Programiranje Internet aplikacija
JavaServer Faces (JSF)

Šta je JSF?

- **Glavna odlika ove tehnologije je da se sastoji od sledećih delova:**
  - skup ugrađenih ulazno-izlaznih komponenti
  - događajima vođen (event driven) programski model, sa opcijama za obradu i reagovanje na događaje
  - model komponenti koji omogućava programerima serverske strane razvoj dodatnih komponenti
- **Neke od JSF komponenti su jednostavne, kao na primer ulazna polja ili dugmad. Druge su dosta sofisticiranije, na primer tabele podataka i stabla.**
- **JSF sadrži sav potreban kod za obradu događaja i organizaciju komponenti.**
- **Aplikativni programeri mogu da zanemare sve nepotrebne detalje i da se usredsrede na sam razvoj aplikacione logike.**
- **Takođe, treba napomenuti da je JSF danas deo Java EE standarda, što znači da je uključena u svaki Java EE aplikacioni server, i da se može jednostavno dodati na Web servere, kao što je Jakarta Tomcat.**

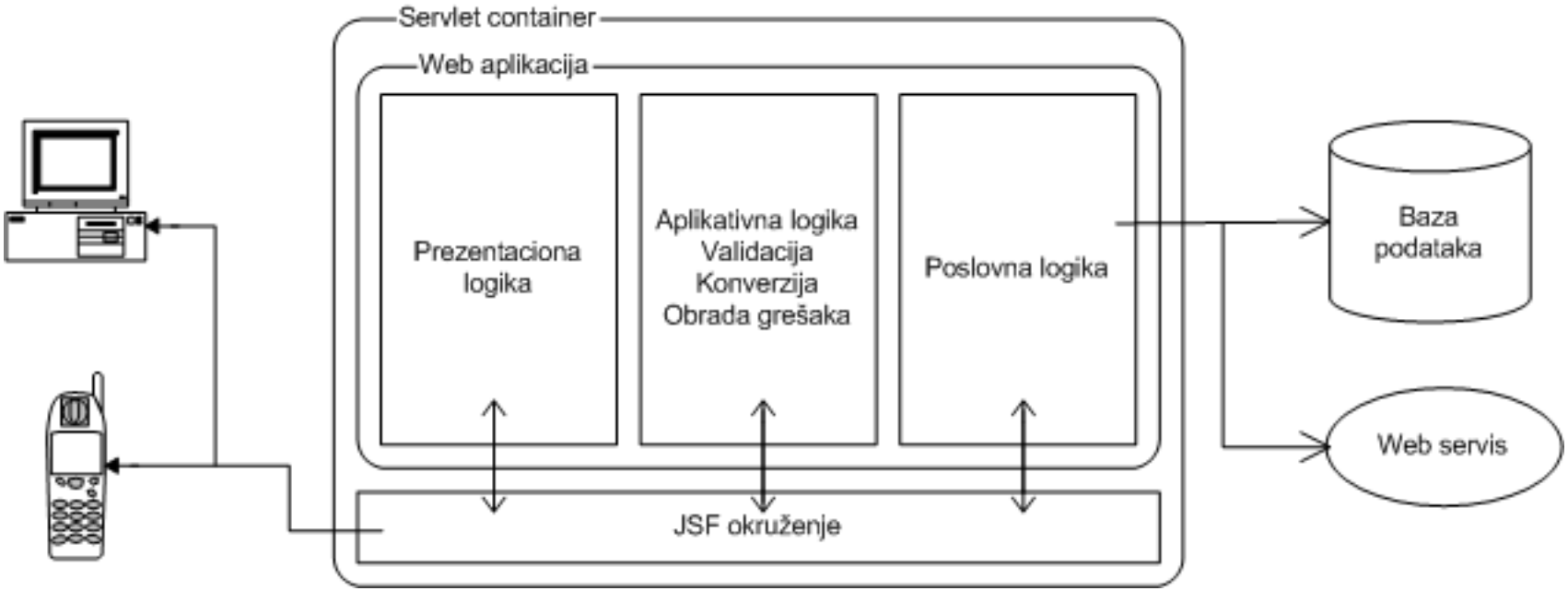
Najvažniji servisi koje JSF obezbeđuje

- MVC arhitektura – sve softverske aplikacije omogućavaju korisnicima da manipulišu sa određenim podacima, ako što su karte za kupovinu, informacije o putovanju, ili bilo koji drugi potrebni podaci za određeni problem. Ovi podaci se nazivaju model (Model). Onaj koji razvija softver proizvodi poglede (Views) na dati model podataka. Kod Internet aplikacija, HTML se koristi da bi se dobili traženi pogledi. JSF omogućava konekciju između pogleda i modela. Na primer, komponenta pogleda se može povezati sa property-ijem Bean-a na sledeći način
- <h:inputText value="#{korisnik.ime}" />
- Ovaj kod dovodi do toga da kada korisnik klikne na dato dugme i forma se pošalje na server, JSF implementacija poziva metod check() koji se nalazi u okviru Bean-a korisniik. Ovaj metod može da izvrši validacione akcije i da izvrši promene modela, kao i da vrati navigacioni ID koji definiše koja će se stranica sledeća prikazati.

Najvažniji servisi koje JSF obezbeđuje

- Konverzija podataka – korisnik unosi podatke u okviru forme u obliku teksta. Objektima povezanim sa poslovnom logikom potrebni su podaci u obliku brojeva, datuma, ili nekog drugog tipa podataka. JSF omogućava da se na jednostavan način specificiraju i primene potrebna pravila konverzije.
- Validacija i obrada grešaka – JSF omogućava jednostavno povezivanje validacionih pravila za sva ulazna polja. Na primer „vrednost u ovo polje je neophodno uneti", „vrednost ovog polja mora biti broj",... Takođe. Kada korisnik unese neodgovarajući podatak, potrebno je prikazati odgovarajuću poruku. JSF umnogome olakšava ovu potrebu.
- Internacionalizacija – JSF sadrži razne opcije za kodiranje posebnih karaktera i selekciju pomoćnih resursa koje se koriste.
- Dodatne komponente – onaj deo tima koji razvija potrebne komponente može veoma kompleksne i složene komponente da pruži na raspolaganje dizajnerima, koji mogu da ih na jednostavan način prikažu na stranici. Na primer može da se razvije komponenta kalendar sa uobičajenom zvučnom podrškom. Tada se može kodirati na stranici sa
- <acme:kalendar value="#{let.dolazak}" poecetak="Pon" />
- Alternativni prikazi – definisano ponašanje JSF je da generiše tagove za HTML stranice. Ali, veoma je jednostavno nadograditi JSF framework da proizvodi oznake za neki drugi opisni jezik, kao što je WML, XUL.

MVC pristup i JSF



Prednosti JSF (u odnosu na standardni MVC)

- **Ugrađene GUI kontrole**

  – JSF sadrži skup API i povezanih ugrađenih tagova koji omogućavaju kreiranje HTML formi sa kompleksnijim interfejsom
- **Obrada događaja**

  – JSF omogućava jednostavno pisanje Java koda koji se poziva kada se forma submituje. Kod može da odgovara određenom dugmetu, promeni određene vrednosti, selekciji korisnika, ...
- **Organizovanje bean-ova**

  – U okviru JSP, može se koristiti property="*" sa jsp:setProperty da bi se automatski popunio bean baziran na request parametrima. JSF vrši nadogradnju ove mogućnosti pomoću nekoliko dodataka, i omogućava jednostavno procesiranje request parametara.
- **Expression Language**

  – JSF sadrži koncizan i moćan jezik za pristup bean propertie-ijima i elementima kolekcija

Prednosti JSF (u odnosu na standardni MVC)

- **Konverzija i validacija polja forme**

  – JSF ima ugrađene opcije za provere da li su vrednosti forme u propisanom formatu i za konvertovanje iz stringa u željeni tip podataka. Ako vrednsot nedostaje ili nije u odgovarajućem formatu, forma se automatski ponovo prikaže sa porukom o grešci i prethodno unetim vrednostima
- **Centralizovana konfiguracija bazirana na fajlovima**

  – Umesto da budu upisane u okviru Java programa, mnoge JSF vrednosti se čuvaju u okviru XML ili property fajlova. Ovakav pristup omogućava izvršavanje promena, bez modifikovanja ili ponovnog kompajliranja Java koda, samo promenama određenog fajla. Takođe, prednost ovakvog pristupa je mogućnost da se Java i Web programeri usredsrede na specifične zadatke, bez razmišljanja o sistemskom layout-u.
- **Konzistentan pristup**

  – JSF ohrabruje konzistentnu upotrebu MVC pristupa u okviru aplikacije.

Nedostaci JSF

- **Potrebno više vremena za učenje**

  – Da bi se realizovao MVC pristup sa standardnom naredbom RequestDispatcher, potrebno je znati samo standardni JSP i servlet API. Da bi se realizovao MVC pristup pomoću JSF frameworka, potrebno je znati standardni JSP i servlet API i veliki i složeni framework koji je otprilike jednak po veličini sa osnovnim sistemom. Ovaj nedostatak dolazi do izražaja posebno kod malih projekata, kratkih rokova, i manjeg iskustva razvojnog tima. Moguće je da se provede isto vremena učeći JSF koliko i za realizaciju samog sistema.
- **Nedovoljna dokumentacija**

  – U poređenju sa standardnim servlet i JSP API, JSF ima samo nekoliko online resursa, a mnogi početnici će smatrati da je online Apache dokumentacija konfuzna i loše organizovana. Takođe, mnogo manje knjiga postoji o ovoj tehnologiji, nego o servletima i JSP.

Nedostaci JSF

- **Manje transparentno**

  – Sa JSF aplikacijama, dosta više stvari se dešava u pozadini, nego kod uobičajenih Java-based Web aplikacija. Kao krajni rezultat, JSF aplikacije su:

  • teže za razumevanje

  • teže za testiranje i optimizaciju

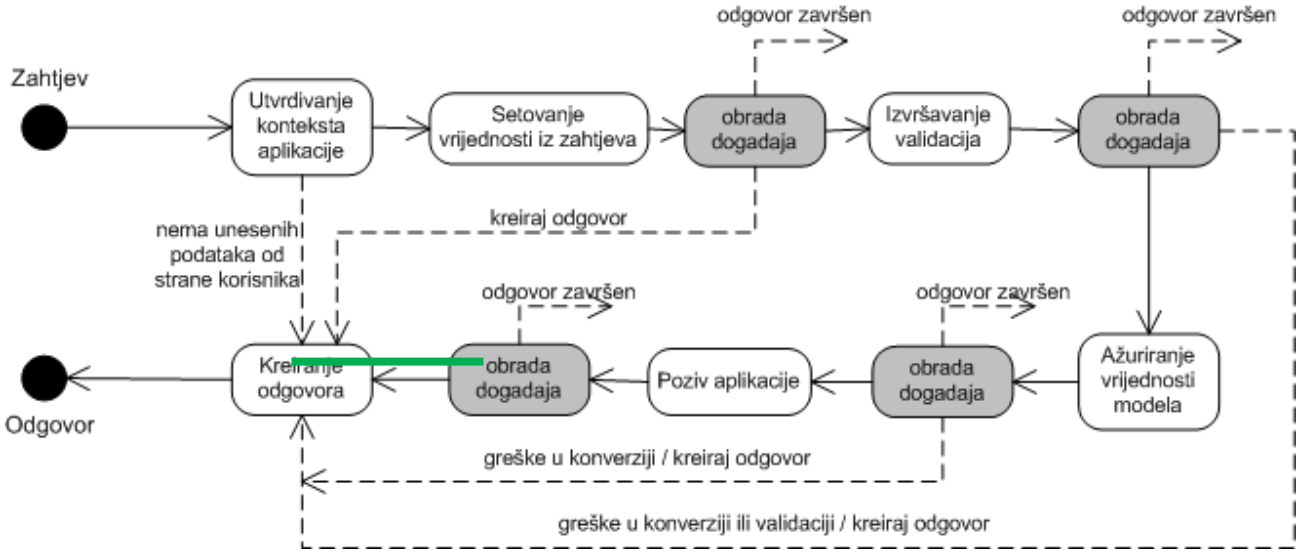
- **Rigidni pristup**

  – Navedeno je da je jedna od prednosti JSF ohrabrivanje za korišćenje MVC pristupa. Iz ovog razloga u okviru JSF je teško koristiti druge pristupe.

JSF 2

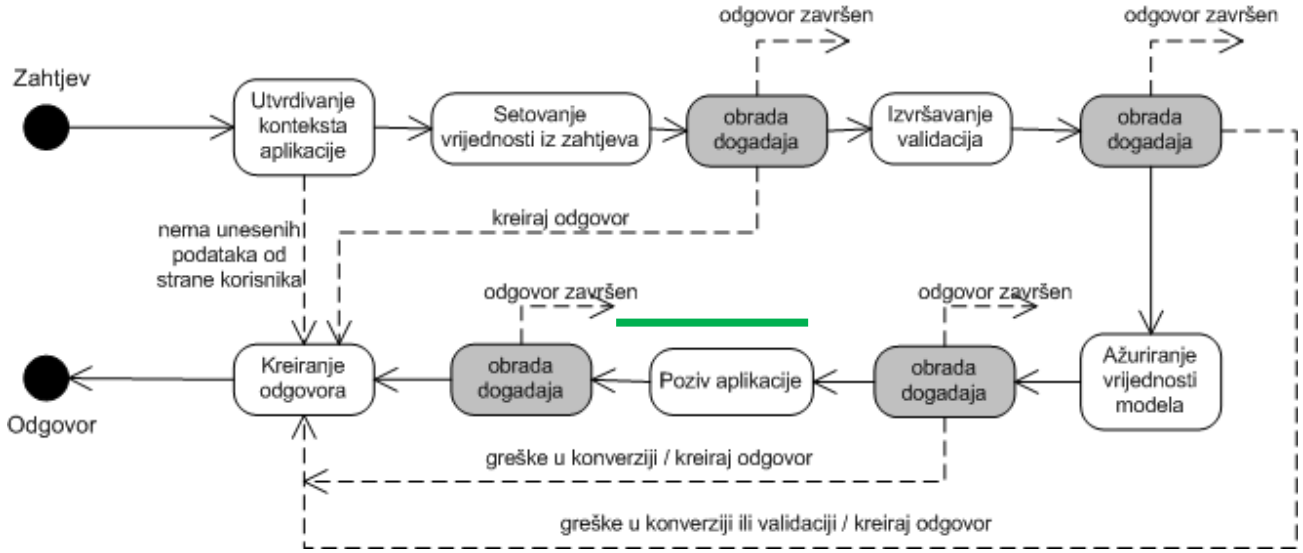
- polazna tačka svih funkcionalnosti JSF aplikacija (bez obzira na verziju specifikacije) leži u njenom ciklusu (*application life cycle*)
- JSF specikacija definiše šest odvojenih faza koje čine ciklus svake aplikacije
- ciklus aplikacije predstavlja niz koraka kroz koje se prolazi od prijema, validacije i obrade korisničkih podataka, pa do kreiranja odgovora i njegovog slanja prema klijentu
- ciklus aplikacije je zadržan od inicijalne verzije okruženja
- unesene izmene odnose se na olakšanu upotrebu okruženja i pojednostavljenje procesa razvoja JSF-baziranih aplikacija

