JSF 2

- najznačajnije unapređenje kompletnog JSF okruženja dolazi sa pojavom verzije 2.0
- u ovoj verziji ispravljeni su nedostaci otkriveni praktičnom primenom i uvedene su značajne promene u logici razvoja Web aplikacija
- specifična priroda unesenih promena arhitekture za posledicu ima paralelnu egzistenciju dve verzije okruženja i nakon objavljivanja konačne specifikacije nove verzije
- određene osobine onemogućavaju jednostavno unapređenje postojećih aplikacija, pa i dalje postoje primene u kojima se starija verzija JSF okruženja zadržala kao dominantna

JSF 2 vs JSF 1.2

- najznačajniju izmenu u novoj verziji specifikacije predstavlja prenos kompletne arhitekture sistema u domen *Java Enterprise* web aplikacija (u skladu sa J2EE specifikacijom)
- ovom promenom su JSF aplikacijama stavljene na raspolaganje funkcionalnosti koje nisu deo same JSF specifikacije, već postoje u okviru J2EE okruženja

JSF 2 vs JSF 1.2

- podrška za anotacije unutar *bean*-ova
- uvođenjem podrške za anotacije unutar *bean*-ova omogućava se deklarisanje opsega važenja *Managed bean*-ova i izvan *faces-config* datoteke
  - *Managed bean*-ove je i dalje moguće registrovati u *faces-config* datoteci

```
// JSF 2.0
@ManagedBean(name="testBean")
@SessionScoped
public class TestBean {
...
}

// JSF 1.X
<managed-bean>
    <managed-bean-name>testBean</managed-bean-name>
    <managed-bean-class>TestBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

- ovakav vid „decentralizacije“ programske logike olakšava dalje unapređenje aplikacije i smanjuje njenu kompleksnost

JSF 2 vs JSF 1.2

- podrška za anotacije unutar *bean*-ova
- osim postojećih opsega dostupnosti *bean*-ova (opseg aplikacije, opseg sesije i opseg zahteva), uvedeni su i:
  - *View Scope*,
  - *Flash Scope* i
  - *Custom Scope*.
- opseg dostupnosti *View* i *Flash Scope Managed bean*-ova veći je od opsega zahteva, a manji od opsega sesije
- *Custom Scope* se specificira korišćenjem EL (*Expression Language*) izraza, a ne ključne reči

```
<managed-bean>
  <managed-bean-name>testBean</managed-bean-name>
  <managed-bean-class>TestBean</managed-bean-class>
  <managed-bean-scope>#{someCustomScope}</managed-bean-scope>
</managed-bean>
```

JSF 2 vs JSF 1.2

- navigacija – implicitna navigaciona pravila
- navigaciona pravila u tom slučaju ne moraju biti eksplicitno navedena, već se na bazi rezultata *action controller* metoda implicitno mapira odgovarajući pogled (*view*) u formi fajla sa odgovarajućim imenom na fajl sistemu
- implicitno navigaciono pravilo se koristi u slučaju kada ne postoji eksplicitno navigaciono pravilo
- mogućnosti iz prethodne verzije ovim dodacima nisu izbačene, pa je odluka o vrsti navigacije koja se koristi ostavljena programeru

```
public String test() {
  if (Math.random() < 0.5) {
    return("first");                    => first.xhtml
  } else {
    return("second");                   => second.xhtml
  }
}
```

JSF 2 vs JSF 1.2

- navigacija – uslovna navigaciona pravila
- drugo poboljšanje navigacionog podsistema ogleda se u uslovnim navigacionim pravilima
- uslovna navigacija se implementira kao EL izraz korišćenjem novog <if> konfiguracionog elementa

