CODESPRING

OUR CODE YOUR SUCCESS

JavaServer Faces

Rövid bevezető

Simon Károly
simon.karoly@codespring.ro

JSF

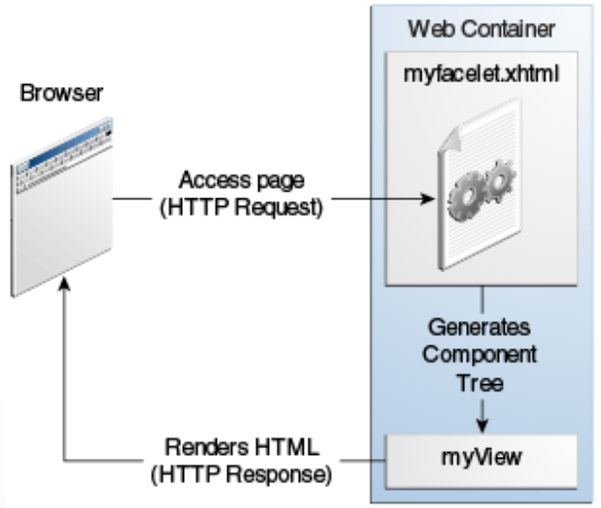
CODESPRING
OUR CODE YOUR SUCCESS

- Szerver oldali UI komponens keretrendszer + MVC
- Alapvető részei:
  - API: az UI komponensek reprezentációja és állapotuk menedzsmentje, eseménykezelés, szerver oldali validáció, adatkonverzió, navigáció meghatározása, nemzetköziesítés támogatása + kiterjeszthetőség
  - Elemkönyvtárak: UI komponensek weboldalakon belüli létrehozására, és összekapcsolására a szerver-oldali objektumokkal
- Lehetőségek:
  - Kliens oldali események összekapcsolása szerver oldali kóddal
  - UI komponensek hozzárendelése (binding) szerver oldali adatokhoz
  - Újrafelhasználható és kiterjeszthető UI komponensek
  - UI állapotának mentése és visszaállítása egy szerver request életciklusán belül
- Egyik legnagyobb előny: megjelenítés és vezérlés különválasztása. A JSP alkalmazásokkal ellentétben lehetőség van UI komponensek állapotának menedzsmentjére, eseménykezelésre.

JSF

CODESPRING
OUR CODE YOUR SUCCESS

- JSF alkalmazások részei
  - Weboldalak
  - JSF tag-ek (komponensek hozzáadása a weboldalakhoz)
  - Menedzselt bean-ek (backing bean-ek: UI komponensek tulajdonságainak és funkcionalitásainak meghatározására)
  - Telepítés leíró (web.xml)
  - Opcionális konfigurációs és erőforrás állományok (pl. faces-config.xml – navigációs szabályok, bean-ek konfigurációja, custom komponensek)
  - Opcionálisan további komponensek: validátorok, konverterek, figyelők, custom komponensek), custom tag-ek.



Forrás: oracle.com

- A myfacelet.xhtml weboldal JSF komponens elemek (tag-ek) segítségével van felépítve, komponenseket ad hozzá a nézethez, amely az oldal szerver oldali reprezentációja.
- A komponensek mellett az oldal még meghatározhat komponensekhez rendelt figyelőket, validátorokat és konvertereket, a komponensek funkcionalitásait megvalósító JavaBean-eket.
- A JSF közvetlenül a Servlet API-ra épül, a megjelenítés szempontjából nem limitál adott technológiára, bár a részét képező Facelet technológia a preferált.

Feladatkörök

- Oldalkészítők (page authors): jelölőnyelv alkalmazása (pl. HTML), grafikus designnal kapcsolatos tapasztalat. A JSF testreszabott elemkönyvtárainak alkalmazása.
- Fejlesztők (action developers): eseménykezelők, konverterek, validátorok, helper osztályok implementációja.
- Komponens fejlesztők (component writers): testreszabott UI komponensek készítése, saját UI osztályok segítségével, vagy a JSF komponensek kiterjesztésével.
- Tervezők (application architects): a web-alkalmazás architektúrájának megtervezése.
- Eszközkészítők (tool vendors): fejlesztői segédeszközök készítése.