JSF 2



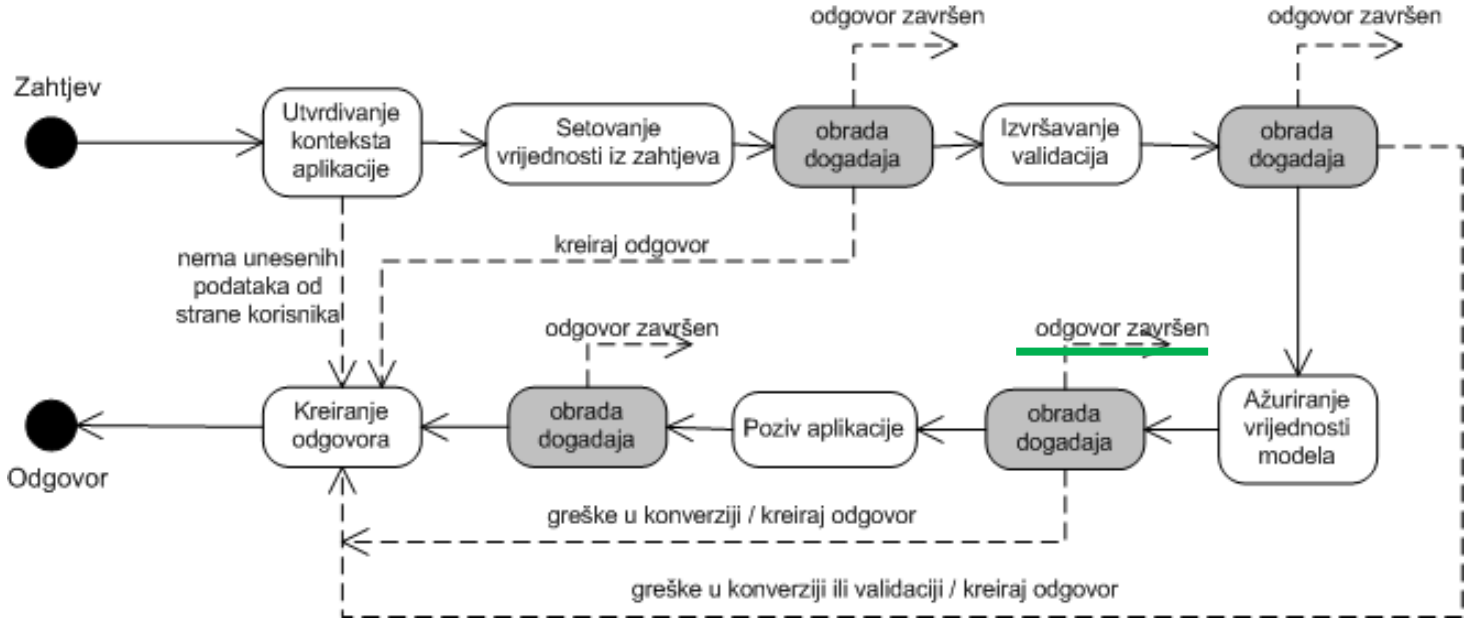
- **JSF framework kontroler koristi view ID (uzet iz zahteva) kako bi pretražio komponente – ako view ne postoji JSF kontroler će kreirati novi, a ako postoji, koristiće ga – view sadrži sve GUI komponente, tj. stablo komponenti**
- **potrebno je razlikovati 3 view instance: new view, initial view i postback**
- **new view – JSF gradi view Faces strane i povezuje event handler-e i validatore sa komponentama – view se čuva u FacesContext objektu**
- **FacesContext objekat sadrži sve informacije koje JSF treba da bi upravljao stanjem GUI komponenti, u tekućem zahtjevu u tekućoj sesiji**
- **initial view – kada se stranica prvi put učitava – JSF kreira prazan view – sa inicijalnog view-a, JSF direktno prelazi na fazu kreiranja odgovora**
- **postback – korisnik se vraća na stranicu koju je ranije posetio – view koji odgovara datoj strani već postoji – radi se restore – JSF radi rekonstrukciju – sledeća faza nakon postback-a je faza "setovanje vrednosti iz zahteva"**

JSF 2



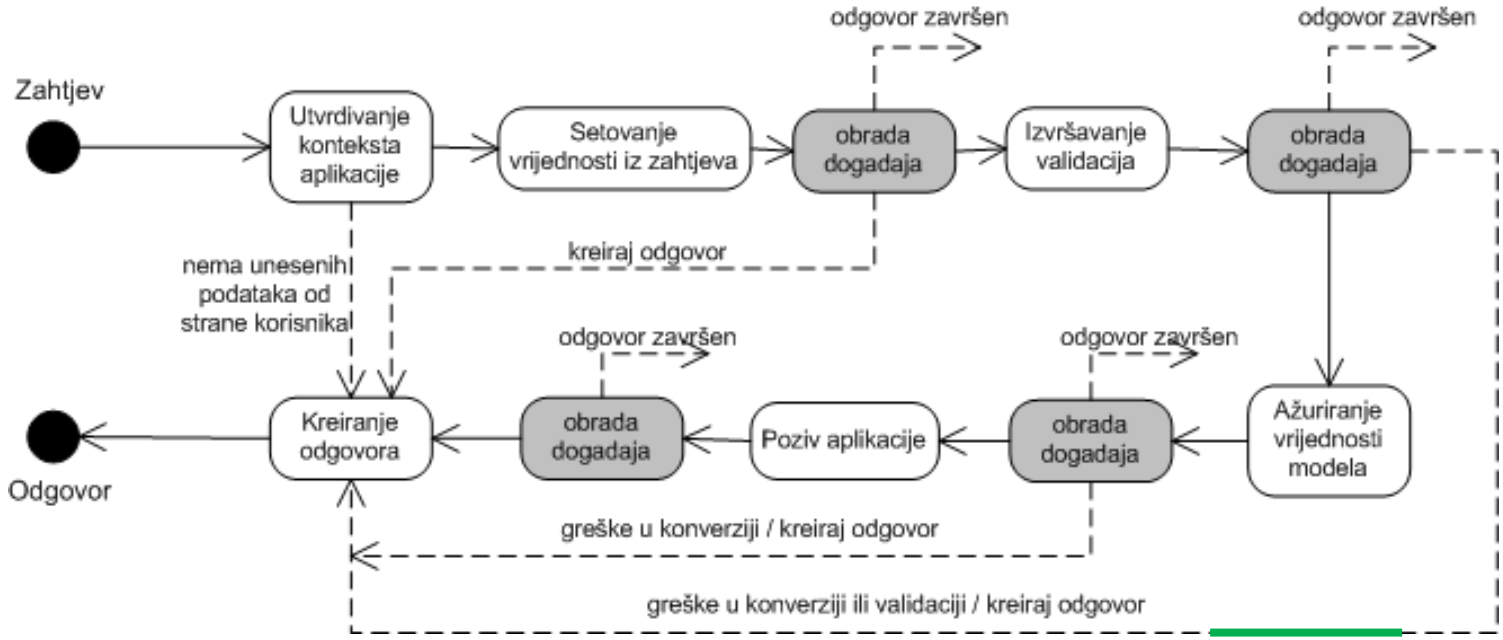
- **u fazi setovanja vrednosti iz zahteva prolazi se kroz sve komponente u okviru stabla komponenti – za svaki objekat komponenti se proverava koja mu vrednost pripada i ona mu se dodeljuje**
- **vrednosti komponenti se uzimaju iz request parametara, ili iz cookie-a ili header-a**
- **ako immediate event handling property komponente nije postavljan na true – vrednosti se samo konvertuju (npr., ako je vrednost polja Integer, vrednost se konvertuje u Integer) – ako konverzija nije uspjela, generiše se poruka o grešci koja će biti prikazana za vreme faze "kreiranje odgovora", zajedno sa svim validacionim greškama (smešta se u FacesContext) – koristi se kada nije potrebno izvršiti validaciju kompletne forme**
- **ako je immediate event handling property komponente postavljan na true, vrednosti se konvertuju u odgovarajući tip i vrši se validacija – konvertovana vrednost se smešta u komponentu – ako konverzija ili validacija nije uspešna, generiše se poruka o grešci...**

JSF 2



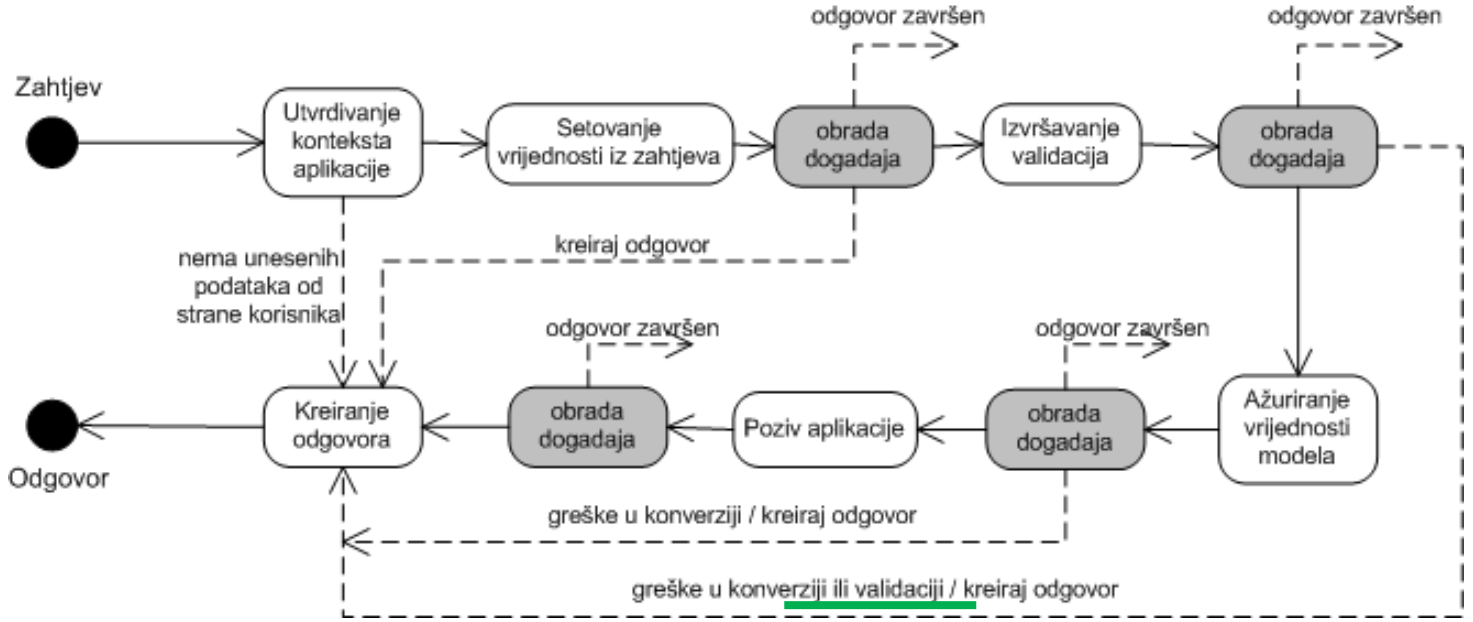
- **u fazi provere validnosti unesenih podataka okruženje pruža mogućnost prekida dalje obrade i generisanja odgovora koji informiše korisnika o neispravnom unosu**
- **poslati stringovi se prvo prebacuju u „lokalne" vrednosti, koje mogu biti objekti bilo kog tipa**
- **u ovoj fazi biće izvršena validacija vrednosti svih komponenti, prema definisanim validacionim pravilima koja mogu biti predefinisana (ugrađena JSF validaciona pravila) ili definisana od strane programera**
- **unesene vrednosti se proveravaju prema validacionim pravilima – ako unesena vrednost nije validna poruka o grešci se dodaje u FacesContext, a komponenta se označava kao nevalidna**
- **ako postoji nevalidna komponenta JSF prelazi u fazu "kreiranje odgovora" gde će biti prikazan trenutni view sa porukama o validacionim greškama**
- **ako ne postoje validacione greške, JSF prelazi u fazu "ažuriranje vrednosti modela"**

JSF 2



- **u ovoj fazi se dobijene vrednosti iz prezentacionog sloja prenose prema modelu podataka, kako bi se u sledećoj fazi mogla izvršiti obrada uz pomoć procedura koje čine deo poslovne logike aplikacije**
- **ažuriraju se vrednosti server-side modela, tako što ažuriraju property-je managed bean-ova**
- **samo property-ji bean-ova koji su vezani za vrednost komponente će biti promenjeni (ažurirani)**
- **kako se ova faza izvršava nakon validacije, može se sa sigurnošću znati da su vrednosti validne, bar na nivou polja forme**

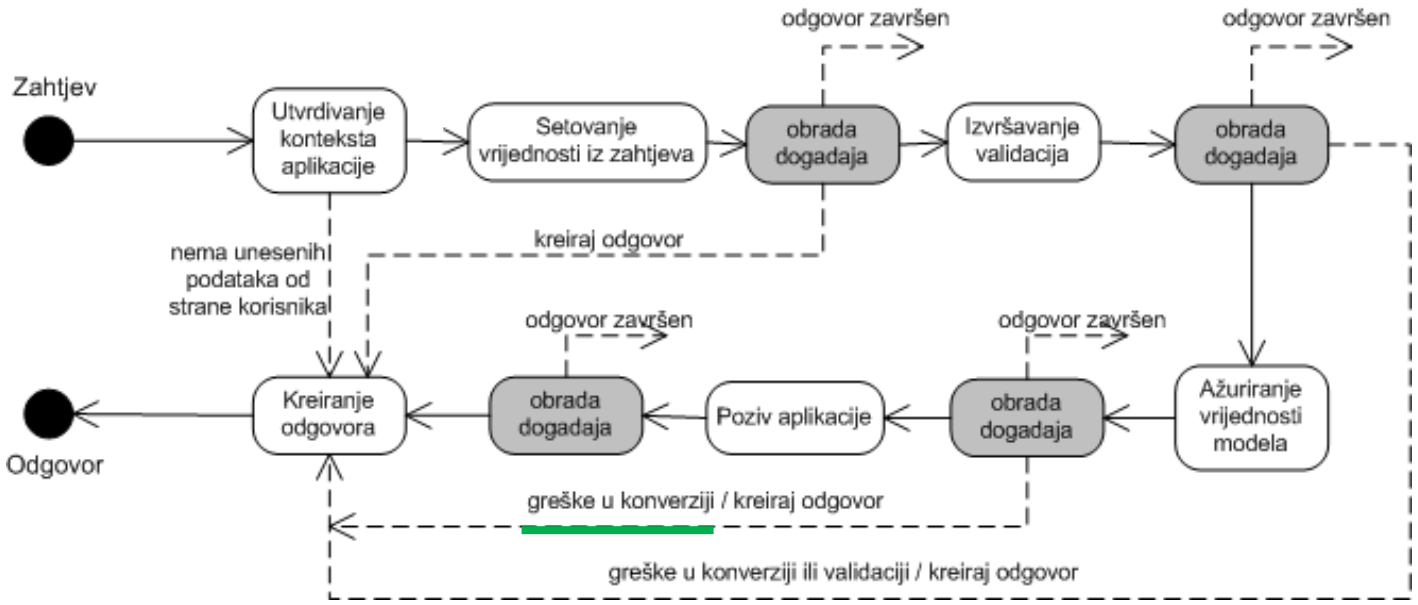
JSF 2



- **pre generisanja odgovora koji će biti prezentovan korisniku, vrši se konačna obrada podataka prosleđenih modelu, kao i odluka o navigacionom pravilu koje će biti upotrebljeno u daljem toku aplikacije**

- **JSF kontroler poziva metodu koja obrađuje podatke unesene sa forme, koji su konvertovani, nad kojim je izvršena validacija i koji su smešteni u managed bean-ove, tako da se sada mogu koristiti za izvršavanje business logike**
- **u ovoj fazi se generiše izlazni string, koji se šalje dijelovima odgovornim za navigaciju, gde se izvršava poziv odgovarajuće stranice**

