Programiranje Internet aplikacija
Java Server Faces (JSF)

Šta je JSF?

- **Skup Web-baziranih GUI kontrola i povezanih obrada?**

  – JSF omogućava veliki broj HTML-orijentisanih GUI kontrola, zajedno sa kodom koji obrađuje njihove događaje.
- **Nezavisan GUI kontrolni framework?**

  – JSF imaju mogućnost da generišu i grafičke kontrole koje nisu HTML, koje koriste protokole koji nisu HTTP
- **U odnosu na Struts?**

  – Kao i Struts, JSF se može koristiti kao MVC framework za generisanje HTML formi, validaciju njihovih vrednosti, realizaciju poslovne logike i prikazivanje rezultata.
- **Koja od navedenih je najvažnija osobina JSF?**

  – Odgovor zavisi od načina upotrebe, ali razlozi 1 i 3 su najčešći za upotrebu 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.**
- **Tada je potrebno koristiti i dodatne JSF biblioteke (videti na http://javaserverfaces.dev.java.net**

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.**

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.

Prednosti JSF (u odnosu na Struts)

- **Ugrađene komponente**

  – U okviru JSF relativno je jednostavno kombinovati kompleksne GUI kontrole u okviru jedne komponente; Struts to nije u mogućnosti
- **Podrška za druge tehnologije razvoja interfejsa**

  – JSF nije limitiran na HTML i HTTP; Struts jeste
- **Pristup beanu pomoću imena**

  – JSF dozvoljava da se bean-u dodeli ime, i da se poziva po imenu u okviru formi. Struts ima složen proces sa nekoliko nivoa direkcije, gde se mora pamtiti koja forma je ulazna za svaku akciju.
- **Expression jezik**

  – JSF expression jezik je više koncizan i moćan nego Struts bean:write tag.

Prednosti JSF (u odnosu na Struts)

- **Jednostavnije definicije kontrolera i bean-a**

  – JSF ne zahteva da klase kontrolera i bean-a nasleđuju određenu parent klasu (n.p., Action) ili da koriste određeni metod (n.p., execute). Struts zahteva.
- **Jednostavniji config fajlovi i struktura sistema**

  – faces-config.xml fajl je mnogo jednostavniji za upotrebu nego struts-config.xml file. Generalno, JSF je jednostavniji.
- **Veći razvoj alata za implementaciju**

  – Pristup sa GUI kontrolama i njihovim obradama, otvara mogućnost jednostavnijeg korišćenja drag-and-drop razvojnih okruženja

Nedostaci JSF (u odnosu na Struts)

- **Ne postoji ekvivalent Tiles tehnologiji**
  - – Struts pruža podršku za realizaciju layout-a stranice; JSF ne
  - • Ali se može iskoristiti Tiles deo iz Struts i korisititi u okviru JSF
- **Dosta slabija automatska validacija**
  - – Struts sadrži validatore za email adrese, brojeve kreditne kartice, regularne izraze, ... JSF sadrži validatore samo za neunete vrednosti, dužinu unosa, i da li je broj u datom opsegu
  - • Ali MyFaces sadrže nekoliko moćnih validatora
- **Nedostatak validacije na klijentskoj strani**
  - – Struts podržava JavaScript-baziranu validaciju polja formi; JSF ne
  - • Shale validatori dodaju validaciju na klijentskoj strani JSF
- **Otežana instalacija**
  - – JSF nema ekvivalnet za struts-blank
- **Samo POST**
  - – JSF ne podržava GET metod

