1)JAVA düyasındaki framework'ler veçözdükleri problemler nelerdir? Kod örneklerini de içermelidir

Framework Nedir?

Hazırlanmış olan kütüphanelerin mevcut olduğu ve bu kütüphanelere yenilerinin eklenebileceği yapıların adıdır. Çerçeve ve çatı gibi anlamlara gelen framework kelimesi, yazılımı yazarken yazlımın kılavuzluğunu, çerçevesini oluşturmayı kolaylaştırır. Yani oluşturulacak olan yazlımın temellerini atmak yerini yazılım yazmaya başlamaya olanak tanır. Yazılımın temeli dediğimiz yapı; yazılım kullanılırken hangi port üzerinden uygulama çalışacak, hangi ortamlarda çalışacak gibi yazılım ayağa kalkarken ihtiyacı olan temel yapıları örnek olarak verebiliriz.

Framework Kullanmanın Avantajları Nelerdir?

Framework kullanımının avantajlarını;

- Gereksiz ve yinelenen kod önlenebilir; kod daha güvenli haldedir.
- Kod tutarlılığı sağlar
- Fonksiyonel olarak sürekli gelişme sağlanır
- Kodun test edilmesi ve hatanın ayıklanması kolay bir şekilde yapılır. Kod sahibi olmayan geliştiricilerin bile yapması mümkündür**.**
- Uygulamanın geliştirilme süresi kısalır

Library ve Framework Arasındaki Farklar Nelerdir?

Library(kütüphane kullanımı) yazılımcıya özgürlük tanır ancak iş yükünü artırır çünkü yazılımcının tüm konfigürasyonları kendisinin yapması gerekir ki bu da yazılımcının kod yazma özgürlüğünü kazanırken daha fazla iş yükü alması anlamına gelir. Yazdıkları kodları uygulama içinde istedikleri yerde kullanabilir herhangi bir kurala tabi olmaz çünkü kuralı kendisi koyar.

Framework ise Library yapısının sunduğu özgürlükleri tanımaz. Framework kullanıyorsanız framework ün belirlediği kurallar üzerine kodunuzu yazmalısınız. Framework özgürlük sağlamaması kullanımının dezavantajları olduğu anlamına gelmez. Framework ile geliştirmeler de uygulamanın ihtiyacı olan kodu direkt olarak yazmanıza odaklanmanızı sağlar.

Java Dünyasında ki Frameworkler Nelerdir?

Java dünyasından birden çok framework olup en çok kullaınlan 10 tane framework aşağıdaki gibi olup bu frameworkler den 3 tanesini açıklayacağız;

- Spring
- Hibernate
- Struts
- Google web toolkit [GWT]
- JavaServer Faces [JSF]
- Grails
- Vaadin
- Blade
- Dropwizard
- Play

JavaServer Faces [JSF]

JavaServer Faces (JSF), Java tabanlı web uygulamaları için kullanıcı arayüzleri oluşturmak için Oracle tarafından geliştirilmiştir. Java Community Process(JCP) girişiminin resmi standardıdır. Oldukça kararlı bir çerçeve.

JSP kodları HTML dilinin içine yazılır ve kendine özel etiket sistemi vardır. Bu sayede HTML ile karışmadan kendi içinde güzel ve düzenli bir performans sunarak kaliteli siteler yapma konusunda yazılımcıya yardımcı olmaktadır.

Bu, bileşen tabanlı bir UI çerçevesidir. JSF, MVC yazılım tasarım modeline dayanmaktadır ve uygulama mantığı ile temsil arasındaki ayrımı tamamen tanımlayan bir mimariye sahiptir.

Kullanım Alanları:

- Bileşen tabanlı UI çerçeveleri
- Yerel uygulamalar oluşturmaya yardımcı olur

Avantajları;

- JSF, Java EE'nin ayrılmaz bir parçasıdır
- Mükemmel araçlar ve zengin kitaplıklar sağlar
- Yeni bir çerçeve sunarak temel uygulamayı değiştirmek zorunda kalmadan mevcut arka uç Java kodunun bir web arayüzü ile genişletilmesine olanak tanır.