JSF 2



- **u ovoj fazi kreira se odgovor i šalje klijentu.**

- **prikazuje se odgovarajući view, sa svim svojim komponentama u trenutnom stanju**
- **kada korisnik sa nove stranice pošalje novu formu, klikne na link ili na drugi način generiše novi zahtev, startuje se novi ciklus**

Osnovne osobine – JSF stranice

- Iako JSF prirodno koristi JSP kao prezentacionu tehnologiju - nije ograničen samo na JSP. Kako se JSF API sloj nalazi direktno iznad Servlet API sloja to omogućava de se neka druga prezentaciona tehnologija koristi umesto JSP-a .
- Da bi mogli da se koriste JSF tagovi svaka JSP strana na vrhu mora imati sledeće dve taglib direktive:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
```

- Prva uključuje *html_basic* a druga *jsf_core* tagove.
- *html_basic* tag-ovi predstavljaju HTML form kontrole i druge osnovne HTML elemente i prikazuju podatke ili prihvataju podatke od korisnika (column, commandButton, commandLink, form, message, outputText, itd.)
- *jsf_core* tagovi se koriste za ključne akcije koje su nezavisne od određenog render-a (na primer tagovi za rad sa događajima (actionListener), tagovi za konverziju podataka (converter, convertDateTime, convertNumber), tagovi za validaciju (validator, validateLength), loadBundle, param, subview, view itd.).

Osnovne osobine – navigacija

- Navigacija je jedna od najbitnijih osobina JSF paketa. Navigacija za neku aplikaciju se definiše u okviru faces-config.xml konfiguracionog fajla:

```
<navigation-rule>
  <from-view-id>/pages/inputname.jsp</from-view-id>
  <navigation-case>
    <from-outcome>greeting</from-outcome>
    <to-view-id>/pages/greeting.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

Osnovne osobine – navigacija

- Definisano je kako će se sa strane inputname.jsp (definisane u okviru from-view-id elementa) preći na stranu greeting.jsp (definisane u okviru to-view-id elementa).
- navigation-rule može imati proizvoljan broj navigation-case-ova od kojih svaki definiše koja se strana sledeća učitava bazirano na logičkom ishodu (definisanom u okviru from-outcome).
- Ishod može biti definisan od strane action atributa UICommand komponente koja vrši submit forme:

```
<h:commandButton action="greeting"
    value="#{msg.button_text}" />
```

- Ishod se takođe može dobiti kao povratna vrednost akcione metode pratećeg bean-a. Ovaj metod vrši neki proces da bi utvrdio ishod. Npr. metod može da proveri da li je lozinka koju je korisnik uneo ispravna. Ako jeste metod vraća success a u suprotnom slučaju failure.
- U prvom slučaju korisnik bi bio prebačen na stranicu u okviru sajta a u drugom da se ponovo učita login strana sa porukom o grešci.

Osnovne osobine – navigacija

- Ako se pažljivo biraju stringovi, moguće je skupiti višestruka pravila navigacije na jedno mesto. Ako želimo da se akcija *prikaz* nalazi u više različitih JSF stranica i da iz svake poziva stranicu *prikaz.jsp*, tada unosimo sledeće pravilo navigacije:

<navigation-rule>

<navigation-case>

<from-outcome>logout</from-outcome>

<to-view-id>/logout.jsp</to-view-id>

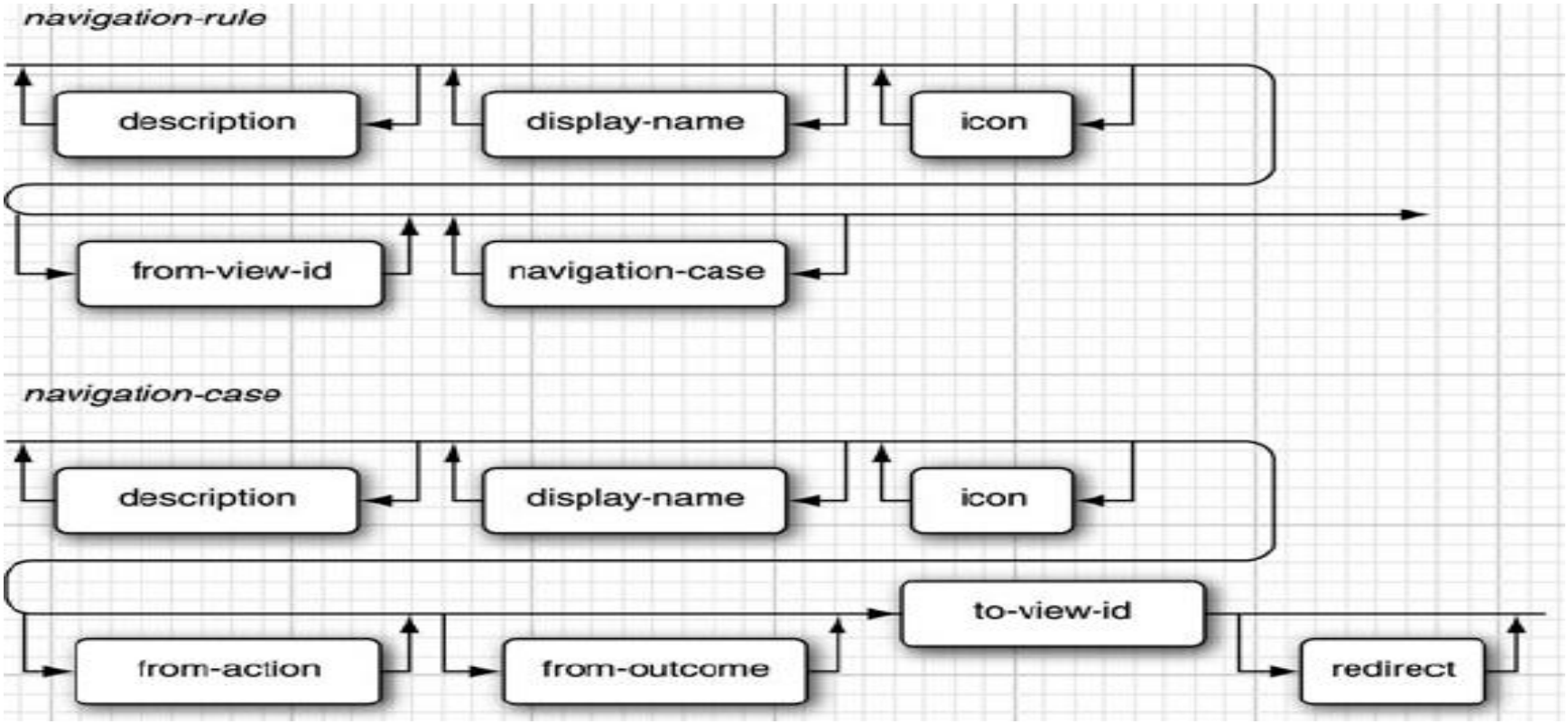
</navigation-case>

</navigation-rule>

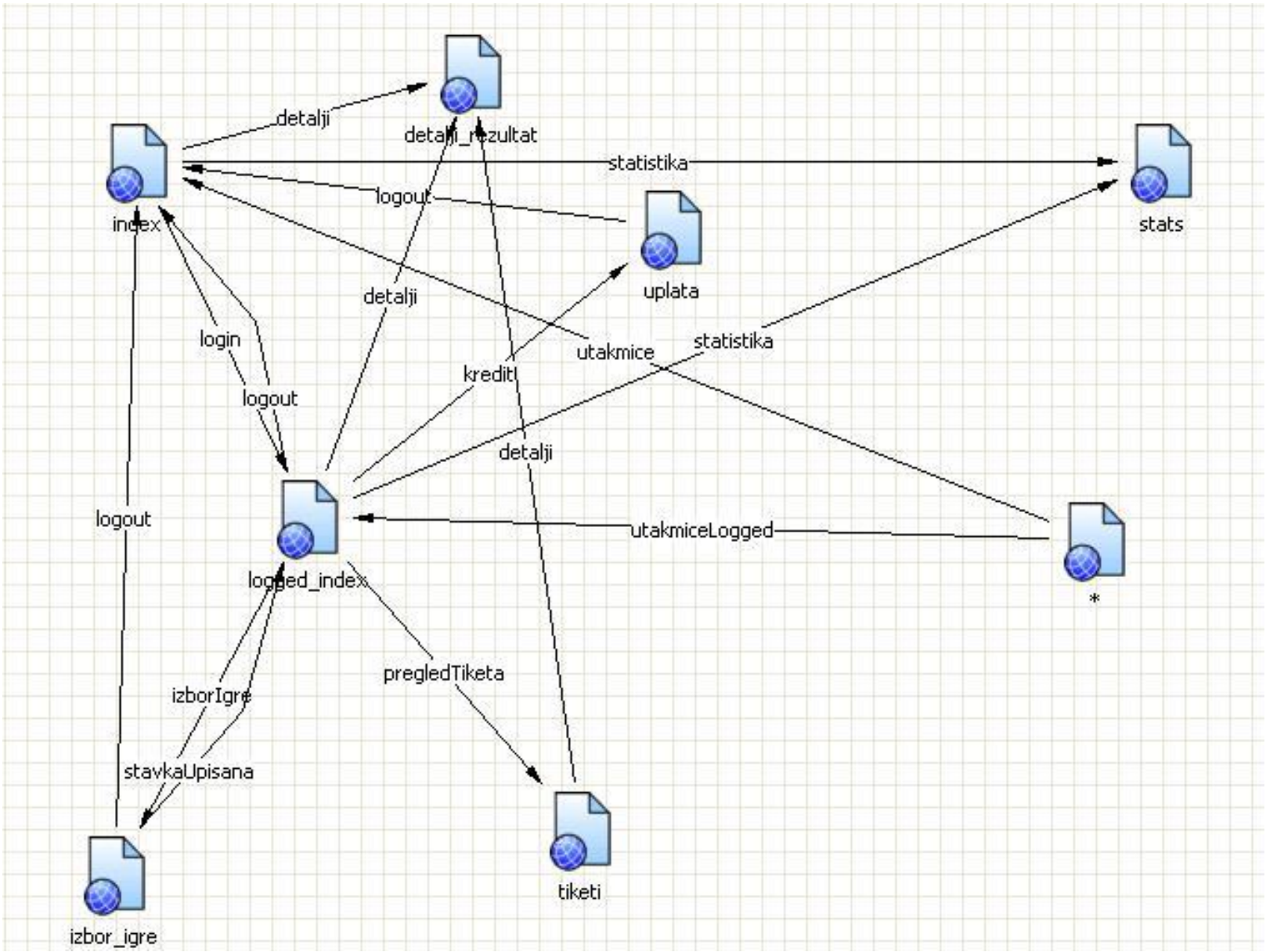
- Ovo pravilo se odnosi na sve stranice zato što nijedan from-view-id element nije definisan**.**

Osnovne osobine – navigacija

- Tehnike koje su prikazane su dovoljne za većinu aplikacija. Ali mogu se primeniti i druga pravila za navigacione elemente u okviru *faces-config.xml* fajla.



Osnovne osobine – navigacija



Osnovne osobine – validacija

- JSF tehnologija podržava mehanizam za validaciju lokalnih podataka koji pripadaju komponentama koje mogu dobijati vrednost (kao npr. text field).
- Validacija se odigrava pre nego što se u odgovarajući model podataka postavi na lokalnu vrednost.
- Definisan je standardan skup klasa koje obavljaju uobičajene validacione provere.
- Osnovna JSF tag biblioteka definiše skup tag-ova koji odgovaraju standardnim implementacijama *Validator* interfejsa.

Osnovne osobine – validacija

Klasa za validaciju	Tag	Funkcija
DoubleRangeValidator	validateDoubleRange	Proverava da li je lokalna vrednost komponente u određenom opsegu. Vrednost mora biti u pokretnom zarezu ili da može da se konvertuje u pokretni zarez.
LengthValidator	validateLength	Proverava da li je dužina lokalne vrednosti komponente u određenom opsegu.Vrednost mora biti String
LongRangeValidator	validateLongRange	Proverava da li je lokalna vrednost komponente u određenom opsegu. Vrednost može biti bilo koji numerički tip ili String koji se može konvertovati u long.

Osnovne osobine - validacija

- **Kada se koriste standardne implementacije *Validator-a* ne treba pisati kod koji će da izvrši validaciju. Potrebno je samo unutar tag-a koji predstavlja komponentu tipa *UIInput* ugnjezditi standardni validacioni tag i obezbediti određene granice ako ih tag zahteva:**

```
<h:inputText value="#{personBean.personName}"
   required="true">
   <f:validateLength minimum="2" maximum="10"/>
</h:inputText>
```

- **Moguće je takođe napraviti sopstveni validator i odgovarajući tag. Postoje dva načina da se to uradi:**
  - Implementirati *Validator* interfejs koji će da izvede validaciju.
  - Implementirati pozadinsku metodu bean-a koja izvodi validaciju.

Osnovne osobine – bean-ovi

- Tipična JSF aplikacija uparuje bean u pozadini sa svakom stranicom u aplikaciji.
- Bean definiše sadržaj i metode koje su pridružene svakoj UI komponenti koja se koristi na stranici.
- Sadržaj samog bean-a može biti povezan ili sa instancom komponente ili sa njenom vrednošću.
- Metode backing bean-a izvode određene funkcije vezane za komponentu kao što su: validacija podataka komponente, obrada događaja koje komponenta okida i izvođenje akcija vezanih za navigaciju kada se aktivira komponenta.
- Atribut *value* tag-a komponente se koristi da poveže sadržaj bean-a sa vrednošću komponente.
- Atribut *binding* tag-a komponente se koristi da poveže sadržaj bean-a sa instancom komponente.