Egyszerű JSF alkalmazás

- A fejlesztés lépései:
  - Oldalak elkészítése az UI komponensek és központi elemek felhasználásával.
  - Navigáció meghatározása a konfigurációs állományban.
  - Backing bean-ek implementációja.
  - Menedzselt bean-ek deklarációja a konfigurációs állományban.
- Példa (forrás Sun/Oracle): "GuessNumber" alkalmazás (egy 0 és 10 közötti számot kell kitalálni).

greeting.xhtml

CODESPRING
OUR CODE YOUR SUCCESS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en"
       xmlns="http://www.w3.org/1999/xhtml"
       xmlns:h="http://xmlns.jcp.org/jsf/html"
       xmlns:f="http://xmlns.jcp.org/jsf/core">
  <h:head>
       <h:outputStylesheet library="css" name="default.css"/>
       <title>Guess Number Facelets Application</title>
  </h:head>
  <h:body>
       <h:form>
          <h:graphicImage value="#{resource['images:wave.med.gif']}" alt="Duke waving his hand"/>
       <h2> Hi, my name is Duke. I am thinking of a number from
                 #{userNumberBean.minimum} to #{userNumberBean.maximum}. Can you guess it?
       </h2>
       <p><h:inputText          id="userNo"
                                title="Enter a number from 0 to 10:"
                                value="#{userNumberBean.userNumber}">
                 <f:validateLongRange        minimum="#{userNumberBean.minimum}"
                                             maximum="#{userNumberBean.maximum}"/>
       </h:inputText>
       <h:commandButton id="submit" value="Submit" action="response"/>
       </p>
       <h:message showSummary="true" showDetail="false"
                    style="color: #d20005;
                    font-family: 'New Century Schoolbook', serif;
                    font-style: oblique;
                    text-decoration: overline"
                    id="errors1"
                    for="userNo"/>
       </h:form>
  </h:body>
</html>
```

Facelet elemek

CODESPRING
OUR CODE YOUR SUCCESS

- Facelet html és core elemek
- form elem: input form, a szerkeszthető elemeket ilyenbe kell helyezni.
- inputText elem: szövegmező (esetünkben értéke – a value attribútum – össze van kötve a UserNumberBean userNumber attribútumával).
- commandButton elem: gomb, az action attribútum határozza meg a navigációt.
- message elem: üzenet (esetünkben hibaüzenet, ha a szövegmezőkbe írt értékek nem felelnek meg a LongRangeValidator implementációnak). A for attribútum határozza meg a komponenst, amelynek validációjával kapcsolatos az üzenet (a komponensnek kell rendelkeznie id azonosítóval, hogy hivatkozhassunk rá a for attribútumból).
- validateLongRange elem: LongRangeValidator kapcsolható komponensekhez. A validátor ellenőrzi, hogy a komponensen belül megadott értékek benne vannak-e egy adott intervallumba. Az intervallum határait a validateLongRange elemen belül adjuk meg (esetünkben a UserNumberBean mezőinek értékei).

response.xhtml

CODESPRING
OUR CODE YOUR SUCCESS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html lang="en"
        xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://xmlns.jcp.org/jsf/html">

  <h:head>
        <h:outputStylesheet library="css" name="default.css"/>
        <title>Guess Number Facelets Application</title>
  </h:head>

  <h:body>
        <h:form>
                <h:graphicImage value="#{resource['images:wave.med.gif']}"
                                                alt="Duke waving his hand"/>
                <h2>
                        <h:outputText id="result" value="#{userNumberBean.response}"/>
                </h2>
                <h:commandButton id="back" value="Back" action="greeting"/>
        </h:form>
  </h:body>
</html>
```

Navigáció meghatározása

- A konfigurációs állományon belül:

```
<navigation-rule>
	<from-view-id>/greeting.xhtml</from-view-id>
	<navigation-case>
		<from-outcome>success</from-outcome>
		<to-view-id>/response.xhtml</to-view-id>
	</navigation-case>
</navigation-rule>
<navigation-rule>
	<from-view-id>/response.xhtml</from-view-id>
	<navigation-case>
		<from-outcome>success</from-outcome>
		<to-view-id>/greeting.xhtml</to-view-id>
	</navigation-case>
</navigation-rule>
```