Postupak konfiguracije JSF

- **JSF aplikacija se instaliraju u obliku WAR fajla – zapakovane aplikacije sa ekstenzijom war i strukturom direktorijuma sa sledećim standardizovanim rasporedom:**

project-name

HTML i JSP stranice

/WEB-INF

konfiguracioni fajlovi

/classes

Javine klase

/lib

Dodatne biblioteke

Postupak konfiguracije JSF

- **U okviru predložene Javine BluePrints dokumentacije (http://java.sun.com/blueprints/code/projectconvertions.com) se preporučuje malo promenjena struktura.**
- **Preporučuje se da se i sam kod čuva u direktorijumu src/java , a da se JSF stranice i konfiguracioni fajlovi čuvju u web direktorijumu.**

project-name

/source

/java

/com

/mojPaket

/korisnikBean.java

/web

Html, jsp, …

/WEB-INF

/classes

/lib

XML fajlovi

Postupak konfiguracije JSF

Direktorijum ili fajl	Objašnjenje
project-name	Direktorijum projekta koji se zove kao i projekat
/source	Ovde se smeštaju java source klase i .properties fajlovi
/web	Ovde se nalaze fajlovi web aplikacije koje koristi server ili servlet container.
/WEB-INF	Ovaj folder sadrži fajlove web aplikacije koji se koriste u runtime-u ali su sakriveni od browser-a
/classes	Ovde se nalaze kompajlirane java klase kao i .proprties fajlovi iskopirani iz JavaSource-a
/lib	Ovde se nalaze biblioteke potrebne za rad aplikacije

Dokumentacija

- **API Javadocs**
  – http://java.sun.com/j2ee/javaserverfaces/1.1/docs/api/
- **Tag library dokumenta**
- http://java.sun.com/j2ee/javaserverfaces/1.1/docs/tlddocs/
- http://horstmann.com/corejsf/faces-config.html
- **MyFaces Reference**
  – http://myfaces.apache.org/
- User's Guide, extensions documentation, FAQs, ...
- **Knjige**

Razvoj JSF aplikacije

- **Start sa početnom aplikacijom koja ima sve potrebne ulaze**
  - Potrebno je
    - JAR files
    - web.xml ulazi
    - config fajlovi
    - properties fajlovi
  - I MyFaces i Sun RI nemaju svoj ekvivalent Struts struts-blank aplikaciji
- **Koristiti ant**
  - MyFaces source distribution koji sadrži ugrađene skriptove
    - Nije uključen u osnovni MyFaces download
  - Sun RI sadrži
    - ***install_dir*/samples/build.properties.sample**
    - **I dalje treba dodati web.xml ulaze**

Dodatne mogućnosti

- **Koristiti primere sa Interneta**
  - Download sa http://www.coreservlets.com/JSF-Tutorial/
  - **jsf-blank-myfaces**
    - Sadrži sve potrebne ulaze
    - Sadrži i JAR fajlove za MyFaces dodatke, portlete,fajl upload, i Tiles
  - **jsf-blank-myfaces-minimal**
    - Sadrži sve potrebne ulaze
    - Ne sadrži dodatke
  - **jsf-blank-Sun-RI**
    - Sadrži sve potrebne ulaze
    - Ne sadrži dodatke