Osnovne osobine – bean-ovi

```
public class PersonBean {
  String personName;
  /**
  * @return Vraca ime
  */
  public String getPersonName() {
    return personName;
  }
  /**
  * @param Ime
  */
  public void setPersonName(String name) {
    personName = name;
  }
}
```

- Vrednost komponente je povezana sa atributom bean-a na JSP stranici:

```
<h:inputText value="#{personBean.personName}" required="true">
  <f:validateLength minimum="2" maximum="15"/>
</h:inputText>
```

Osnovne osobine – bean-ovi

- Posle implementacije bean-ova koji će se koristiti u aplikaciji potrebno je konfigurisati ih u faces-config.xml konfiguracionom fajlu.
- To je potrebno da bi JSF mogao da automatski kreira nove instance bean-ova kad god se ukaže potreba:

```
<managed-bean>
  <managed-bean-name>personBean</managed-bean-name>
  <managed-bean-class>jsfks.PersonBean</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

JSF i bean-ovi

- Mnoge JSF komponente korisničkog interfejsa sadrže atribut *value* kojim je moguće direktno specifikovati vrednost, ili vezati komponentu za vrednost koja se dobija iz polja nekog bean-a. Zavisno od toga da li je u pitanju ulazna ili izlazna komponenta korisničkog interfejsa, na isti način je moguće i čitanje i upis vrijednosti referenciranog polja.

- <h:outputText value="#{userBean.ime}"/>
- <h:inputText value="#{userBean.ime}"/>

- *Getter*, u prvom slučaju, se poziva pri iscrtavanju komponente, dok se *setter*, u drugom primeru, poziva pri obradi odgovora

Realizacija navigacije

- **JSF kontrola toka**
- **Osnovni koraci kod upotrebe JSF**
- **Statička navigacija**
  – Mapiranje u jedan rezultat
- **Dinamička navigacija**
  – Mapiranje u više rezultata

JSF kontrola toka

- **Forma se prikaže**
  – Forma koristi f:view i h:form
- **Forma se pošalje**
  – Originalni URL i ACTION URL su http://…/*blah*.faces
- **Bean se instancira**
  – Navodi se u managed-beans delu faces-config.xml
  – setter metodi navedeni u h:inputText (...) se izvršavaju
  • dobijene vrednosti su vrednosti iz tekst polja u trenutku slanja
- **Poziva se metod action controller-a**
  – Navodi se u atrubutu action od h:commandButton
- **action metod vraća uslov**
  – String koji se poklapa sa vrednostima from-outcome iz navigacionih pravila faces-config.xml
- **Rezultujuća stranica se prikazuje**
  – Stranica koristi h:outputText da bi prikazala propertie-ije bean-a

Koraci kod upotrebe JSF

1. **Kreirati bean sa propertie-ijim za svako polje forme**
   - Par getter/setter metoda za svako polje forme
2. **Kreirati početnu formu**
   - Koristiti f:view, h:form, h:*blah*, i h:commandButton
3. **Specificirati action controller metod**
   - Koristiti action atribut h:commandButton
4. **Kreirati action controller metod**
   - U okviru bean-a iz tačke #1. Pregledati podatke sa forme, primeniti poslovnu logiku, smestiti rezultate u bean-ove, i vratiti uslove
5. **Promeniti faces-config.xml**
   - Deklarisati form bean i pravila navigacije
6. **Kreirati JSP stranice**
   - Za svaki uslov koji se vraća kao rezultat
7. **Zaštiti osnovne JSP stranice od pristupa**
   - Koristiti filter ili podešavanja bezbednosti

Primer 1: Upotreba bean-ova

- **Isto kao u prethodnom primeru**
  - Originalni URL:
  - http://*hostname*/jsf-test/register3.faces
  - Kada se forma pošalje 3 moguća rezultata
  - Poruka o nelegalnoj email adresi
  - Poruka o nelegalnoj šifri
  - Uspešno
- **Razlike**
  - Action controller prihvata podatke zahteva pomoćui bean-a
  - Izlazne stranice pristupaju propertie-ijima bean-a
- **Glavni zadaci**
  - Definisati bean sa propertie-ijima odgovarajućim podacima forme
  - Deklarisati bean-ove u faces-config.xml
  - Prikazati propertie-ije bean-a

Glavni zadaci

- **Napraviti property-ije bean-a za svaki parametar zahteva**

public class MyBean {
public String get*Blah*() {...}
public void set*Blah*(String newValue) {...} ...
}

- **Koristiti faces-config.xml za deklaraciju bean-a**

<faces-config>
<managed-bean>
<managed-bean-name>*beanName*</managed-bean-name>
<managed-bean-class>*package*.MyBean</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>...
</faces-config>

- **Koristiti h:inputText za povezivanje tekst polja sa property-ijem**

<h:inputText value="#{*beanName*.*propertyName*}"/>

- **Koristiti h:outputText da bi se prikazali property-iji bean-a**

<h:outputText value="#{*beanName*.*propertyName*}"/>

Korak 1: Kreiranje bean-a

- **Po jedan property za svaki parametar zahteva**
  – Ako ulazna forma ima value="#{name.foo}", tada bean treba da ima metode getFoo i setFoo.
- **Dodatne property-ije za svaku izlaznu vrednost**
- **Popunjavanje bean-a**
  – Sistem će popuniti vrednosti property-ija automatski
  – Stringovi se konvertuju sa jsp:setProperty
  • Forma se ponovo prikazuje, ako ima grešaka, validacija
- **Action controller**
  – Metod može direktno pristupiti property-ijima bean-a
  • ExternalContext je potreban u slučaju kada se pristupa request headers, cookies, ...
  – Metod sadrži i poslovne objekte i čuva ih u okviru property-ija rezervisanih za izlazne vrednosti

Korak 1: Kreiranje bean-a

- **Prezentacija parametara zahteva**

```
package coreservlets;
public class RegistrationBean {
private String email = "user@host";
private String password = "";
public String getEmail() {
return(email);
}
public void setEmail(String email) {
this.email = email;
}
public String getPassword() {
return(password);
}
public void setPassword(String password) {
this.password = password; }
```

Korak 1: Kreiranje bean-a

- **Kod za smeštanje rezultata (RegistrationBean, ...)**

```
...
private SuggestionBean suggestion;
public SuggestionBean getSuggestion() {
return(suggestion);
}
```

Korak 1: Kreiranje bean-a

- **Kod za prikaz rezultata**

```
package coreservlets;
public class SuggestionBean {
private String email;
private String password;
public SuggestionBean(String email, String password) {
this.email = email;
this.password = password;
}
public String getEmail() {
return(email);
}
public String getPassword() {
return(password);
}
}
```

Korak 1: Kreiranje bean-a

- **Kod za prikaz rezultata**

```
package coreservlets;
public class SuggestionUtils {
private static String[] suggestedAddresses =
{ "president@whitehouse.gov",
"gates@microsoft.com",
"palmisano@ibm.com",
"ellison@oracle.com" };
private static String chars =
"abcdefghijklmnopqrstuvwxyz0123456789#@$%^&*?!";
public static SuggestionBean getSuggestionBean() {
String address = randomString(suggestedAddresses);
String password = randomString(chars, 8);
return(new SuggestionBean(address, password));
}
...
```

Korak 2: Kreiranje početne forme

- **Isto kao u prethodnom primeru, osim**
  – Tagovi h:input*Blah* tpomoću vrednosti atributa definišu odgovarajući bean property
  • Takođe definišu početnu vrednost tekst polja
  – Nije potrebno definisati id atribut
- **Primer koda**

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>…
<h:form>
Email address:
<h:inputText value="#{registrationBean.email}"/><BR>
Password:
<h:inputSecret value="#{registrationBean.password}"/><BR>
<h:commandButton value="Sign Me Up!"
action="#{registrationBean.register}"/>
</h:form>…
</f:view>
```

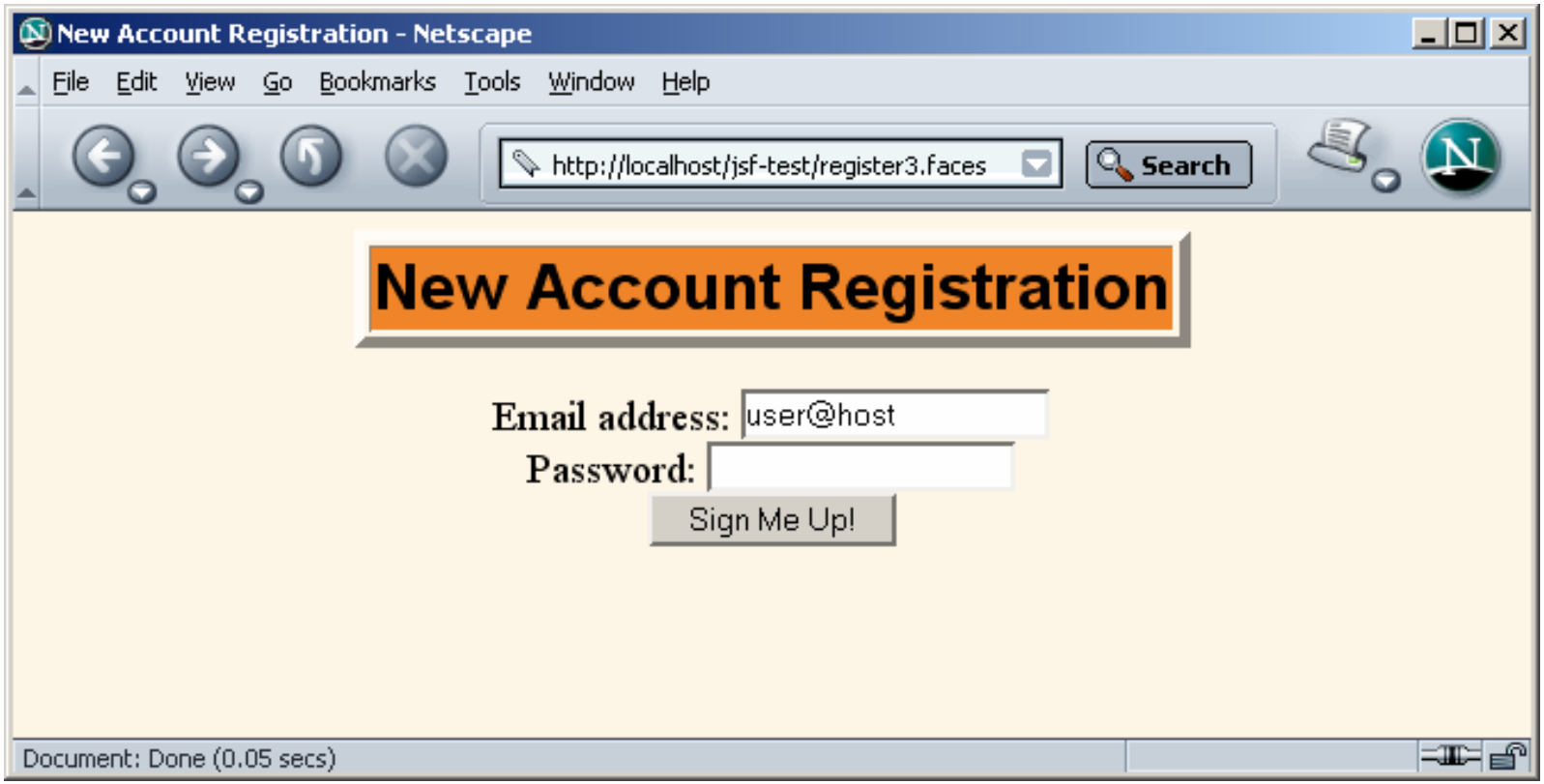
Korak 3: definisati akciju

- **Isto kao u prethodnom primeru**

```
<h:form>
Email address:
<h:inputText value="#{registrationBean.email}"/><BR>
Password:
<h:inputSecret
   value="#{registrationBean.password}"/><BR>
<h:commandButton value="Sign Me Up!"
action="#{registrationBean.register}"/>
</h:form>
```

Korak 3: definisati akciju

- File je *tomcat_dir*/webapps/jsf-test/register3.jsp
- URL je http://localhost/jsf-test/register3.faces
- Vrednost user@host je definisana u okviru bean-a



Korak 4: kreirati akciju

- **Moguće je pristupiti property-ijima bean-a direktno**
  – Sistem ih automatski popunjava *pre* poziva action metoda
  – Nije potrebno pristupati objektu ExternalContext ili pozivati request.getParameter
- **Rezultati se smeštaju u property-ije**
  – Koriste se results-beans kao i kod traditionalnog MVC pristupa
- **I dalje se rezultujući stringovi poklapaju sa mogućim izalzima**
  – Kao u prethodnom primeru
- **Objekat ExternalContext se ponekad koristi**
  – Za pristup cookies, podacima sesije, request headers, ...

Korak 4: kreirati akciju

```
public class RegistrationBean {
// Property-iji su prikazani na prethodnim slajdovima
public String register() {
if ((email == null) ||
(email.trim().length() < 3) ||
(email.indexOf("@") == -1)) {
suggestion = SuggestionUtils.getSuggestionBean();
return("bad-address");
} else if ((password == null) ||
(password.trim().length() < 6)) {
suggestion = SuggestionUtils.getSuggestionBean();
return("bad-password");
} else {
return("success");
}
}
}
```

Korak 5: Promeniti faces-config.xml

- **Deklarisanje bean-a**

```
...
<faces-config>
<managed-bean>
<managed-bean-name>
registrationBean
</managed-bean-name>
<managed-bean-class>
coreservlets.RegistrationBean
</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
...
</faces-config>
```

Korak 5: Promeniti faces-config.xml

- **Specificirati pravila navigacije**

```
...
<faces-config>
...
<navigation-rule>
<from-view-id>/register3.jsp</from-view-id>
<navigation-case>
<from-outcome>bad-address</from-outcome>
<to-view-id>/WEB-INF/results/bad-address3.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>bad-password</from-outcome>
<to-view-id>/WEB-INF/results/bad-password3.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>success</from-outcome>
<to-view-id>/WEB-INF/results/result3.jsp</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>
```

Korak 6: Kreiranje izlaznih JSP strana

- Koristiti h:outputText da bi se pristupilo property-ijima bean-a

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<!DOCTYPE ...>
<HTML>
...
<h:outputText value="#{beanName.propertyName}"/>
...
</HTML>
</f:view>
```

Korak 6: Kreiranje izlaznih JSP strana

- .../jsf-test/WEB-INF/results/bad-address3.jsp

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<!DOCTYPE ...>
<HTML>
...
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">Illegal Email Address</TH></TR>
</TABLE>
<P>
The address
"<h:outputText value="#{registrationBean.email}"/>"
is not of the form username@hostname (e.g.,
<h:outputText value="#{ registrationBean.suggestion.email}"/>).
<P>
Please <A HREF="register3.faces">try again</A>.
...
</HTML>
</f:view>
```

Rezultat


Illegal Email Address - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/jsf-test/register3.faces
Search
Illegal Email Address
The address "Bill Gates" is not of the form username@hostname (e.g., ellison@oracle.com).
Please try again.
Document: Done (0.13 secs)

Korak 6: Kreiranje izlaznih JSP strana

- .../jsf-test/WEB-INF/results/bad-password3.jsp

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<!DOCTYPE ...>
<HTML>
...
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">Illegal Password</TH></TR>
</TABLE>
<P>
The password
"<h:outputText value="#{registrationBean.password}"/>"
is too short; it must contain at least six characters.
Here is a possible password:
<h:outputText
value="#{registrationBean.suggestion.password}"/>.
<P>
Please <A HREF="register3.faces">try again</A>.
...
</HTML>
</f:view>
```

Rezultat


Illegal Password - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/jsf-test/register3.faces
Search
Illegal Password
The password "steve" is too short; it must contain at least six characters. Here is a possible password: z3lk&87h.
Please try again.
Document: Done (0.1 secs)

Properties fajlovi – fiksni stringovi

1. Kreirati .properties fajl u okviru WEB-INF/classes

– Sadrži jednostavne parove keyName=value

2. Učitati fajl pomoću f:loadBundle

– basename definiše osnovno ime fajla

– var definiše promenljivu (Map) koja će sadržati rezultat

• Relativno u odnosu na WEB-INF/classes, .properties se podrazumeva

• Za WEB-INF/classes/messages.properties

<f:loadBundle basename="messages" var="msgs"/>

• Za WEB-INF/classes/package1/test.properties

<f:loadBundle basename="package1.test" var="msgs"/>

3. Izlazne poruke koruste uobičajene EL konstrukcije

– #{msgs.keyName}

WEB-INF/classes/ messages1.properties

```
title=Registration
text=Please enter your first name, last name,
and email address.
firstNamePrompt=Enter first name
lastNamePrompt=Enter last name
emailAddressPrompt=Enter email address
buttonLabel=Register Me
```

signup1.jsp (.faces)

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<f:loadBundle basename="messages1" var="msgs"/>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>
<h:outputText value="#{msgs.title}"/>
</TITLE>
<LINK REL="STYLESHEET"
HREF="./css/styles.css"
TYPE="text/css">
</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">
<h:outputText value="#{msgs.title}"/></TH></TR>
</TABLE>
<BR>
<h:outputText value="#{msgs.text}"/>
<P>
```

signup1.jsp (.faces)

```
<h:form>
<h:outputText value="#{msgs.firstNamePrompt}"/>:
<h:inputText value="#{person.firstName}"/>
<BR>
<h:outputText value="#{msgs.lastNamePrompt}"/>:
<h:inputText value="#{person.lastName}"/>
<BR>
<h:outputText value="#{msgs.emailAddressPrompt}"/>:
<h:inputText value="#{person.emailAddress}"/>
<BR>
<h:commandButton
value="#{msgs.buttonLabel}"
action="#{person.doRegistration}"/>
</h:form>
</CENTER></BODY></HTML>
```

Stringovi sa parametrima

1. Kreira se .properties fajl u okviru WEB-INF/classes

– Vrednosti sadrže {0}, {1}, {2}, ...