- Minden navigation-rule elem meghatározza, hogy egy adott oldalról hogyan navigálhatunk az alkalmazáson belüli más oldalakra. A különböző eseteket elkülönítő kondíciók a from-outcome elemekkel határozhatóak meg. Egy adott kimenet (outcome) a formot elküldő (submit) UICommand komponens action attribútumával határozható meg, vagy lehet a backing bean egyik action metódusának visszatérített értéke. Pl:

```
<h:commandButton id="submit" action="#{userNumberBean.getOrderStatus}" value="Submit" />
```

web.xml

CODESPRING
OUR CODE YOUR SUCCESS

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                             http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
      <context-param>
          <param-name>javax.faces.PROJECT_STAGE</param-name>
          <param-value>Development</param-value>
      </context-param>

      <servlet>
          <servlet-name>Faces Servlet</servlet-name>
          <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
          <load-on-startup>1</load-on-startup>
      </servlet>

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

      <session-config>
          <session-timeout> 30 </session-timeout>
      </session-config>

      <welcome-file-list>
          <welcome-file>greeting.xhtml</welcome-file>
      </welcome-file-list>

</web-app>
```

Backing bean-ek

CODESPRING
OUR CODE YOUR SUCCESS

```
package javaeetutorial.guessnumber;

import java.io.Serializable;
import java.util.Random;
import javax.enterprise.context.SessionScoped;
import javax.inject.Named;

@Named
@SessionScoped
public class UserNumberBean implements Serializable {

        private static final long serialVersionUID = 544335115139686872 4L;
        Integer randomInt = null;
        Integer userNumber = null;
        String response = null;
        private int maximum = 10;
        private int minimum = 0;

        public UserNumberBean() {
                Random randomGR = new Random();
                randomInt = new Integer(randomGR.nextInt(maximum + 1));
                // Print number to server log
                System.out.println("Duke's number: " + randomInt);
        }
```

Backing bean-ek

CODESPRING
OUR CODE YOUR SUCCESS

```
public void setUserNumber(Integer user_number) {
        userNumber = user_number;
}

public Integer getUserNumber() {
        return userNumber;
}

public String getResponse() {
        if ((userNumber == null) || (userNumber.compareTo(randomInt) != 0)) {
                return "Sorry, " + userNumber + " is incorrect.";
        } else {
                return "Yay! You got it!";
        }
}

public int getMaximum() {
        return (this.maximum);
}

public void setMaximum(int maximum) {
        this.maximum = maximum;
}

public int getMinimum() {
        return (this.minimum);
}

public void setMinimum(int minimum) { t
        his.minimum = minimum;
}
}
```

Backing bean-ek

- Általában minden JSF oldalhoz tartozik egy backing bean, amely az oldalon belül használt komponensek tulajdonságait és a kapcsolódó metódusokat (validáció, eseménykezelés, navigációval kapcsolatos feldolgozás) tartalmazza.
- A bean adott attribútumának értéke a value attribútum segítségével köthető egy adott komponenshez. A binding attribútum segítségével a komponens példány köthető hozzá egy bean tulajdonsághoz.
- A tulajdonságok esetében bármilyen primitív típust alkalmazhatunk, vagy bármilyen referencia típust, ha ahhoz létezik megfelelő konverter. A konverziót a JSF automatikusan végzi, de mi is megírhatunk konvertereket.
- Az oldalról, a komponensnek megfelelő elemen belül, a backing bean metódusaira is hivatkozni lehet.

Managed Bean-ek

- Annotációs mechanizmust alkalmazhatunk, vagy a konfigurációs állományban deklarálhatjuk a bean-eket.

