<input type="submit" name="button1" value="送信">
</form>
```

Action Class

```
public class ActionTestA {

    //BusinessLogicクラスにアノテーションをつける（複数でも可）
    @Service
    BusinessLogicTestA bl1;

    //Requestデータを格納したFormクラスを受け取る
    @FormInjection
    TestA testA;

    //ActionMethodの名前を指定
    @ActionMethod("testMethod")
    public void actionMethod() {
        System.out.println("アクションクラスのメソッドが実行されました！！！！");
        bl1.testMethod(testA);
    }

    //ActionMethodの名前を指定
    @ActionMethod("testMethod2")
    public void actionMethod2() {
        System.out.println("アクションクラスのメソッドが実行されました！！！！");
        bl1.testMethod(testA);
    }
}
```

FrameWorkDesign(ユーザ開発分)②

Form Class

Form Class
(次画面用)

```
package beans;

public class TestA {

    private String str1;

    private String str2;

    public String getStr2() {
        return str2;
    }

    public void setStr2(String str2) {
        this.str2 = str2;
    }

    public String getStr1() {
        return str1;
    }

    public void setStr1(String str1) {
        this.str1 = str1;
    }

}
```

生存期間を識別するアノテーション
をつける必要あり。

BusinessLogic
Interface



BusinessLogic
Impl

```
public interface AuthorizationService {
    //ログイン機能
    Result login(UserForm userForm);
}
```

```
public class AuthorizationServiceImpl implements AuthorizationService{

    @Repository
    private Repository repos;

    @Override
    public Result login(UserForm userForm) {
        System.out.println("ログイン処理の開始");

        UserEntity ue = repos.findByids(userForm);

        return null;
    }

}
```

FrameWorkDesign(ユーザ開発分)③

Repository
Interface

未実装

Action
DefinitionConfig.yaml

```
!!container.ActionDefinition
name: actionA
type: actions.ActionTestA
---
!!container.ActionDefinition
name: actionB
type: actions.ActionTestB
---
!!container.ActionDefinition
name: actionC
type: actions.ActionTestC
```

BusinessLogic
DefinitionConfig.yaml

```
!!container.BusinessLogicDefinition
interfaceClass: businessLogic.BusinessLogic
name: bl1
type: businessLogic.BusinessLogicTestA
---
!!container.BusinessLogicDefinition
interfaceClass: businessLogic.BusinessLogic
name: bl2
type: businessLogic.BusinessLogicTestB
---
!!container.BusinessLogicDefinition
interfaceClass: businessLogic.BusinessLogic
name: bl3
type: businessLogic.BusinessLogicTestC
---
!!container.BusinessLogicDefinition
interfaceClass: businessLogic.BusinessLogic
name: bl4
type: businessLogic.BusinessLogicTestD
```

Bean
DefinitionConfig.yaml

```
!!container.BeanDefinition
name: TestA
type: beans.TestA
---
!!container.BeanDefinition
name: TestB
type: beans.TestB
---
!!container.BeanDefinition
name: TestC
type: beans.TestC
```

DBInfo.yaml

```
#DBの設定
!!container.DBConfig
driver: com.mysql.cj.jdbc.Driver #driver名
url: jdbc:mysql://localhost:3306/TaskDB #DBのURL
user: yuma #DBのユーザ
password: password #DBのパスワード
numberOfAccess: 3
```

FrameWorkDesign(FrameWork機能)①

HogeHogeServlet

DivisionRepository

DivisionRepository

DivisionRepository

DivisionRepository

JPARepository

実装