– N.p., nekoIme=blah {0} blah {1}

2. Učitati fajl sa f:loadBundle kao i ranije

– basename definiše osnovno ime fajla

– var definiše promenljivu koja će sadržati rezultat

3. Izlazne poruke koriste h:outputFormat

– vrednost definiše osnovnu poruku

– ugnežđene f:param daju vrednosti za zamenu

– N.p.:

```
<h:outputFormat value="#{msgs.someName}">
<f:param value="value for 0th entry"/>
<f:param value="value for 1st entry"/>
</h:outputFormat>
```

messages2.properties

```
title=Registration
firstName=first name
lastName=last name
emailAddress=email address
text=Please enter your {0}, {1}, and {2}.
prompt=Enter {0}
buttonLabel=Register Me
```

signup2.jsp (.faces)

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view locale="#{facesContext.externalContext.request.locale}">
<f:loadBundle basename="messages2" var="msgs"/>
...
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">
<h:outputText value="#{msgs.title}"/></TH></TR>
</TABLE>
<BR>
<h:outputFormat value="#{msgs.text}">
<f:param value="#{msgs.firstName}"/>
<f:param value="#{msgs.lastName}"/>
<f:param value="#{msgs.emailAddress}"/>
</h:outputFormat>
<P>
<h:form>
<h:outputFormat value="#{msgs.prompt}">
<f:param value="#{msgs.firstName}"/>
</h:outputFormat>:
<h:inputText value="#{person.firstName}"/>
```

signup2.jsp (.faces)

```
<BR>
<h:outputFormat value="#{msgs.prompt}">
<f:param value="#{msgs.lastName}"/>
</h:outputFormat>:
<h:inputText value="#{person.lastName}"/>
<BR>
<h:outputFormat value="#{msgs.prompt}">
<f:param value="#{msgs.emailAddress}"/>
</h:outputFormat>:
<h:inputText value="#{person.emailAddress}"/>
<BR>
<h:commandButton
value="#{msgs.buttonLabel}"
action="#{person.doRegistration}"/>
</h:form>
</CENTER></BODY></HTML>
</f:view>
```

Obrada događaja

- **Postoje dve varijante događaja koje generiši korisnici**
  - Događaji koji strartuju neki back-end proces
  - Događaji koji utiču samo na format korisničkog interfejsa
- **JSF izvršava podelu koda koji izvršava ove operacije na *action controllers* i *event listeners***
  - Action controllers obrađuju slanje glavne forme
    - Okidaju se posle popunjavanja bean-ova
    - Okidaju se posle validacione logike
    - Vraća stringove koji utiču na dalju navigaciju
  - Event listeners obrađuju ulazno izlazne događaje
    - Uobičajeno je da se okidaju pre popunjavanja bean-ova
    - Uobičajeno je da izbegavaju validacionu logiku
    - Nikada direktno ne utiču na navigaciju

Tipovi event listener-a

- **ActionListener**

– Poziva se pomoću submit dugmeta, slika, i linkova sa odgovarajućim JavaScript kodom
  - <h:commandButton value="..." .../>
  - <h:commandButton image="..." .../>
  - <h:commandLink .../>

– Automatski submit-uju formu

- **ValueChangeListener**

– Poziva se pomoću combo boxes, checkboxes, radio dugmadi, tekst polja, ...
  - <h:selectOneMenu .../>
  - <h:selectBooleanCheckbox.../>
  - <h:selectOneRadio .../>
  - <h:inputText .../>

– *Ne* submit-uju formu automatski

Upotreba ActionListener-a

- **Neko dugme treba da pošalje formu i pokrene backend procese**
  - Koristi se <h:commandButton action="..." ...>
- **Druga vrsta utiče samo na UI**
  - Koristi se <h:commandButton actionListener="..." .../>
  - Uobičajeno je da se ovaj proces izvršava pre nego što se popune bean-ovi i posebno pre validacije
    - Većina formi je nekompletno popunjeno kada se UI podešava
  - Zato se korsiti atribut "immediate" da bi se naznačilo izvršavanje pre navedenih operacija

<h:commandButton actionListener="..."
immediate="true" .../>

Upotreba ActionListener-a

- **Listeneri su obično u form bean klasi**
  – Mogu biti i u odvojenoj klasi ako se koristi FacesContext da bi se dobio request ili session objekat i potražio form bean eksplicitno
  - **Definišu ActionEvent kao argument**
  – Nema vraćanja rezultata (*nije* String kao kod action controllers)
  – ActionEvent je u javax.faces.event
  – ActionEvent ima getComponent metod koji definiše UIComponent reference
  - Iz UIComponent, može se dobiti component ID, renderer, i druge informacije niskog nivoa

```
public void someMethod(ActionEvent event) {
  doSomeSideEffects();
}
```

Primer

- **Svaka UI kontrola ima "disabled" property koji je po default false**
- **Treba koristiti button za koji će biti vezan ActionListener da bi se zabranila ili odobrila upotreba određene kontrole**
- **Primer**

  – Prihvata promenljive name i job title da bi ih prikazao

  – Kreira polja za potvrdu da bi korisnik selektovao foreground i background boje

  <h:selectOneMenu value="..." disabled="..." ...>

  <f:selectItems value="..."/>

  </h:selectOneMenu>

  • Vrednost za f:selectItems mora da bude SelectItem niz

  – Postaviće se dugme koje zabranjuje ili dozvoljava rad sa poljima za potvrdu

Koraci kod formiranja JSF strana

1. **Kreira se bean sa property-ijima za podatke sa forme**
   - Par getter/setter metoda za svako polje forme
2. **Kreira se ulazna forma**
   - Kreira se f:view, h:form, h:*blah*, i h:commandButton
3. **Specificirati action controller metod**
   - Koristi se action atribut h:commandButton
4. **Kreira se action controller i action listener metode**
   - U okviru bean-a iz tačke #1.
5. **Promeniti faces-config.xml**
   - Deklarisati form bean i pravila navigacije
6. **Kreirati JSP stranice**
   - Svaku za svaki return uslov
7. **Zaštiti pristup JSP stranicama**
   - Koristiti podešavanja filtera ili bezbednosti

I korak

```
package coreservlets;
import javax.faces.model.*;
import javax.faces.event.*;

public class ResumeBean {
  ...
  private String fgColor = "BLACK";
  private String bgColor = "WHITE";
  private SelectItem[] availableColorNames =
    { new SelectItem("BLACK"),
      new SelectItem("WHITE"),
      new SelectItem("SILVER"),
      new SelectItem("RED"),
      new SelectItem("GREEN"),
      new SelectItem("BLUE") };

  public String getFgColor() { return(fgColor); }

  public void setFgColor(String fgColor) {
    this.fgColor = fgColor;
  } ...

  public SelectItem[] getAvailableColors() { ... }
```

I korak

```
...
private String name = "";
private String jobTitle = "";
...
public String getName() { return(name); }
public void setName(String name) {
this.name = name;
}
public String getJobTitle() { return(jobTitle); }
public void setJobTitle(String jobTitle) {
this.jobTitle = jobTitle;
}
```

II korak

- **Nove mogućnosti**

– Koristiti h:selectOneMenu da bi se formirao combobox

– Koristiti f:selectItems da bi se popunila lista

– Koristiti h:commandButton sa actionListener -om

• Napisati event listener kod koji se izvršava kada se klikne na dugme

– Promenit labelu (vrednost) dugmeta u zavisnosti od toga šta dugme radi

- Primer koda

```
<h:commandButton
value="#{someBean.buttonLabel}"
actionListener="#{someBean.doSideEffect}"
immediate="true"/>
```

II korak

```
<h:form>
...
Foreground color:
<h:selectOneMenu value="#{resumeBean.fgColor}"
disabled="#{!resumeBean.colorSupported}">
<f:selectItems value="#{resumeBean.availableColors}"/>
</h:selectOneMenu>
<BR>
Background color:
<h:selectOneMenu value="#{resumeBean.bgColor}"
disabled="#{!resumeBean.colorSupported}">
<f:selectItems value="#{resumeBean.availableColors}"/>
</h:selectOneMenu><BR>
<h:commandButton
value="#{resumeBean.colorSupportLabel}"
actionListener="#{resumeBean.toggleColorSupport}"
immediate="true"/>
...
</h:form>
```

III korak

```
<h:form>
...
<h:commandButton
value="#{resumeBean.colorSupportLabel}"
actionListener="#{resumeBean.toggleColorSupport}"
immediate="true"/>
...
</h:form>
```

III korak

- Isti pristup kao i u prethodnom primeru

```
<h:form>
...
Name:
<h:inputText value="#{resumeBean.name}"/><BR>
Job Title:
<h:inputText value="#{resumeBean.jobTitle}"/><P>
<h:commandButton value="Show Preview"
action="#{resumeBean.showPreview}"/>
</h:form>
```

IV korak

- **Bočni efekti**

– Logički flag povezan sa dugmetom se postavlja ako je boja podržana, ali i deaktivira oba combobox-a

```
...
public void toggleColorSupport(ActionEvent event) {
isColorSupported = !isColorSupported;
}
```

IV korak

- **Isti pristup kao u prethodnom primeru**

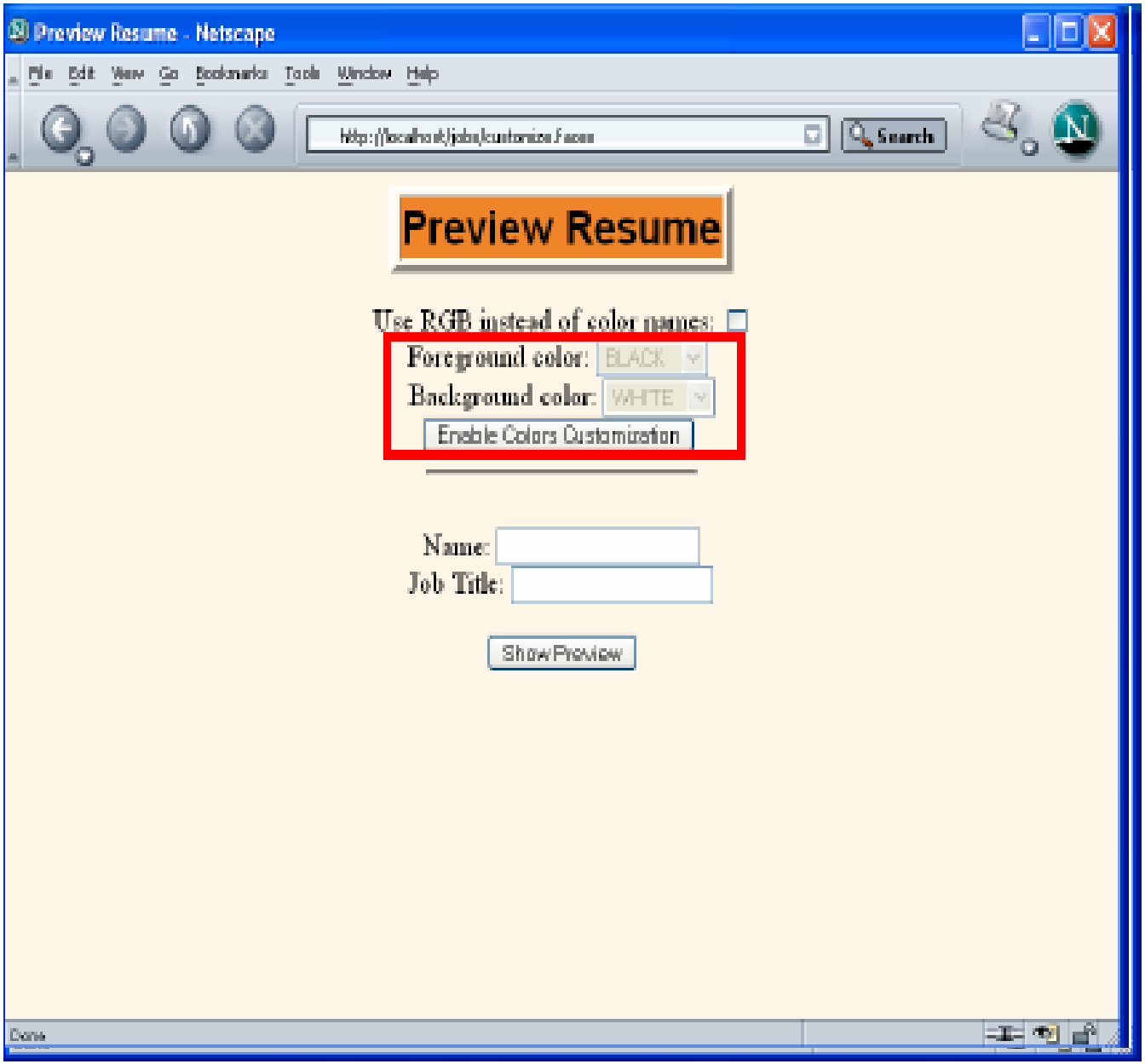
– Kod za kontrolu je unutar form bean-a (ResumeBean)

– Proverava se da li su unete boje različite

...

```
public String showPreview() {
if (isColorSupported && fgColor.equals(bgColor)) {
return("same-color");
} else {
return("success");
}
}
```

Početni rezultat



Korak 5

- Deklarisanje bean-a

```
...
<faces-config>
<managed-bean>
<managed-bean-name>resumeBean</managed-
  bean-name>
<managed-bean-class>
coreservlets.ResumeBean
</managed-bean-class>
<managed-bean-scope>session</managed-bean-
  scope>
</managed-bean>
...
</faces-config>
```

Korak 5

- **Specificiranje pravila navigacije**