```
<managed-bean>
    <description>
      The backing bean that backs up the guessNumber Web application
    </description>
    <managed-bean-name>UserNumberBean</managed-bean-name>
    <managed-bean-class>javaeetutorial.guessNumber.UserNumberBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
      <managed-property>
        <property-name>minimum</property-name>
        <property-class>long</property-class>
        <value>0</value>
      </managed-property>
      <managed-property>
        <property-name>maximum</property-name>
        <property-class>long</property-class>
        <value>10</value>
      </managed-property>
  </managed-bean>
```

- A példányt a rendszer az első hivatkozásnál inicializálja és a session-ben tárolja (azok a tulajdonságok amelyeknek nem adtunk kezdő értéket a konstruktornak megfelelően lesznek inicializálva).

UI komponens modell

CODESPRING
OUR CODE YOUR SUCCESS

- A JSF biztosít:
  - UIComponent osztályokat UI komponensek állapotának és viselkedésének meghatározására.
  - Renderelési modellt a megjelenítéshez.
  - Eseményfigyelők használatát és eseménykezelést támogató modellt.
  - Átalakítási modellt, amely lehetővé teszi, hogy adatkonvertereket regisztrálhassunk a komponensekhez.
  - Validációs modellt, amely validátorok regisztációját teszi lehetővé.
- Komponens osztályok:

  UIData (DataModel példány által reprezentált adatkollekció), UIColumn (az UIData egy oszlopa), UICommand (esemény forrása lehet), UIForm (a HTML form-hoz hasonló), UIGraphics (kép), UIInput (beviteli mező), UIMessage (lokalizált üzenet), UIOutput (kimenet megjelenítése az oldalon), UIPanel (tároló), UISelectBoolean (checkbox), UISelectItems (choice), UISelectItem (egy item a choice-on belül), UISelectMany (több item kiválasztása), UISelectOne (egy item kiválasztása), UIViewRoot (a komponens hierarchia gyökere).
- Többféleképpen renderelhetőek (pl. egy UICommand lehet gomb vagy link).
- Interfészek (a viselkedés meghatározása):

  ActionSource (a komponens esemény forrása lehet), EditableValueHolder (szerkeszthető komponens), NamingContainer (tároló amelynek minden gyereke azonosítóval rendelkezik), StateHolder (a komponens állapottal rendelkezik, amelyet meg kell őrizni két kérés között), ValueHolder (értéket tárol).
- Ezeket közvetlen módon csak új komponensek implementációjánál használjuk.

Renderelés

CODESPRING
OUR CODE YOUR SUCCESS

- A komponens osztályok a komponens viselkedését, funkcionalitását határozzák meg, a renderelés külön osztályok feladata.
- A renderelő csomag (rendering kit) határozza meg, hogy a komponensek hogyan lesznek megjelenítve. Minden komponens osztályhoz tartozik egy vagy több Renderer osztály. Pl. egy UISelectOne komponenshez háromféle renderer tartozik: lehet radio button, combo box, vagy list box.
- A JSF elemek meghatározzák az osztályt és a renderert is. Pl. commandButton, commandLink.
- Komponenseknek megfelelő alapvető elemek (a megfelelő HTML elemekként vannak renderelve):

  column, commandButton, commandLink, dataTable, form, graphicImage, inputHidden, inputSecret, inputText, inputTextArea, message, messages, outputLabel, outputLink, outputFormat, outputText, panelGrid, panelGroup, selectBooleanCheckbox, selectItem, selectItems, selectManyCheckbox, selectManyListbox, selectManyMenu, selectOneListbox, selectOneMenu, selectOneRadio, stb.
- Külső eszköztárak használatának lehetősége.

Konverzió és validáció

CODESPRING
OUR CODE YOUR SUCCESS

- Konverzió
  - Egy komponens szerver oldali objektummal (backing bean) való összekapcsolása esetén két különböző "nézetről" beszélhetünk: a model nézetről (pl. java.util.Date) és a prezentációs nézetről (pl. egy mm/dd/yy string).
  - A konverziót a JSF automatikusan valósítja meg.
  - Amennyiben speciális konverziót szeretnénk, az UIOutput osztály lehetőséget ad Converter objektumok regisztrálására.
- Validáció:
  - A JSF Validator implementációkat biztosít a bemeneti komponensek értékeinek validálásához.
  - Saját validátorokat hozhatunk létre a Validator interfész implementációjával, vagy a backing bean-ekben implementált validációs metódusok segítségével.
  - A létrehozott validátorokat regisztrálnunk kell az alkalmazáson belül, és a validator tag segítségével regisztrálnunk kell őket az adott komponensnél (vagy custom tag-et kell létrehoznunk).

Események és figyelők

CODESPRING
OUR CODE YOUR SUCCESS