Aşağıdaki örnekte PersonBean.class ını XHTML sayfasına yazdırma işlemini yapalım

```
package _01.hellojsf;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;


@ManagedBean

//@ManagedBean(name="person")

@SessionScoped
```

```
public class PersonBean {
	private String name;
	private String password;

	public PersonBean(String name, String password) {
		super();
		this.name = name;
		this.password = password;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
}
```

login.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html">
```

```
<h:head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <title>Welcome</title>
</h:head>
<h:body>
   <h:form>
      <h3>Please enter your name and password.</h3>
      <table>
         <tr>
            <td>Name:</td>
            <td><h:inputText value="#{personBean.name}" /></td>
         </tr>
         <tr>
            <td>Password:</td>
            <td><h:inputSecret value="#{personBean.password}" /></td>
         </tr>
      </table>
      <p>
         <h:commandButton value="Login" action="welcome" />
      </p>
   </h:form>
</h:body>
</html>
```

JSF Expression Language , JSP Expression Language'ten farklı olarak çalışır. Sayfa goruntulendiginde , **PersonBean** 'in getName metodunu çağırır , sayfada submit/gönderme işlemi olduğunda setName metodu çağrılmaktadır. JSF, setter metot'larını çağırmaktan sorumludur.

h:commandButton da action alanı page navigation için kullanılmaktadır. Burada **welcome.xhtml** url adresine navigation işlemi gerçekleşecektir.

```
<h:commandButton value="Login" action="welcome" />
```

welcome.xhtml

Burada dikkat etmemiz gereken nokta **#{personBean.name}** ifadesidir. Bu ifadeyi(Expression) h:inputText de value alanında da kullandık. h:inputText component için setName metodu çalışacaktır fakat burada #{personBean.name} ifadesi getName 'in

çalışmasını sağlar.
JSF, getter metotlarını çağırmaktan sorumludur.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html">
<h:head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <title>Welcome</title>
</h:head>
<h:body>
   <h3>Welcome to www.injavawetrust.com JSF tutorial!
      #{personBean.name}!</h3>
</h:body>
</html>
```

Servlet Konfigurasyonu

Java Server Faces , MVC (Model View Controller) pattern yapısı üzerine inşa edilmiştir. javax.faces.webapp.FacesServlet , istekleri karşılamak (handle request) ve sayfa yönlendirmesi (page navigation) görevlerini üstlenir. FacesServlet' in istekleri karşılayabilmesi icin web.xml dosyasında servlet tanımını yapmamız gereklidir.

web.xml

```
<servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
   </servlet>
```

<servlet> etiketi içerisinde **<servlet-name>** etiketi yer almaktadır. **<servlet-name>** de dilediğimiz ismi verebiliriz.
<sevlet-class> etiketi yardımı ile paket ismi.sinif ismini yazıyoruz.
<servlet-mapping> etiketinde **<servlet-name>** alanına url tanımlaması yapacağımız Servlet ismini veriyoruz. Burada dikkat ederseniz önceki tanımladığımız **<servlet-name>** etiketindeki isim ile ayni isme sahip yani Faces Servlet.
<url-pattern> etiketi yardımı ile Servlet'imiz icin url tanımlaması yapıyoruz. *.xhtml ; sonu xhtml ile biten tüm url adresleri için Faces Servlet çalıssın anlamına gelmektedir. *.xhtml seklindeki mapping e extension mapping denilmektedir.

```
  <servlet-mapping>
```

```
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.xhtml</url-pattern>
  </servlet-mapping>
```

Hibernate

Hibernate ORM, Java için kararlı object-relational mapping çerçevesidir. Java programlama dili ile ilişkisel veritabanı yönetim sistemleri (RDBMS- Relational Database Management Systems) arasında daha iyi iletişimi mümkün kılar. Hibernate bir ORM kütüphanesidir. Veritabanı üzerinde yapılan işlemleri kolaylaştıran bu framework için Java sınıflarının veritabanı dönüşümünü yapıyor diyebiliriz. Aynı zamanda veri çekme ve veri sorgulama işlerinde de oldukça yardımı dokunan bir Framework'tür.