```
...
<faces-config>
...
<navigation-rule>
<from-view-id>/customize.jsp</from-view-id>
<navigation-case>
<from-outcome>same-color</from-outcome>
<to-view-id>/WEB-INF/results/same-color.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>success</from-outcome>
<to-view-id>/WEB-INF/results/show-preview.jsp</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>
```

Korak 6

- **WEB-INF/results/same-color.jsp**

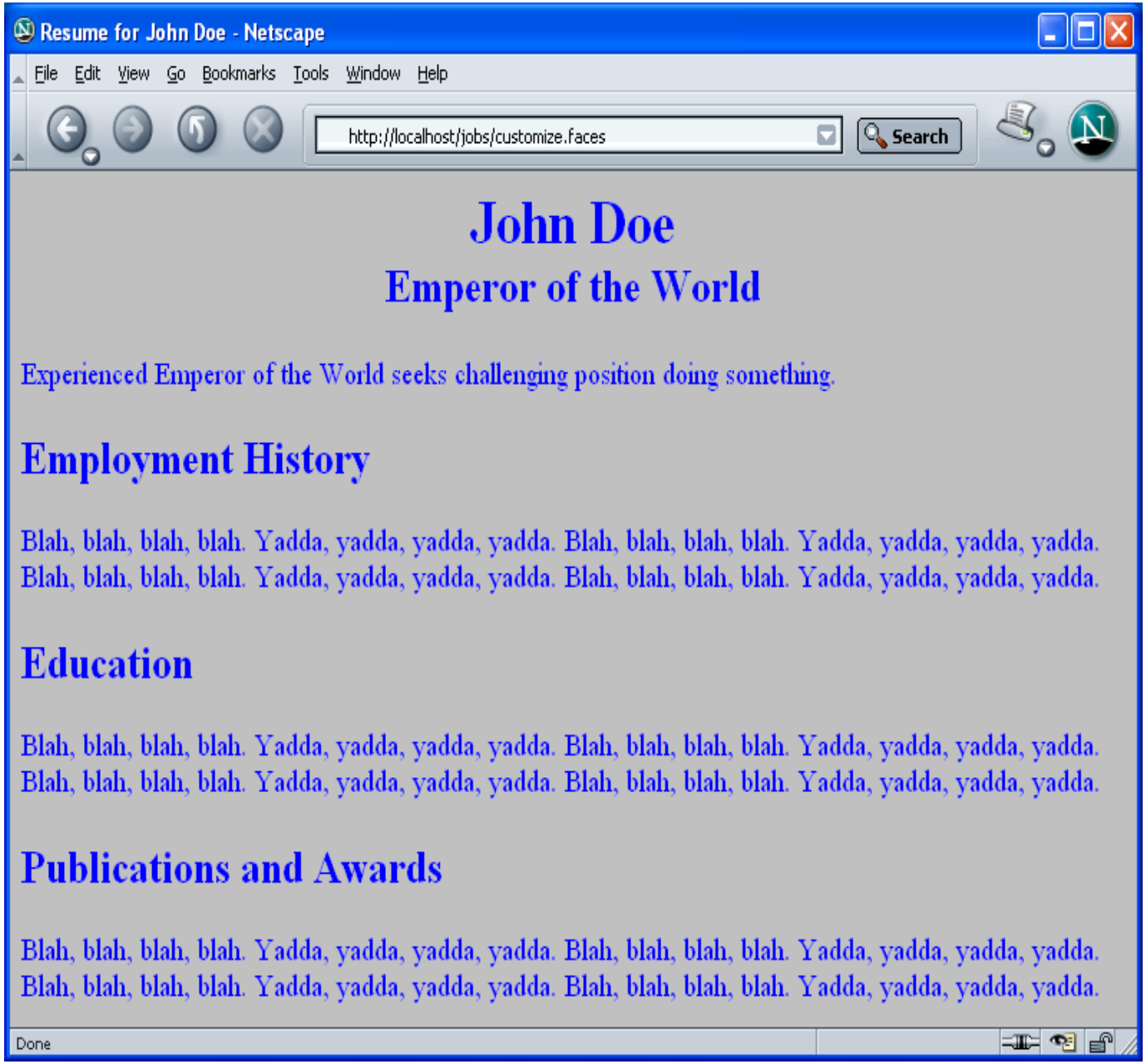
```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<!DOCTYPE ...>
<HTML>
...
You chose
"<h:outputText value="#{resumeBean.fgColor}"/>"
as both the foreground and background color.
<P>
You don't deserve to get a job, but I suppose
we will let you <A HREF="customize.faces">try again</A>.
...
</HTML>
</f:view>
```

Korak 6

- WEB-INF/results/show-preview.jsp

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<!DOCTYPE ...>
<HTML>
...
<BODY TEXT="<h:outputText value="#{resumeBean.fgColor}"/>"
BGCOLOR="<h:outputText value="#{resumeBean.bgColor}"/>">
<H1 ALIGN="CENTER">
<h:outputText value="#{resumeBean.name}"/><BR>
<SMALL><h:outputText value="#{resumeBean.jobTitle}"/>
</SMALL></H1>
Experienced <h:outputText value="#{resumeBean.jobTitle}"/>
seeks challenging position doing something.
<H2>Employment History</H2>
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
...
</HTML>
</f:view>
```

Korak 6


Resume for John Doe - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/jobs/customize.faces
Search
John Doe
Emperor of the World
Experienced Emperor of the World seeks challenging position doing something.
Employment History
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
Education
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
Publications and Awards
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
Done

Korak 7

```
<web-app>
...
<security-constraint>
<display-name>
Prevent access to raw JSP pages that are for JSF pages.
</display-name>
<web-resource-collection>
<web-resource-name>Raw-JSF-JSP-Pages</web-resource-
   name>
<!-- Add url-pattern for EACH raw JSP page -->
<url-pattern>/customize.jsp</url-pattern>
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Upotreba ValueChangeListener-a u JSP

- **ActionListener je vezan za dugme**
  – Forma se automatski šalje kada se dugme pritisne
- **ValueChangeListener je povezan za combobox, listbox, radio button, checkbox, textfield, ...**
  – Forma se šalje automatski
  – Potrebno je dodati JavaScript da bi se forma poslala
  onclick="submit()" or onchange="submit()"
  – Postoji i nekompatibilnost između Netscape i IE
  • Netscape, Mozilla, i Opera okidaju onchange događaj kada se promeni selekcija combobox-a, kada je radio button selektovan, ili kada checkbox promeni vrednost
  • Internet Explorer okida događaj nakon promene selekcije, ali samo kada neka druga GUI kontrola prihvati fokus

Upotreba ValueChangeListener-a u JSP

- **Listener je obično u klasi form bean-a**
  - Diskutovano ranije
- **Prihvata ValueChangeEvent kao argument**
  - Postoje korisni ValueChangeEvent metodi
    - getComponent ()
    - getOldValue (prethodna vrednost GUI elementa)
    - getNewValue (trenutna vrednost GUI elementa)
  - Potrebno dok bean možda još nije popunjen
  - Vrednost za checkbox je tipa Boolean
  - Vrednost za radio button ili tekstualno polje je povezano sa parametrima zahteva
  - Primer

```
public void someMethod(ValueChangeEvent event) {
boolean flag =
((Boolean)event.getNewValue()).booleanValue();
takeActionBasedOn(flag);
}
```

Primer: promena boje

- **h:selectOneMenu koristi f:selectItems da bi dobio listu combobox ulaza**
- **Koristi se checkbox da bi se pozvao ValueChangeListener**
- **Postoje dve definisane liste**
  - Imena boja
  - RGB vrednosti

JSF koraci

1. **Kreira se bean sa property-ijima za podatke sa forme**
2. **Kreira se ulazna forma pomoću** f:view i h:form
3. **Specificirati action controller metod pomoću action atributa od h:commandButton**
4. **Kreiraju se action controller-i i drugi tipovi event listener-a**
5. **Promeniti faces-config.xml da bi se deklarisali form bean i pravila navigacije**
6. **Kreirati JSP stranice za svaki return uslov**
7. **Zaštiti pristup JSP stranicama**

Korak 1

```
private SelectItem[] availableColorNames =
  { new SelectItem("BLACK"),
    new SelectItem("WHITE"),
    new SelectItem("SILVER"),
    new SelectItem("RED"),
    new SelectItem("GREEN"),
    new SelectItem("BLUE") };
private SelectItem[] availableColorValues =
    { new SelectItem("#000000"),
      new SelectItem("#FFFFFF"),
      new SelectItem("#C0C0C0"),
      new SelectItem("#FF0000"),
      new SelectItem("#00FF00"),
      new SelectItem("#0000FF") };
private boolean isUsingColorNames = true;

public SelectItem[] getAvailableColors() {
  if (isUsingColorNames) {
    return(availableColorNames);
  } else {
    return(availableColorValues);
```

Koraci 2 i 3

- **Nove mogućnosti**

– Koristiti h:selectBooleanCheckbox da bi se kreirao checkbox

– Koristiti valueChangeListener za realizaciju event listener koda

– Koristiti onclick da bi se automatski slala fomra kada checkbox menja vrednosti

- **Primer**

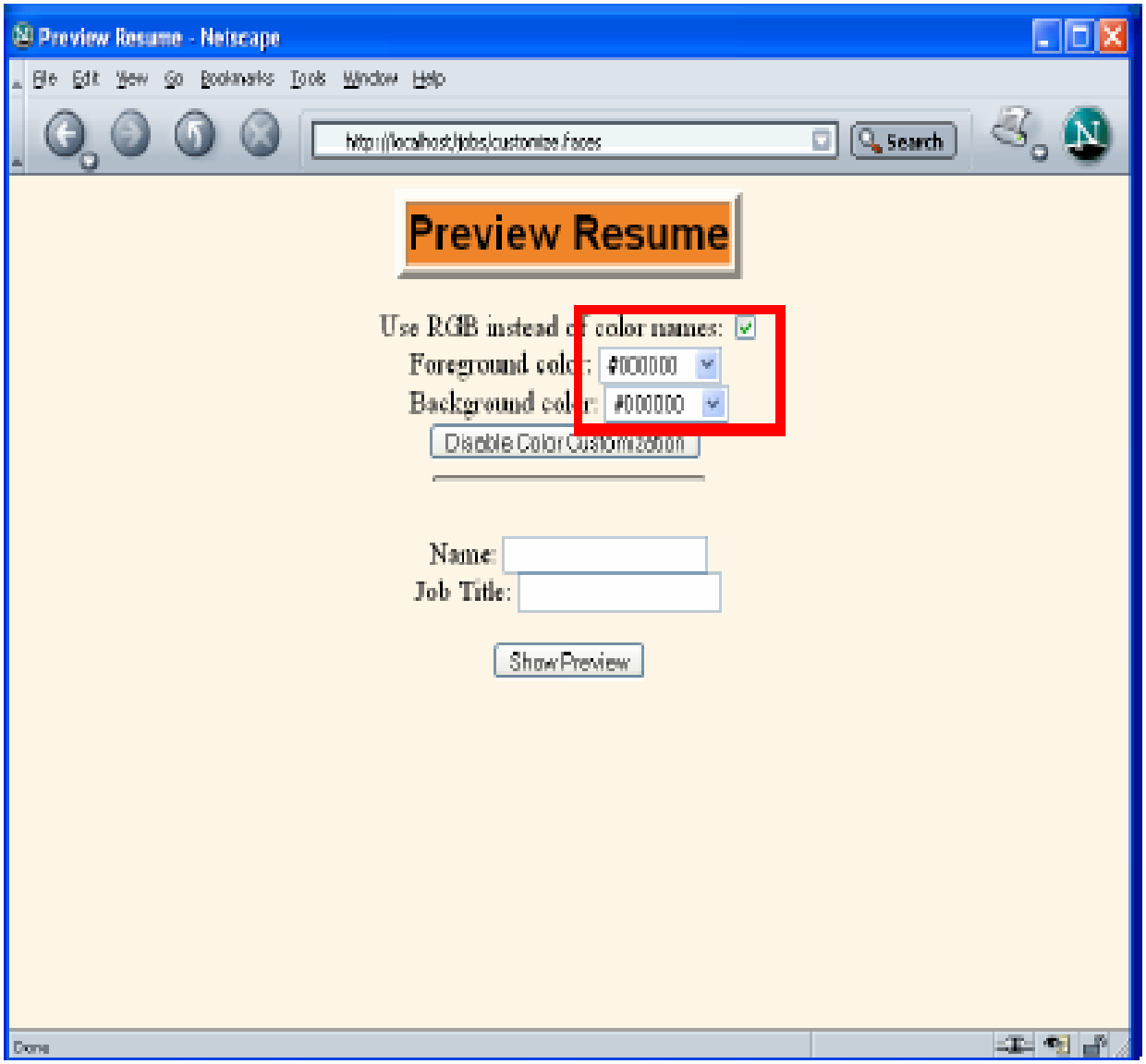
```
<h:selectBooleanCheckbox
valueChangeListener="#{resumeBean.changeColorMode}"
onclick="submit()"
immediate="true"/>
```

Korak 4

- Kada je checkbox selektovan, menjaju se izbori boja da bi se dobile RGB vrednosti umesto imena boja

```
public void changeColorMode(ValueChangeEvent
    event) {
boolean flag =
((Boolean)event.getNewValue()).booleanValue();
setUsingColorNames(!flag);
}
```

Rezultati


Preview Resume - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/jobs/customize.faces
Search
Preview Resume
Use RGB instead of color names:
Foreground color:
#000000
Background color:
#000000
Disable Color Customization
Name:
Job Title:
Show Preview
Done

Zajednička upotreba action listener-a i action controller-a

- **Obično, action listener-i i action controller-i su povezani sa različitim objektima**
- **Nekada je potrebno koristiti oba**

  – Action listener-i imaju pristup do niskog nivoa detalja GUI objekata
- **Najčešći primeri:**
- **Mape slika na serverskoj strani**

  – h:commandButton sa slikom umesto vrednosti rezultuje u mapi slike

  – Trenutna dolazeća imena parametara zahteva su *clientID*.x i *clientID*.y
- Samo listener može otkriti ID klijenta

Primer: izbor pozadine sa mape slike

- **Prikazati paletu boja korisniku**
- **Dozvoiliti mu da klikne na željenu boju za pozadinu**
- **Naći odgovarajuću RGB vrednost**
  - Pročitati *clientID*.x vrednost
  - Normalizovati je na 0-256 u zavisnosti od širine slike
  - Prikazati izlaz kao hex cifre unutar #RRGGBB stringa
  - Uraditi sve u okviru action listener-a
- **Proslediti JSP stranici**
  - Uraditi u okviru action controller

Korak 1

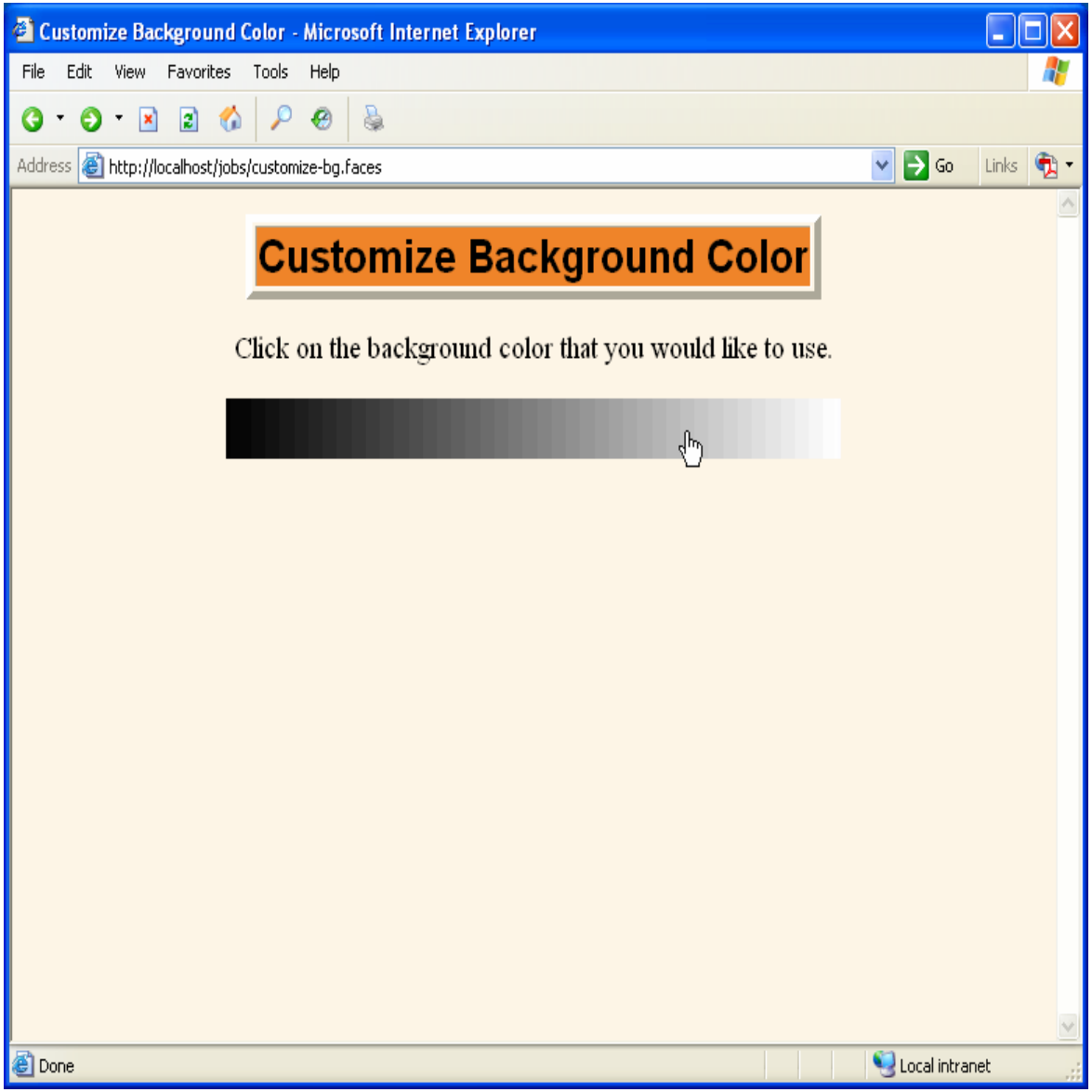
```
package coreservlets;
...
public class ColorBean {
private String bgColor = "WHITE";
public String getBgColor() { return(bgColor); }
public void setBgColor(String bgColor) {
this.bgColor = bgColor;
}
```

Korak 2

- **Koristiti h:commandButton sa slikom umesto vrednosti**
  - Rezultat je mapa slike na sereverskoj strani
- **Kod**