- Egyes komponensek eseményeket generálhatnak. Az eseménnyel kapcsolatos információk Event objektumokban vannak tárolva. A figyeléshez az alkalmazásnak Listener példányokat kell regisztráljon a forrás komponenseknél.
- JSF eseménytípusok: érték változása (UIInput interfészt implementáló komponens tartalmának megváltozása), action event-ek (ActionSource interfészt implementáló komponens aktiválása), adatmodellel kapcsolatos események (UIData komponens sorának kiválasztása).
- A figyelés két módon történhet: figyelő hozzárendelése valueChangeListener, vagy actionListener elem beágyazásával a komponensnek megfelelő elembe, illetve a backing bean adott metódusának implementálásával, majd a komponensnek megfelelő elemen belül a metódus komponenshez történő csatolásával (method binding).
- Az események kezelése a komponens immediate tulajdonságának függvényében történhet az "invoke application" vagy "apply request" fázisban az action event-ek esetében, a "process validation" vagy az "apply request" fázisban az értékváltozások esetében.

Életciklus

- Nézet helyreállítási fázis (restore view phase): akkor kezdődik, amikor egy kérés érkezik egy JSF oldalhoz (pl. klikk buttonra vagy linkre). A JSF felépíti a nézetet, hozzákapcsolja a komponensekhez a megfelelő objektumokat (figyelők, validátorok stb.) és elmenti a nézetet a FacesContext példányba. Ha első kérés volt, akkor egy üres nézet épül fel és továbblépünk a "render response" fázisba. Ha "postback" kérés volt, akkor az oldalnak megfelelő nézet már létezik, így a JSF az elmentett állapotinformációk alapján visszaállíthatja.
- Kérés értékek alkalmazásának fázisa (Apply Request Values Phase): a visszaállításkor minden komponens kiolvassa a neki megfelelő értéket a kérés objektumból (a decode metódus segítségével). Ha a konverzió nem működik, hibaüzenet generálódik, amely a FacesContext-be kerül (a render response fázisban lesz megjelenítve a validációval kapcsolatos esetleges hibaüzenetekkel együtt). Ha események keletkeznek ebben a fázisban a JSF továbbítja ezeket a figyelőknek. Ha vannak komponensek, amelyeknek az immediate attribútuma true, az azoknak megfelelő validáció, konverzió és eseménykezelés már ebben a fázisban megtörténik. Ha valamelyik decode metódus vagy figyelő meghívja a FacesContext renderResponse metódusát, a JSF továbblépik a render response fázisba. Ha az alkalmazásnak ennél a pontnál át kell irányítania másik web komponenshez, vagy olyan választ kell generálnia, amelyik már nem tartalmaz JSF komponenseket a FacesContext.responseComplete metódusát hívhatja meg.

Életciklus

- Validálási fázis (process validations phase): az előző fázis végén a komponenseknek megfelelő értékek be lesznek állítva, és az üzenetek, illetve események el lesznek tárolva. Ebben a fázisban a JSF végrehajtja az összes regisztrált validátornak megfelelő műveleteket. Ha hiba lép fel az üzenet a FacesContext példányba kerül, és továbblépünk a render response fázisra. Ha a validate műveletek vagy figyelők renderResponse metódust hívnak hasonlóan továbblépünk. Ha események lépnek fel ebben a fázisban azok továbbítva lesznek a megfelelő figyelőknek.
- Modell értékek frissítésének fázisa (update model values phase): amennyiben a validáció sikeres a rendszer frissítheti a (bemeneti komponenseknek megfelelő) szerver oldali objektumok tartalmát. Ha a konverzió hibát ad továbblépünk a render response fázisra. Ha valamelyik updateModels metódus vagy figyelő renderResponse metódust hív, szintén továbblépünk. Ha események lépnek fel ebben a fázisban azok továbbítva lesznek a megfelelő figyelőknek.
- Alkalmazás meghívásának fázisa (invoke application phase): a JSF kezeli az alkalmazás szintű eseményeket (form elküldése, link más oldalhoz). Ha az oldal állapotinformációk alapján volt visszaállítva és egyes komponenseknél események léptek fel, ezek továbbítva lesznek a megfelelő figyelőkhöz.

Életciklus

- Render Response Phase: a JSF felépíti a nézetet és az oldal renderelését kéri. Ha első kérésről van szó, a komponensek hozzá lesznek adva a nézetnek megfelelő komponens hierarchiához (component tree). Ha a kérés "postback" az előzőleg felépített hierarchia szerint lesz renderelve az olal. Ha hibák léptek fel az előzőekben, amennyiben az oldal tartalmaz message elemeket, a hibaüzenetek meg lesznek jelenítve az oldalon (a message elemeknek megfelelő helyen). Miután a nézet renderelése megtörtént, az állapot el lesz mentve, így további kérések hozzáférhetnek.