```
<navigation-rule>
   <display-name>page1.xhtml</display-name>
   <from-view-id>/page1.xhtml</from-view-id>
   <navigation-case>
       <from-outcome>success</from-outcome>
       <to-view-id>/page2.xhtml</to-view-id>
       <if>#{testBean.someCondition}</if>
   </navigation-case>
</navigation-rule>
```

JSF 2 vs JSF 1.2

- navigacija – mehanizam navigacije sa izvorišne na odredišnu stranu
- JSF 1.x specifikacija definiše *forward* serverske strane kao mehanizam navigacije sa izvorišne na odredišnu stranu
- pored ovog mehanizma navigacije, JSF 2.0 specifikacija definiše i redirekciju strane kao mehanizam navigacije sa izvorišne na odredišnu stranu
- kao podrazumevani mehanizam navigacije koristi se *forward* serverske strane, dok se redirekcija aktivira dodavanjem “faces-redirect=true” stringa na kraj izlaznog stringa
- drugi način aktiviranja redirekcije jeste korišćenje <redirect/> elementa u okviru <navigation-case> elementa navigacionog pravila

JSF 2 vs JSF 1.2

- navigacija – mehanizam navigacije sa izvorišne na odredišnu stranu
- drugi način aktiviranja redirekcije jeste korišćenje <redirect/> elementa u okviru <navigation-case> elementa navigacionog pravila

```
<navigation-rule>
   <display-name>page1.xhtml</display-name>
   <from-view-id>/page1.xhtml</from-view-id>
   <navigation-case>
       <from-outcome>success</from-outcome>
       <to-view-id>/page2.xhtml</to-view-id>
       <redirect />
   </navigation-case>
</navigation-rule>
```

JSF 2 vs JSF 1.2

- template-ing
- *Facelets* biblioteka je zbog svojih dobrih osobina uvedena kao osnovni okvir za implementaciju interfejsa prema krajnjim korisnicima, čime su uvedene značajne izmene u načinu projektovanja prezentacionog sloja web aplikacije
- *Facelets* biblioteka je razvijena kako bi se JSF aplikacijama omogućio *templating* mehanizam, tj. mehanizam boljeg i efikasnijeg iskorišćenja postojećeg programskog koda "razbijanjem" kompletne strane na delove koji se mogu iskoristiti u identičnoj formi na više strana u okviru iste aplikacije (zaglavlja strane, podnožja strane, meniji itd.).

JSF 2 vs JSF 1.2

- template-ing
- u JSF verziji 1, postojalo je više sličnih biblioteka (*Tapestry, Tiles*) od kojih nijedna nije bila standardizovana u okviru JSF specifikacije
- zbog uočenih prednosti *Facelets* biblioteke nad ostalim bibliotekama iste namene, ona je uvršćena u samu specifikaciju donoseći mogućnosti *templating*-a u okviru standardne specifikacije
- JSP podrška za izgradnju korisničkog interfejsa i dalje je zadržana u JSF 2.0 specifikaciji, s tim da novouvedene osobine u formi novih tagova uvedenih u JSF 2.0 nisu podržane
  - primer: nije moguće koristiti ugrađenu AJAX podršku ili korisnički definisane komponente

JSF 2 vs JSF 1.2

- ugrađena AJAX podrška
- podrška za AJAX komponente predstavlja značajno unapređenje u novoj verziji JSF specifikacije
- JSF 1.x aplikacije su AJAX funkcionalnosti preuzimale iz velikog broja postojećih biblioteka koje su predstavljale nadogradnju osnovne arhitekture
- JSF 2.0 donosi integrisanu podršku za AJAX funkcionalnosti, najviše u domenu parcijalnog osvežavanja strane i sličnih aspekata asinhrone komunikacije