Java gibi nesne yönelimli bir dil ile çalıştığınızda, Paradigma Uyuşmazlığı olarak da adlandırılan Nesne-İlişkisel Empedans Uyuşmazlığı(Object-Relational Impedance Mismatch) adlı bir sorunla karşılaşırsınız. Bunun nedeni, OO dillerinin ve RDBMS'lerin verileri farklı şekilde işlemesidir, bu da ciddi uyumsuzluk sorunlarına yol açabilir. Bu Hazırda Bekletme, size Java'nın uyumsuzluk sorunlarının üstesinden gelen bir çerçeve sağlar.

Yararları;

- Nesne yönelimli deyimi izleyerek kalıcı sınıflar geliştirmenize olanak tanır.
- Kodda çok küçük değişiklikler kullanarak herhangi bir veritabanıyla iletişim kurmanıza olanak tanır, nesneler ve ilişkisel kelimeler arasındaki boşluğu kapatır.
- Java varlıkları üzerinde veritabanı işlemini gerçekleştirmenizi sağlayan gelişmiş bir ORM çerçevesidir.

Avantajlar

- Taşınabilirlik, üretkenlik, sürdürülebilirlik
- Ücretsiz ve açık kaynak çerçevesi
- JDBC API'sinden çok sayıda tekrarlayan kodu kaldırır

Hibarnate Örnek;

Hibernate Maven Dependencies;

```
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.3.7.Final</version>
 </dependency>
 <dependency>
   <groupId>com.h2database</groupId>
   <artifactId>h2</artifactId>
   <version>1.4.200</version>
</dependency>
```

Veritabanı olarak H2 database kulandık burda konfigürasyonda istediğiniz DB konfigürasyonunu yapabilirsiniz.

Hibernate Configuration

hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">org.h2.Driver</property>
    <property name="hibernate.connection.url">jdbc:h2:mem:test</property>
    <property name="hibernate.connection.username">sa</property>
    <property name="hibernate.connection.password"></property>
    <property name="hibernate.dialect">org.hibernate.dialect.H2Dialect</property>
    <property name="show_sql">true</property>
    <property name="hbm2ddl.auto">create-drop</property>
    <mapping
class="com.howtodoinjava.hibernate.test.dto.EmployeeEntity"></mapping>
  </session-factory>
</hibernate-configuration>
```

Entity class; *EmployeeEntity.java*

```
package hibernate.test.dto;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;

import org.hibernate.annotations.OptimisticLockType;

@Entity
@org.hibernate.annotations.Entity(optimisticLock = OptimisticLockType.ALL)
@Table(name = "Employee", uniqueConstraints = {
    @UniqueConstraint(columnNames = "ID"),
    @UniqueConstraint(columnNames = "EMAIL") })
public class EmployeeEntity implements Serializable {

  private static final long serialVersionUID = -1798070786993154676L;

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "ID", unique = true, nullable = false)
  private Integer employeeId;

  @Column(name = "EMAIL", unique = true, nullable = false, length = 100)
  private String email;
```

```
  @Column(name = "FIRST_NAME", unique = false, nullable = false, length = 100)
  private String firstName;

  @Column(name = "LAST_NAME", unique = false, nullable = false, length = 100)
  private String lastName;

  // Accessors and mutators for all four fields
}
```

Hibernate Session Factory;

tüm konfigürasyon seçeneklerini ve entity tanımlarını hibernate.cgf.xml'de tanımlıyor ve SessionFactory'yi tek bir ifade de oluşturuyoruz.