```
<h:commandButton
image="images/GrayBar.gif"
actionListener="#{colorBean.selectGrayLevel}"
action="#{colorBean.showPreview}"/>
```

Korak 2


Customize Background Color - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/jobs/customize-bg.faces
Go
Links
Customize Background Color
Click on the background color that you would like to use.
Done
Local intranet

Korak 3

- Koristiti i actionListener i action u okviru istog dugmeta

```
<h:form>
...
<h:commandButton
image="images/GrayBar.gif"
actionListener="#{colorBean.selectGrayLevel}"
action="#{colorBean.showPreview}"/>
</h:form>
```

Korak 3

- Listener traži x vrednost i kreira boju

```
public void selectGrayLevel(ActionEvent event) {
FacesContext context = FacesContext.getCurrentInstance();
String clientId = event.getComponent().getClientId(context);
HttpServletRequest request =
(HttpServletRequest)context.getExternalContext().getRequest();
String grayLevelString = request.getParameter(clientId + ".x");
int grayLevel = 220;
try {
grayLevel = Integer.parseInt(grayLevelString);
} catch(NumberFormatException nfe) {}
// Image is 440 pixels, so scale.
grayLevel = (256 * grayLevel) / 440;
String rVal = Integer.toString(grayLevel, 16);
setBgColor("#" + rVal + rVal + rVal);
}
```

- Controller definiše izlaznu stranu

```
public String showPreview() {
return("success");
}
```

Korak 5

- **Declarisanje bean-a**

```
...
<faces-config>
...
<managed-bean>
<managed-bean-name>colorBean</managed-bean-name>
<managed-bean-class>
coreservlets.ColorBean
</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
...
</faces-config>
```

Korak 5

- **Specifikacija pravila navigacije**

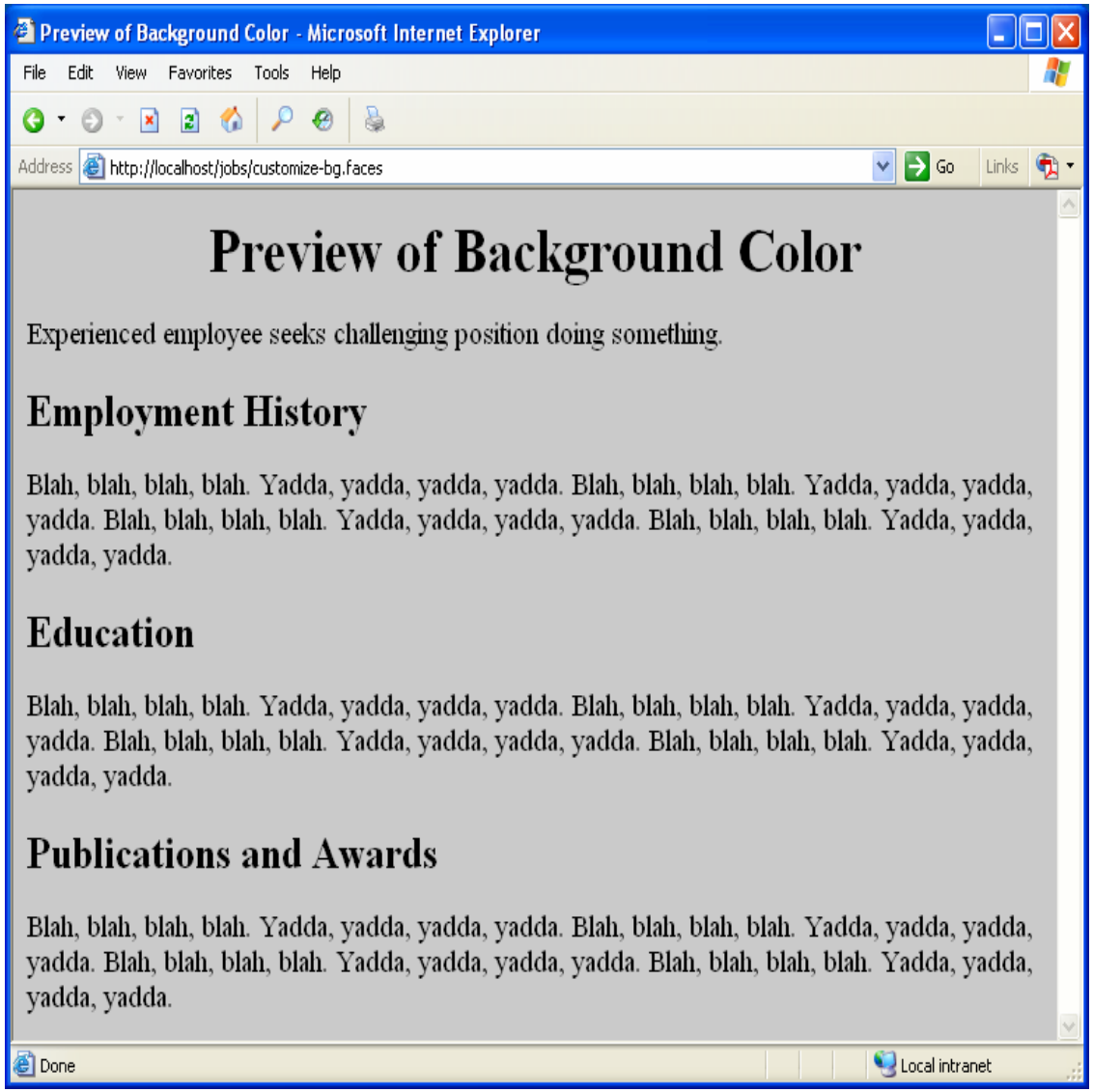
```
...
<faces-config>
...
<navigation-rule>
<from-view-id>/customize-bg.jsp</from-view-id>
<navigation-case>
<from-outcome>success</from-outcome>
<to-view-id>
/WEB-INF/results/show-preview2.jsp
</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>
```

Korak 6

- WEB-INF/results/show-preview2.jsp

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Preview of Background
    Color</TITLE></HEAD>
<BODY BGCOLOR="<h:outputText
    value="#{colorBean.bgColor}"/>">
<H1 ALIGN="CENTER">Preview of Background Color</H1>
Experienced employee
seeks challenging position doing something.
<H2>Employment History</H2>
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
...
</HTML>
</f:view>
```

Rezultat


Preview of Background Color - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/jobs/customize-bg.faces
Go
Links
Preview of Background Color
Experienced employee seeks challenging position doing something.
Employment History
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
Education
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
Publications and Awards
Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda. Blah, blah, blah, blah. Yadda, yadda, yadda, yadda.
Done
Local intranet

Korak 7

```
<web-app>
...
<security-constraint>
<display-name>
Prevent access to raw JSP pages that are for JSF pages.
</display-name>
<web-resource-collection>
<web-resource-name>Raw-JSF-JSP-Pages</web-resource-
   name>
<!-- Add url-pattern for EACH raw JSP page -->
<url-pattern>/customize.jsp</url-pattern>
<url-pattern>/customize-bg.jsp</url-pattern>
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Validacija

- **Ručna validacija**

– Koriste se string property-iji za bean
– Sprovede se validacija u setter metodama i/ili action controller-ima
– Vraća se null da bi se ponovo prikazala forma
– Kreiraju se odgovarjuće poruke o greškama

- **Implicitna automatska validacija**

– Koriste se int, double, ... bean property-iji. Ili required.
– Sistem ponovo prikazuje formu, ako postoji greška prilikom konverzije
– Koristi se h:message da se prikaže poruka za određeno polje

- **Eksplicitna validacija**

– Koriste se f:convertNumber, f:convertDateTime, f:validateLength, f:validateDoubleRange, ili f:validateLongRange
– Sistem ponovo prikazuje forma u slučaju greške; opet h:message

- **Kreiranje sopstvenih metoda validacija**
- **Kreiranje uobičajenih validacija**

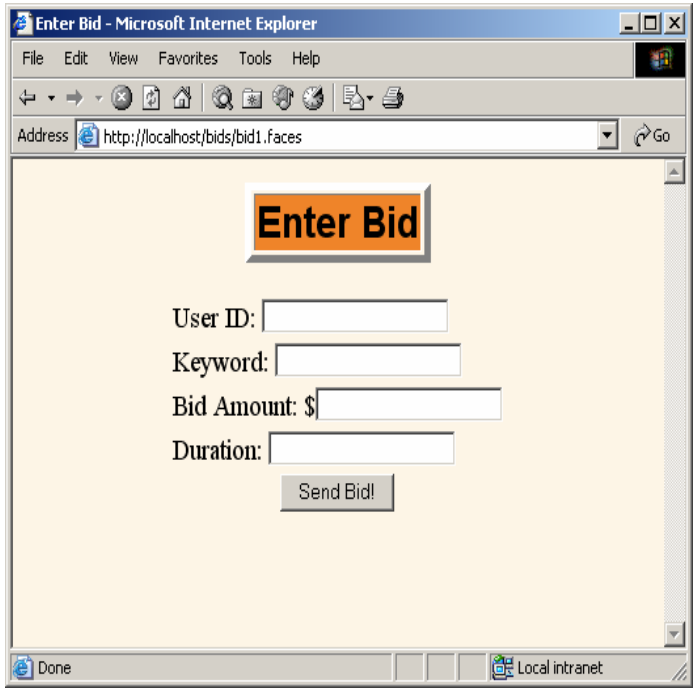
– Kreira se FacesMessage, u okviru ValidatorException, throw

Ručna validacija

- **Setter metodi konvertuju iz stringova**
  - Koriste se try/catch blokovi
  - Koristi se logika specifična za aplikaciju
- **Action controller proverava vrednosti**
  - Ako su vrednosti u redu, vraća se uobičajeni izlaz
  - Ako vrednosti nedostaju ili su nelegalne, smešta se poruka o grešci u bean i vraća se null
- **Početna forma**
  - Prikazuje poruke o grešci
    - Poruke su prazni stringovi po defaultu
    - u okviru h:outputText, treba koristiti escape="false" ako poruka sadrži HTML tagove

Ručna validacija - primer

- **Ideja**
  - Skupljaju se cene za ključne reči na search engine sajtu
- **Atributi**
  - UserID
    - Ne može da nedostaje
  - Keyword
    - Ne može da nedostaje
  - Cena
    - Mora da bude legalan racionalni broj
    - Mora da bude najmanje 0.10
  - Trajanje
    - Mora da bude legalan int
    - Mora da bude najmanje 15



Bean kod

```
package coreservlets;
import java.util.ArrayList;
public class BidBean1 {
private String userID = "";
private String keyword = "";
private String bidAmount;
private double numericBidAmount = 0;
private String bidDuration;
private int numericBidDuration = 0;
private ArrayList errorMessages = new ArrayList();
public String getUserID() { return(userID); }
    public void setUserID(String userID) {
        this.userID = userID.trim();
    }
    public String getKeyword() { return(keyword); }
    public void setKeyword(String keyword) {
        this.keyword = keyword.trim();
}
```

Bean kod

```
public String getBidAmount() { return(bidAmount); }
public void setBidAmount(String bidAmount) {
this.bidAmount = bidAmount;
try {
numericBidAmount = Double.parseDouble(bidAmount);
} catch(NumberFormatException nfe) {}
}
public double getNumericBidAmount() {
return(numericBidAmount);
}
public String getBidDuration() { return(bidDuration); }
public void setBidDuration(String bidDuration) {
this.bidDuration = bidDuration;
try {
numericBidDuration = Integer.parseInt(bidDuration);
} catch(NumberFormatException nfe) {}
}
public int getNumericBidDuration() {
return(numericBidDuration);
}
```

Bean kod – Action Controller

```
public String doBid() {
errorMessages = new ArrayList();
if (getUserID().equals("")) {
    errorMessages.add("UserID required");
}
if (getKeyword().equals("")) {
    errorMessages.add("Keyword required");
}
if (getNumericBidAmount() <= 0.10) {
    errorMessages.add("Bid amount must be at least $0.10.");
}
if (getNumericBidDuration() < 15) {
    errorMessages.add("Duration must be at least 15 days.");
}
if (errorMessages.size() > 0) {
    return(null);
} else {
return("success");
}
}
```

Bean kod – Error message

```
public String getErrorMessages() {
String message;
if (errorMessages.size() == 0) {
    message = "";
} else {
message = "<FONT COLOR=RED><B><UL>\n";
for(int i=0; i<errorMessages.size(); i++) {
message = message + "<LI>" +
(String)errorMessages.get(i) + "\n";
}
message = message + "</UL></B></FONT>\n";
}
return(message);
}
```

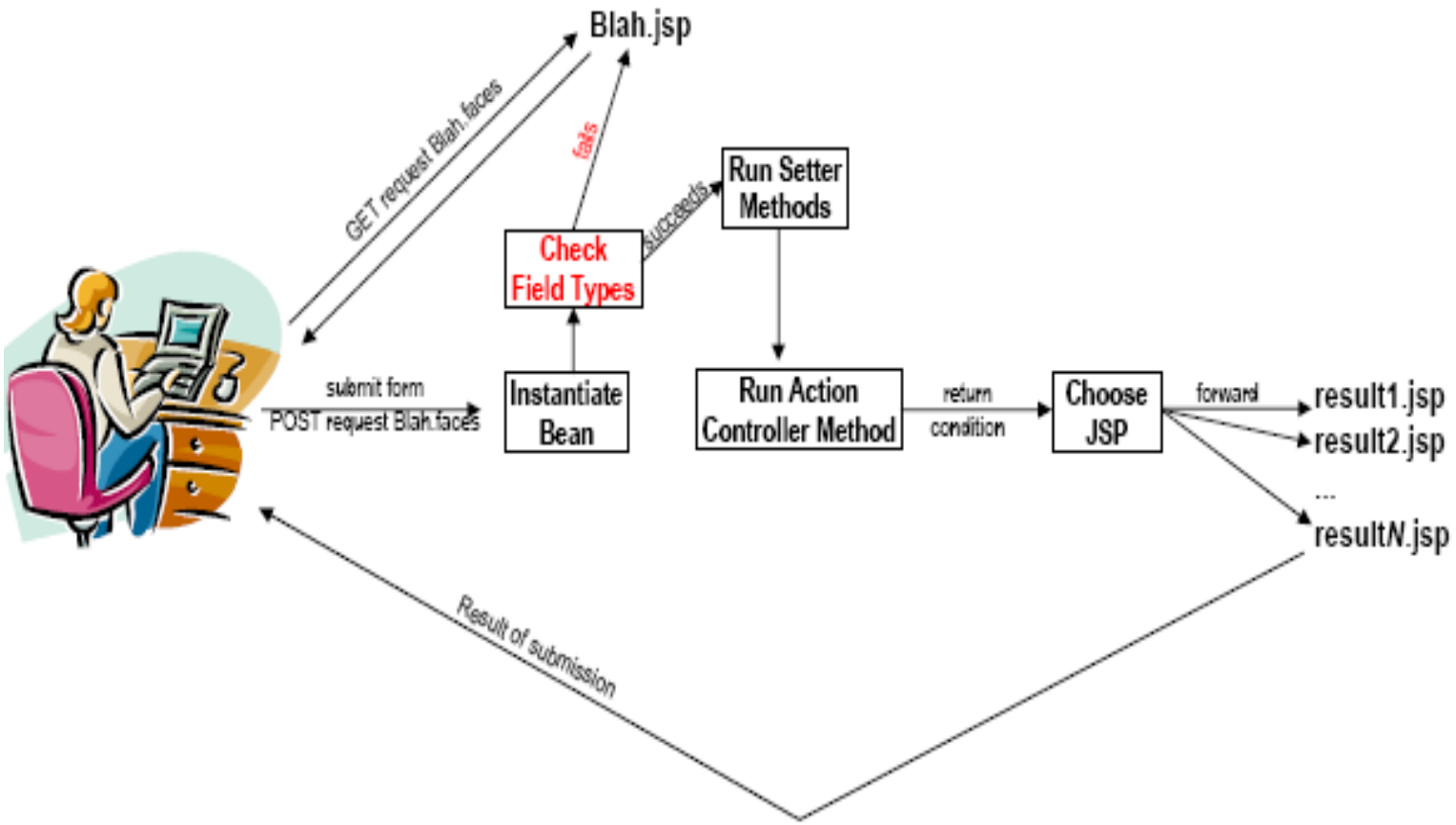
bid1.jsp