```
<h:inputText value="#{testBean.text}">
    <f:ajax execute="@form" event="keyup" render="result"/>
</h:inputText>
```

JSF 2 vs JSF 1.2

- ugrađena AJAX podrška
- pored ugrađene AJAX podrške, zadržana je i mogućnost integracije spoljašnjih komponenata iz bilo koje od biblioteka koje podržavaju novu specifikaciju
- bitno je napomenuti da, zbog prethodno navedenih razlika arhitektura sistema, postoje problemi u kompatibilnosti AJAX biblioteka razvijenih za starije aplikacije u odnosu na zahteve koje postavlja nova specifikacija
- broj komponenata koje se mogu iskoristiti u novim aplikacijama je u konstantnom porastu i već uveliko prelazi minimum koji je potreban za implementaciju čak i naprednijih aplikacija, iako se radi o relativno novoj specifikaciji

JSF 2 vs JSF 1.2

- podrška za debug-ovanje
- razvoj moderne web aplikacije nije jednostavan, pa je podrška za *debug*-ovanje u fazi razvoja aplikacije je veoma značajan aspekt aplikativnog okruženja
- JSF 2.0 specifikacija uvodi PROJECT_STAGE parametar u *web.xml* konfiguracionoj datoteci aplikacije koji može imati vrednosti: *Production*, *Development*, *UnitTest*, *SystemTest* i *Extension*

```
<context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
</context-param>
```

- korišćenjem ovog parametra mnoge greške koje bi kod JSF 1.x aplikacija ostale neprimećene, sada mogu biti jednostavno detektovane

JSF 2 vs JSF 1.2

- korišćenje JSF EL za ispisivanje vrednosti property-ja bean-ova
- prema ranijoj specifikaciji ovo nije bio slučaj
- primer korišćenja *outputText* komponente u JSF 1.x i korišćenje JSF EL za ispis vrednosti *property*-ja *bean*-a

```
// JSF 2.0
#{ userBean.usernameProperty }

// JSF 1.x
<h:outputText value="#{userBean.usernameProperty}"/>
```

- korišćenje *outputText* komponente u JSF 2.0 aplikacijama je i dalje omogućeno, s tim što se ona obavezno mora koristiti u slučaju kada ju je potrebno opciono prikazivati ili kada je potrebno ispisivati HTML kod u JSF stranicu.

JSF 2 vs JSF 1.2

- jednostavnije kreiranje korisničkih komponenti
- veoma bitno poboljšanje koje donosi JSF 2.0 specifikacija
- podrška za implementaciju korisničkih komponenti koja je postojala u JSF 1.x aplikacijama bila je veoma značajna jer je dovela do kreiranja velikog broja biblioteka, poput *RichFaces, IceFaces, Tomahawk, WebGalileo* i ADF
- ovaj API u verziji JSF 1.x bio je veoma komplikovan
- JSF 2.0 specifikacija uvodi novi pristup u implementaciji korisničkih komponenti

JSF 2 vs JSF 1.2

- jednostavnije kreiranje korisničkih komponenti
- novi pristup baziran je na *facelets*-ima i omogućava relativno jednostavno kreiranje novih jednostavnih i srednje komplikovanih komponenti

```
<!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"
      xmlns:composite="http://java.sun.com/jsf/composite">
<head>
<title>Test Composite Component</title>
</head>
<body>
<composite:interface>
   <composite:attribute name="test"/>
</composite:interface>
<composite:implementation>
   <h:outputText value="Hello, #{cc.attrs.test}!"/>
</composite:implementation>
</body>
</html>
```

JSF 2 vs JSF 1.2

- podrška za GET zahteve
- JSF 2.0 specifikacija uvodi i značajniju podršku za GET zahteve
- ova podrška uključuje i nove komponente <h:link> i <h:button>

```
<h:link outcome="success">
  <f:param name="param1" value="abc"/>
</h:link>
```

Anotacije

- @ManagedBean anotacija

```
@ManagedBean
public class TestBean{
    private String abc;
    ...
}
```

- navođenje: #{testBean.abc}, gde naziv bean-a odgovara nazivu klase, s tim što je prvo slovo naziva malo slovo
- request scope je podrazumevani scope
- abc je naziv property-ja (koji ima odgovarajuće getere i setere) ili naziv metode

- ako action controller metoda vraća string “xyz”, i ako ne postoje eksplicitno definisana navigaciona pravila, onda je rezultujuća strana xyz.xhtml

Anotacije

- name atribut @ManagedBean-a