```
package com.howtodoinjava.hibernate.test;

import java.io.File;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
  private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
            return new Configuration().configure(new
File("hibernate.cgf.xml")).buildSessionFactory();

        }
        catch (Throwable ex) {
            // Make sure you log the exception, as it might be swallowed
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void shutdown() {
      // Close caches and connection pools
      getSessionFactory().close();
    }
}
```

hibernate.cgf.xml dosyasının doğru yolunu kullanmayı unutmayın.

Demo; TestHibernate.java

```
package com.howtodoinjava.hibernate.test;

import org.hibernate.Session;

import com.howtodoinjava.hibernate.test.dto.EmployeeEntity;
```

```
public class TestHibernate {

  public static void main(String[] args) {
    Session session = HibernateUtil.getSessionFactory().openSession();
    session.beginTransaction();

    //Add new Employee object
    EmployeeEntity emp = new EmployeeEntity();
    emp.setEmail("demo-user@mail.com");
    emp.setFirstName("demo");
    emp.setLastName("user");

    session.save(emp);

    session.getTransaction().commit();
    HibernateUtil.shutdown();
  }
}
```

Console Görüntüsü;

```
Hibernate: drop table Employee if exists

Hibernate: create table Employee (ID integer generated by default as identity,
EMAIL varchar(100)
not null, FIRST_NAME varchar(100) not null, LAST_NAME varchar(100) not null,
primary key (ID))

Hibernate: alter table Employee add constraint UK_ardf0f11mfa6tujs3hflthwdv unique
(EMAIL)

Hibernate: insert into Employee (ID, EMAIL, FIRST_NAME, LAST_NAME) values (null,
?, ?, ?)

Hibernate: drop table Employee if exists
```

Struts

Bu, Apache Software Foundation (ASF) tarafından sürdürülen başka bir kurumsal düzeyde framework tür. Geliştiricilerin bakımı kolay bir Java uygulaması oluşturmasına olanak tanıyan bu tam özellikli bir Java web uygulaması framework dür. İki versiyonu var. Struts 1 ve Struts 2, OpenSymphony ve Struts 1'in webwork çerçevesinin birleşimidir. Ancak, Apache Struts'un yükseltilmiş versiyonu olduğu için tüm şirketler Struts 2'yi kullanmayı tercih ediyor.

Kullanım Alanları:

- Struts 2 Framework u, MVC tabanlı bir web uygulaması geliştirmek için kullanılır.
- Geliştiricilerin MVC mimarisini benimsemelerine yardımcı olmak için Java Servlet API'sini kullanır ve genişletir.

Yararları:

- Bu framework dokümantasyonu, aktif web geliştiricileri için yazılmıştır ve Java web uygulamalarının nasıl oluşturulduğuna dair bir çalışma bilgisi olduğunu varsayar.
- Geliştirme süresini kısaltır ve uygulamanın yönetilebilirliğini kolaylaştırır
- Merkezi Konfigürasyon sunar, yani bilgileri Java programlarına kodlamak yerine birçok Struts değeri XML veya özellik dosyalarında temsil edilir.
- Platformda oluşturulmayan görevleri gerçekleştirmek için Struts'u diğer Java framework leriyle entegre edebilirsiniz.

Struts framework kullanan bazı şirketler;

- Infosys
- Accenture
- NexGen Technologies

MessageStore.java

```
package org.apache.struts.helloworld.model;

public class MessageStore {
    private String message;

    public MessageStore() {
        message = "Hello Struts User";
    }

    public String getMessage() {
        return message;
    }
}
```

HelloWorldAction.java

```
package org.apache.struts.helloworld.action;

import org.apache.struts.helloworld.model.MessageStore;

import com.opensymphony.xwork2.ActionSupport;

public class HelloWorldAction extends ActionSupport {
    private MessageStore messageStore;
```

```
    public String execute() {
        messageStore = new MessageStore() ;

        return SUCCESS;
    }

    public MessageStore getMessageStore() {
        return messageStore;
    }
}
```

HelloWorld.jsp

```
<!DOCTYPE html>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Hello World!</title>
  </head>
  <body>
    <h2><s:property value="messageStore.message" /></h2>
  </body>
</html>
```

struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
                "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
                "http://struts.apache.org/dtds/struts-2.5.dtd">
<struts>
    <constant name="struts.devMode" value="true" />

    <package name="basicstruts2" extends="struts-default">
        <action name="index">
```