```
<h:form>
<h:outputText value="#{bidBean1.errorMessages}‟ escape="false"/>
<TABLE>
<TR>
<TD>User ID:
<h:inputText value="#{bidBean1.userID}"/></TD></TR>
<TR>
<TD>Keyword:
<h:inputText value="#{bidBean1.keyword}"/></TD></TR>
<TR>
<TD>Bid Amount:
$<h:inputText value="#{bidBean1.bidAmount}"/></TD></TR>
<TR>
<TD>Duration:
<h:inputText value="#{bidBean1.bidDuration}"/></TD></TR>
<TR><TH>
<h:commandButton value="Send Bid!"
action="#{bidBean1.doBid}"/></TH></TR>
</TABLE>
</h:form>
```

Rezultat


Bid Accepted - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/bids/bid1.faces
Go
Bid Accepted
You have bid successfully.
• User ID: a-123
• Keyword: jsf
• Bid Amount: $0.20
• Duration: 90
(Version 1)
Done
Local intranet

JSF pojednostavljena kontrola toka



Implicitna automatska validacija

- **Definišu se bean property-iji na standardne tipove podataka**
  - int, long, double, boolean, char, ...
- **Sistem pokušava da izvrši automatsku konverziju, kao kod jsp:setProperty**
  - Integer.parseInt, Double.parseDouble, ...
- **Ako postoji greška, forma se ponovo prikazuje**
  - I smešta se poruka o grešci
- **Može se dodati atribut required za svaki ulazni element, da bi se specificiralo da prazne vrednosti predstavljaju grešku**
  - Koristi se **h:message da bi se prikazale poruke i greškama**
  - h:message vraća prazan string ako nema poruke
  - h:message prihvata styleClass za CSS ime stila
- **Dodaje se atribut immediate da bi se preskočila validacija**
  - Na primer za h:commandButton sa logout ili cancel operacijama

Implicitna automatska validacija - primer

- **Promena BidBean property-ija**
  - bidAmount je double
  - bidDuration je int
- **Promena BidBean action controller**
  - Uklanja se sva validaciona logika.
  - Treba primeniti da se specifična vrednost za cenu i trajanja više ne proverava
- **Promena ulazne forme**
  - Dodaje se atribut required za userID i keyword polja
  - Dodaje se atribut id za svako polje
  - Dodaje se h:message (sa odgovarajućim id) za svako ulazno polje

Bean kod

```
package coreservlets;
public class BidBean2 {
private String userID;
private String keyword;
private double bidAmount;
private int bidDuration;
...
public double getBidAmount() { return(bidAmount); }
public void setBidAmount(double bidAmount) {
this.bidAmount = bidAmount;
}
public int getBidDuration() { return(bidDuration); }
public void setBidDuration(int bidDuration) {
this.bidDuration = bidDuration;
}
```

Bean kod – Action Controller

```
public String doBid() {
    return("success");
}
```

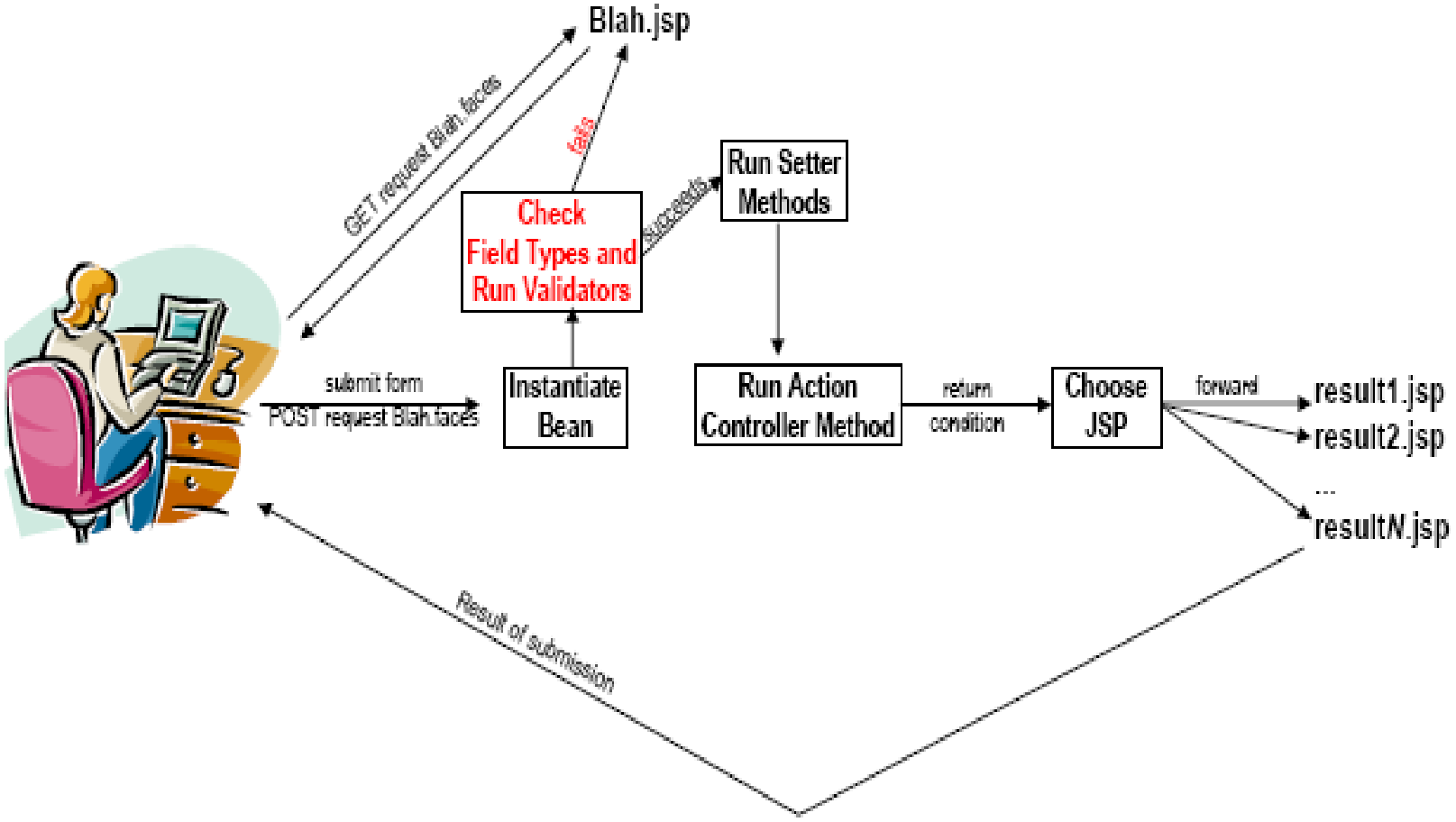
Ulazna forma

```
<h:form>
<TABLE>
<TR>
<TD>User ID: <h:inputText value="#{bidBean2.userID}"
required="true" id="userID"/></TD>
<TD><h:message for="userID" styleClass="RED"/></TD></TR>
<TR>
<TD>Keyword: <h:inputText value="#{bidBean2.keyword}"
required="true" id="keyword"/></TD>
<TD><h:message for="keyword" styleClass="RED"/></TD></TR>
<TR>
<TD>Bid Amount: $<h:inputText value="#{bidBean2.bidAmount}"
id="amount"/></TD>
<TD><h:message for="amount" styleClass="RED"/></TD></TR>
<TR>
<TD>Duration: <h:inputText value="#{bidBean2.bidDuration}"
id="duration"/></TD>
<TD><h:message for="duration" styleClass="RED"/></TD></TR>
...</TABLE>...</h:form>
```

Rezultat


Bid Accepted - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/bids/bid2.faces
Go
Bid Accepted
You have bid successfully.
• User ID: a-123
• Keyword: jsf
• Bid Amount: $1.25
• Duration: 30
(Version 2)
Local intranet

JSF kontola toka



Eksplicitna validacija

- **Definisati bean property-ije na proste tipove podataka**
- – int, long, double, boolean, char, ...
- **Dodaje se f:validate*Blah* ili f:convert*Blah* elementima**
  - **Sistem proverava da li polja odgovaraju pravilima**
  - – f:validate*Blah* atributi dozvoljavaju da se kontroliše format
- **Ako postoji greška prilikom validacije, forma se ponovo prikazuje**
  - – I smešta se poruka o grešci
- **Ostale mogućnosti su iste**
  - – I dalje se može dodati atribut required za bilo koji ulazni element
  - – I dalje se sa use h:message prikazuju poruke
  - • h:message vraća prazan string ako nema poruke
  - – I dalje se dodaje atribut immediate attribute da bi se izbegla validacija

Eksplicitna validacija primer

- **BidBean ostaje nepromenjen u odnosu na prethodni primer**
- **Promena ulazne forme**

– Pretpostaviti da userID mora biti 5 ili 6 karaktera
– Pretpostaviti da keyword mora biti 3 ili više karaktera
– Pretpostaviti da bid amount mora biti najmanje 10 centi
– Pretpostaviti da bid duration mora biti najmanje 15 dana

Početna forma

```
<h:form>
<TABLE>
<TR>
<TD>User ID:
<h:inputText value="#{bidBean2.userID}" id="userID">
<f:validateLength minimum="5" maximum="6"/>
</h:inputText></TD>
<TD><h:message for="userID" tyleClass="RED"/></TD></TR>
<TR>
<TD>Keyword:
<h:inputText value="#{bidBean2.keyword}" id="keyword">
<f:validateLength minimum="3"/>
</h:inputText></TD>
<TD><h:message for="keyword"
   styleClass="RED"/></TD></TR>
```

Početna forma

```
<TR>
<TD>Bid Amount:
$<h:inputText value="#{bidBean2.bidAmount}" id="amount">
<f:validateDoubleRange minimum="0.10"/>
</h:inputText></TD>
<TD><h:message for="amount"
    styleClass="RED"/></TD></TR>
<TR>
<TD>Duration:
<h:inputText value="#{bidBean2.bidDuration}" id="duration">
<f:validateLongRange minimum="15"/>
</h:inputText></TD>
<TD><h:message for="duration" styleClass="RED"/>
    </TD></TR>
<TR><TH COLSPAN=2>
<h:commandButton value="Send Bid!"
    action="#{bidBean2.doBid}"/></TH></TR>
</TABLE>
```

Rezultat


Bid Accepted - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/bids/bid3.faces
Go
Bid Accepted
You have bid successfully.
• User ID: a-123
• Keyword: jsf
• Bid Amount: $1.25
• Duration: 90
(Version 3)
Done
Local intranet

Konverzija vs validacija

- **I f:convert*Blah* i f:validate*Blah* proveravaju format i prikazuju ponovo formu u slučaju greške**
  - Ali f:convert*Blah* takođe i menja format u kome se polje prikazuje
  - f:validate*Blah* ima smisla samo sa h:inputText
  - f:convert*Blah* ima smisla koristiti i sa h:inputText i sa h:outputText
- **Primer**

```
<h:inputText value="#{orderBean.discountCode}">
   <f:convertNumber maxFractionDigits="2"/>
</h:inputText>
```

  - Prikazuje 0.75 a ne 0.749

Atributi

- **f:convertNumber**
  - currencyCode, currencySymbol
  - groupingUsed
  - integerOnly
  - locale
  - max(min)FractionDigits
  - max(min)IntegerDigits
  - pattern (ala DecimalFormat)
  - type
    - number, currency, percentage
- **f:convertDateTime**
  - type
    - date, time, both
  - dateStyle, timeStyle
    - default, short, medium, long,full
  - pattern (ala SimpleDateFormat)
  - locale
  - timeZone
- **f:validateLength**
  - minimum
  - maximum
- **f:validateLongRange**
  - minimum
  - maximum
- **f:validateDoubleRange**
  - minimum
  - maximum

Uobičajene poruke o greškama

- **Kreira se i učita globalni properties fajl**

<application>

<message-bundle>...</message-bundle>

</application>

- **Koristiti eksplicitna imena property-ija kao što su navedena u Messages.properties**
  - javax.faces.component.UIInput.CONVERSION
  - javax.faces.component.UIInput.REQUIRED
- **Primer**

javax.faces.component.UIInput.CONVERSION=Illegal format!

javax.faces.component.UIInput.REQUIRED=Missing value!

Pisanje sopstvenih metoda validacije

- **JSP**
  – Za ulaznu komponentu eksplicitno specificirati ime metoda

<h:inputText id="someID“ validator="#{someBean.someMethod}"/>

  – Koristi se h:message kao i ranije

<h:message for="someID"/>

- **Java**
  – Pomoću ValidatorException sa FacesMessage ako validacija nije uspela.
  - Ništa ne raditi ako je validacija uspela.
  – Argumenti metoda:
  • FacesContext
  – Sadržaj
  • UIComponent
  – Nad komponentom će se izvršiti validacija
  • Object
  – Poslata vrednost (primitive koriste omotače)

Pisanje sopstvenih metoda validacije - JSP

```
...
<TR>
<TD>Bid Amount:
$<h:inputText value="#{bidBean2.bidAmount}"
id="amount" required="true"
validator="#{bidBean2.validateBidAmount}"/>
</TD>
<TD><h:message for="amount"
   styleClass="RED"/></TD></TR>
...
```

Pisanje sopstvenih metoda validacije - Java

```
public void validateBidAmount(FacesContext context,
UIComponent componentToValidate,
Object value)
throws ValidatorException {
double bidAmount = ((Double)value).doubleValue();
double previousHighestBid = currentHighestBid();
if (bidAmount <= previousHighestBid) {
FacesMessage message =
new FacesMessage("Bid must be higher than current" +
"highest bid ($" +previousHighestBid + ").");
throw new ValidatorException(message);
}
}
public double currentHighestBid() {
return(0.23); // Get from database in real life
}
```

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 smješta 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 – podrazumivano
  - @form – h:form koji okražuje f:ajax – kada je potrebno da se pošalju vrijednosti 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 promjenljive 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 promjenljive 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 promjenljive 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 izmjenu u grafičkom dizajnu (makar i najmanju)**

Looping

- **rad sa sadržajem promjenljive 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 posljednji 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.ba" 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