```
@ManagedBean(name=“testBean2")
public class TestBean{
    private String abc;
    ...
 }
```

- navođenje: #{testBean2.abc}, gde je testBean2 vrednost name atributa
- request scope je i dalje podrazumevani scope

Anotacije

- eager atribut @ManagedBean-a

```
@ManagedBean(eager=true)
@ApplicationScoped
public class TestBean{
    ...
    ...
}
```

- ako je “eager=true” i scope je application, onda bean mora biti kreiran u trenutku učitavanja aplikacije, a ne u trenutku prvog referenciranja bean-a, tj. pre opsluživanja bilo kojeg zahteva

```
<managed-bean eager="true">
<managed-bean-name>cbbhBean</managed-bean-name>
<managed-bean-class>net.etfbl.s_cube.external.beans.CBBHBean</managed-bean-class>
<managed-bean-scope>application</managed-bean-scope>
</managed-bean>
```

Anotacije

- scope
- @RequestScoped
  - podrazumevani
  - nova instanca se kreira pri svakom HTTP zahtjevu
- @SessionScoped
  - scope sesije
  - korisnik sa istim cookie-jem, ako sesija nije istekla, uvek pristupa istom sesijskom bean-u
  - bean bi trebao biti serijalizabilan
- @ApplicationScoped
  - scope aplikacije
  - dele ga svi korisnici aplikacije
  - ovaj bean ili nema stanje ili je potrebno izvršiti sinhronizaciju pristupa

Anotacije

- scope
- @ViewScoped
  - ista instanca bean-a se koristi dok god je korisnik na istoj stranici, npr. u slučaju AJAX zahteva
  - trebao bi biti serijalizabilan
- @CustomScoped(value="#{someMap}")
  - bean je smešten u Map objekat, i programer upravlja njegovim životnim vekom
- @NoneScoped
  - bean nije smešten u scope
  - korisno u slučajevima kada bean treba biti referenciran od strane drugih bean-ova koji su u nekom od scope-ova

Anotacije

- scope
- obično se smešta iza @ManagedBean

```
@ManagedBean
@SessionScoped
public class TestBean {
 …
}
```

Anotacije

- @ManagedProperty
- JSF podržava dependency injection
  - moguće je dodeliti vrednosti property-ju managed bean-a, bez hard kodovanja u definiciji klase
- način I:

  ```
  @ManagedProperty(value="#{someBean}")
  private SomeType someField;
  ```
- način II:
  - <managed-property> u faces-config.xml datoteci
  - isto kao i u verziji JSF 1.*x*

- setter metoda za property je obavezna

  ```
  setSomeField
  ```
- ako ime setter metode ne odgovara imenu property-ja, moguće je koristiti “name” atribut @ManagedProperty-ja

AJAX

- f:ajax