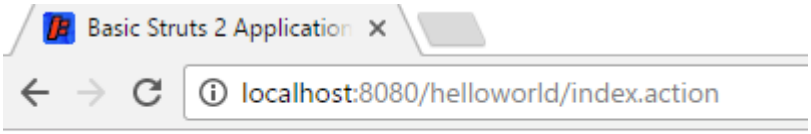
```
            <result>/index.jsp</result>
        </action>

        <action name="hello" class="org.apache.struts.helloworld.action.HelloWorld
Action" method="execute">
            <result name="success">/HelloWorld.jsp</result>
        </action>
    </package>
</struts>
```

index.jsp

```
<!DOCTYPE html>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Basic Struts 2 Application - Welcome</title>
    </head>
    <body>
        <h1>Welcome To Struts 2!</h1>
        <p><a href="<s:url action='hello'/>">Hello World</a></p>
    </body>
</html>
```

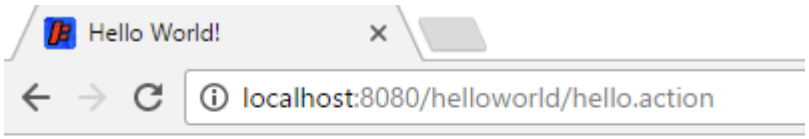
Uygulamaya çalıştırmak için `mvn jetty:run` çalıştırın
URL http://localhost:8080/helloworld/index.action



Welcome To Struts 2!

Hello World

Hello World tıklanırsa;



Hello Struts User

2)Spring Framework'ünün kullandığı design paternlar nelerdir?

- Inversion of Control (IOC)
- Dependency Injection (DI)
- Proxy Pattern
- Singleton Pattern
- Factory Pattern
- Decorator Pattern
- Strategy Pattern
- Conclusion

3)Creational patterns neler?Önceki ödevde oluşturulan nesnelerinizi Factory Design patterni ile oluşacak şekilde düzenleyin

6 tip creational design pattern bulunur;

1. Factory Method Pattern
2. Abstract Factory Pattern
3. Singleton Pattern
4. Prototype Pattern
5. Builder Pattern
6. Object Pool Pattern

1.Factory Method Pattern

Sınıf oluşturma üzerine bir pattern'dir. Sınıflar çağrılırken sürekli new lenmeme üzerine kurulmuş bir pattern'dir. Üst sınıflarda nesneler oluşturmak için arabirimler sağlar ve alt sınıfların bu nesneleri kendileri için kullanmalarını sağlayan Creational pattern yapısıdır.

```
public interface Car {

void name();
void since(int year);

}// Car genel bir sınıftır ve birden fazla car türü olabilir


public class Ford implements Car {

	@Override
	public void name() {
		System.out.println("Araba Markası Ford");
	}

	@Override
	public void since(int year) {
		System.out.println(year +  "  yaşında.");
	}

}//Burda Car sınıfında ki method lar override edilip yeni bir Car tür olan Ford class oluşturulmuş

public class Fiat implements Car {

	@Override
	public void name() {
		System.out.println("Araba Markası Fiat");
	}

	@Override
	public void since(int year) {
		System.out.println(year +  "  yaşında.");
	}

}//Burada Car sınıfında ki method lar override edilip yeni bir Car tür olan Fiat class oluşturulmuş


public class CarFactory {

	public static Car createCar(Class aClass) throws IllegalAccessException,
	InstantiationException {
		return (Car) aClass.newInstance();
	}
}// Burda CarFactory araba fabrikası olarak düşünebiliriz
```

```
public class Main {
	public static void main(String[] args) {
		try {
			Ford ford = (Ford) CarrFactory.createCar(Ford.class);
			asus.since(1234);
			ford.name();

			Fiat fiat = (Fiat) CarrFactory.createCar(Fiat.class);
			fiat.name();
			}
		catch (Exception e) {
			e.printStackTrace();
			}
	}
}
```

Yararlanılan Kaynaklar:

http://www.injavawetrust.com/hello-jsf-world/
https://www.edureka.co/blog/java-frameworks/#What_are_Java_frameworks?
https://www.niobehosting.com/blog/java/
https://howtodoinjava.com/hibernate/hibernate-hello-world-application/
https://struts.apache.org/getting-started/hello-world-using-struts2.html
https://hibernate.org/