```
<h:inputText value="#{testBean.text}">
<f:ajax execute="@form" event="keyup"
   render="result"/>
</h:inputText>
...
<h:outputText value="#{testBean.text}" id="result"/>
```

AJAX

- f:ajax atributi

- render
  - id elemenata ili id-evi liste elemenata koji se trebaju osvežiti (izmeniti u DOM-u) nakon izvršavanja AJAX zahteva
  - često je to h:outputText
  - specijalne vrednosti @this, @form, @none i @all – obično se ne koriste kao vrednosti render atributa
- execute
  - elementi koji se šalju na serversku stranu radi procesiranja
  - često input elementi kao što je h:inputText ili h:selectOneMenu
  - @this – element koji okružuje f:ajax – podrazumevano
  - @form – h:form koji okražuje f:ajax – kada je potrebno da se pošalju vrednosti svih parametara forme
  - @none – ništa se ne šalje
  - @all – svi JSF UI elementi sa stranice
- event
  - DOM event na koji se aktivira AJAX zahtev (npr., keyup, blur)
- onevent
  - JavaScript funkcija koja se pokreće kada se “okida” event

Looping

- rad sa sadržajem promenljive veličine
- mogućnosti:
  - bean metoda koja vraća string ili HTML kod
  - h:dataTable
  - korisnička kompozitna komponenta
  - korišćenje ui:repeat

Looping

- rad sa sadržajem promenljive veličine
- bean metoda koja vraća string ili HTML kod
  - kolekcija se pretvara u string ili HTML
  - korisno
    - kada je izlaz jednostavan tekst ili veoma jednostavan HTML
    - kada izlaz ne koristi CSS
  - nije korisno
    - kada je potrebna veća kontrola nad izlazom

Looping

- rad sa sadržajem promenljive veličine
- h:dataTable
  - ugrađena komponenta pretvara kolekciju u HTML tabelu
  - korisno
    - kada je potrebno kreirati HTML tabelu na osnovu podataka sadržanih u kolekciji
  - nije korisno
    - kada je potrebno kreirati nešto što nije HTML tabela
    - kada različiti delovi tabele dolaze iz različitih izvora

Looping

- rad sa sadržajem promenljive veličine
- korisnička kompozitna komponenta
  - korisnička komponenta pretvara kolekciju u HTML izlaz
  - korisno
    - kada je potrebna kreirati nešto što nije HTML tabela na osnovu podataka sadržanih u kolekciji
  - nije korisno
    - kada su podaci u formatu drugačijem od onog koji je očekivan
    - kada je potrebno napraviti izmenu u grafičkom dizajnu (makar i najmanju)

Looping

- rad sa sadržajem promenljive veličine
- korišćenje ui:repeat
  - facelets looping za kreiranje HTML-a unutar stranice
  - korisno
    - kada je potrebna veća kontrola nad izlazom
    - kada jedna od prethodnih opcija nije odgovarajuća
  - nije korisno
    - kada HTML kod stranice postaje suviše kompleksan

Looping

- rad sa sadržajem promenljive veličine
- korišćenje ui:repeat
  - uključiti JSTL 1.2 u CLASSPATH
  - uključiti facelets namespace xmlns:ui="http://java.sun.com/jsf/facelets"
  - koristiti ui:repeat isto kao i JSTL c:forEach
    - razlika: c:forEach se izvršava kada se stablo komponenti gradi, a ui:repeat kada se stablo renderuje

```
<ui:repeat var="x” value="#{testBean.collection}">
...<HTML kod>...
#{x.someProperty}
...</HTML kod>...
</ui:repeat>
```

Looping

- rad sa sadržajem promenljive veličine
- atributi ui:repeat
  - var – ime kojim će biti referenciran element kolekcije
  - value – EL izraz koji specificira kolekciju
  - varStatus – ime kojim će biti referenciran status objekt (korisne osobine ovog objekta: index, first/last, even/odd)
  - offset – specificira početni element kolekcije – podrazumevana vrednost je 0
  - size – specificira poslednji element kolekcije – podrazumevana vrednost je poslednji element kolekcije
  - step – specificira korak pri prolasku kroz kolekciju – podrazumevana vrednost je 1

Korisničke kompozitne komponente

- definisanje komponente
  - abc.xhtml smešta se u “resources/xyz” folder
  - dostupni atributi se definišu u composite:interface
  - izlaz se specificira u composite:implementation
  - u fajlu komponente koristi se composite namespace

    xmlns:composite="http://java.sun.com/jsf/composite"

- u facelets stranici koristi se component namespace

  xmlns:scube=http://java.sun.com/jsf/composite/scubecomponents

- nakon toga, moguće je koristiti komponentu u facelets stranici

  <scube:cbbhcurrencyexchange panelStyleClass=“exchange" />

Korisničke kompozitne komponente

```
<composite:interface>
<composite:attribute name="panelStyleClass"/>
</composite:interface>

<composite:implementation>
<h:panelGrid styleClass="#{cc.attrs.panelStyleClass}">
<rich:panel header="#{msgs.cbbh_currency_exchange}">
<h:form>
<h:panelGrid columns="3" styleClass="#{cc.attrs.panelStyleClass}" frame="border">
<h:outputText value="#{msgs.cbbh_amount}:"/>
<h:inputText id="currencyKM" size="10" value="#{cbbhBean.currencyKM}">
<f:validateLongRange minimum="0" maximum="100000"/>
      <f:ajax execute="@this" render="conversionValue" event="blur"/>
</h:inputText>
<h:outputText value="KM"/>
<h:outputText value="#{msgs.cbbh_currency}:"/>
<h:selectOneMenu value="#{cbbhBean.selectedCurrency}">
<f:selectItem itemValue="" itemLabel="#{msgs.cbbh_choose}" />
      <f:selectItems value="#{cbbhBean.selectItems}" />
      <f:ajax execute="@this" render="conversionValue" event="valueChange"/>
   </h:selectOneMenu>
   <h:outputText/>
<h:outputText value="#{msgs.cbbh_conversion}"/>
<h:outputText id="conversionValue" value="#{cbbhBean.conversionValue}">
<f:convertNumber maxFractionDigits="2"/>
</h:outputText>
<h:outputText/>
<h:outputText/>
<h:outputLink value="http://www.cbbh.rs" target="_blank" title="CBBH"><h:outputText value="#{msgs.yahoo_source}: CBBH"/></h:outputLink>
   </h:panelGrid>
   </h:form>
   </rich:panel>
   </h:panelGrid>
</composite:implementation>
```

f:viewParam, GET i bookmarking

- f:viewParam omogućava povezivanje property-ja bean-a sa request parametrima
- iz ovog nastaju nove mogućnosti:
  - tagovi koji koriste GET, a ne POST metodu i šalju parametre kroz URL
  - slanje podataka iz ne-JSF forme ka JSF stranama
  - bookmark stranica
- ovo je nova osobina JSF 2.0

f:viewParam, GET i bookmarking

- prihvatanje request parametara

```
<!DOCTYPE …>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html">
<f:metadata>
<f:viewParam name="param1" value="#{bean.prop1}"/>
<f:viewParam name="param2" value="#{bean.prop2}"/>
</f:metadata>
<h:head>…</h:head>
<h:body>
Blah, blah, #{bean.prop1}
Blah, blah, #{bean.prop2}
Blah, blah, #{bean.derivedProp}
</h:body>
</html>
```

f:viewParam, GET i bookmarking

- slanje request parametara
- komponente
  - h:link
  - h:button

```
<h:link outcome="abc?param1=v1&param2=v2"
value="Click"/>
<h:button outcome="abc?param1=v1&param2=v2"
value="Click"/>
```

  - u ovom primeru odredište je abc.xhtml

f:viewParam, GET i bookmarking

- slanje podataka JSF aplikaciji iz ne-JSF aplikacije
- kreira se “klasična” HTML forma

```
<form action="abc.jsf">
Param 1:
<input type="text" name="param1"/><br/>
Param 2:
<input type="text" name="param2"/><br/>
<input type="submit" value="Go"/>
</form>
```

f:viewParam, GET i bookmarking

- bookmarking
- redirekcija sa viewParam
- za implicitnu navigaciju dodaje se
  “includeViewParams=true” na kraj outcome stringa
- za eksplicitnu navigaciju korišćenjem faces-config.xml datoteke dodaje se
  <redirect include-view-params="true"/> u navigaciona pravila

JSF 2

- JSF 2.1.0
  - http://javaserverfaces.java.net/download.html
- JSF specifikacija
  - http://javaserverfaces-spec-public.java.net/