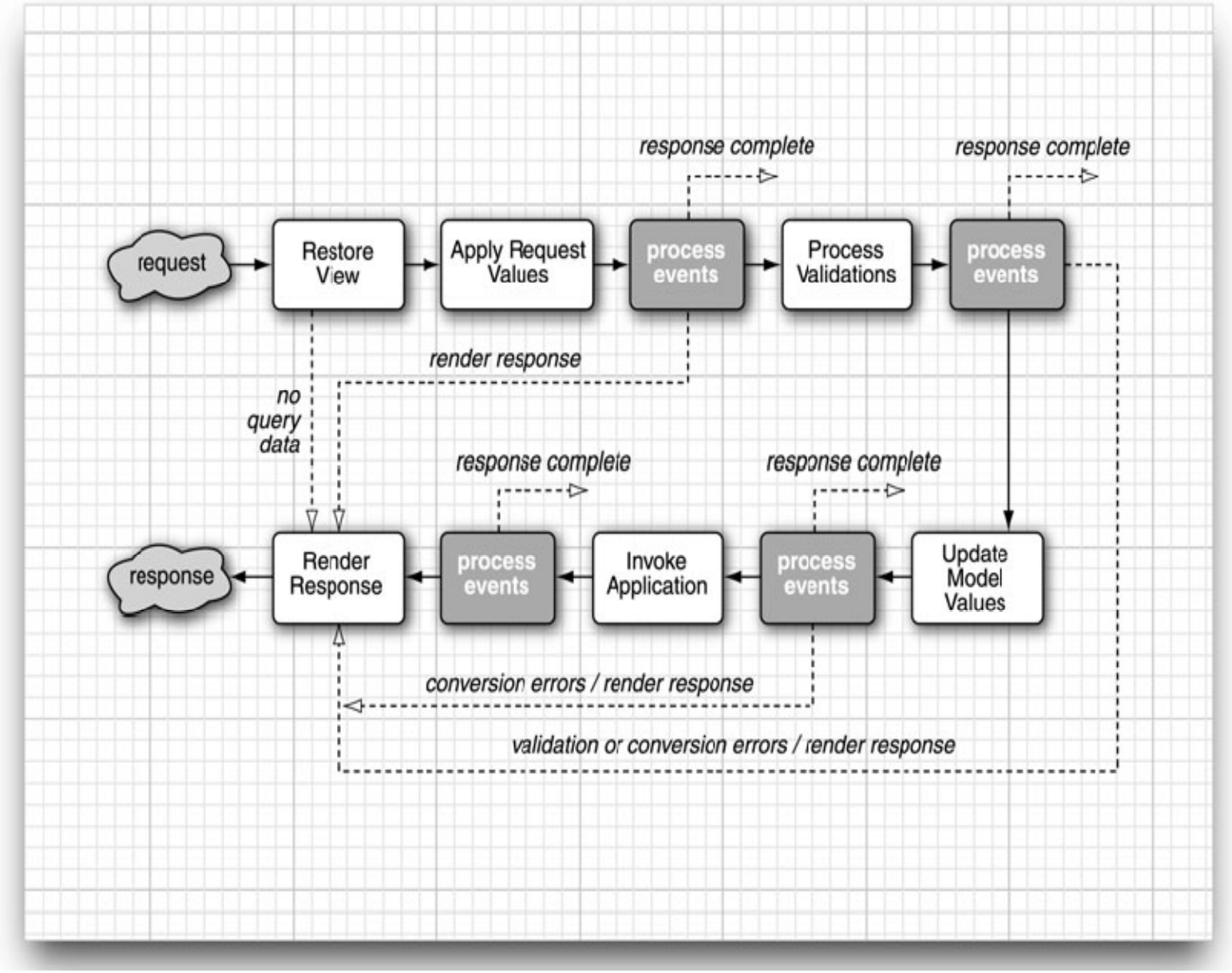
Životni ciklus

- **JSF specifikacija definiše 6 različitih faza**
- **Faza *Restore View* vraća stablo komponenti, za stranicu koja je ranije prikazivana, odnosno pravi novo stablo komponenti ako se stranica prikazuje prvi put. Ako je stranica ranije prikazivana, sve komponente se postavljaju na početno stanje. To znači da JSF automatski postavlja informacije o traženoj formi. Na primer kada korisnik unese i pošalje određene nelegalne podatke, koji se neće propustiti na dalju obradu tokom njihove provere, ulazni podaci se korisniku ponovo prikazuju, tako da korisnik može da ih ispravi. Ako zahtev ne sadrži ulazne podatke, JSF implementacija prelazi na *Render Response* fazu. Ovo se događa kada se stranica prikazuje prvi put.**
- **Sledeća faza je *Apply Request Values.* U okviru ove faze JSF implementacija prolazi kroz objekte komponenti u okviru stabla komponenti. Za svaki objekat komponenti se proverava koja mu vrednost pripada i ona mu se dodeljuje. U ovoj fazi se dodaju događaji vezani za dugme ili link u red događaja.**

Životni ciklus

- U okviru *Process Validation* poslati stringovi se prvo prebacuju u „lokalne" vrednosti, koji mogu biti objekti bilo kog tipa. Kada se diznira JSF stranica, mogu se za klijentsku stranicu vezati validatori koji izvršavaju proveru korektnosti lokalnih vrednosti. Ako je validacija uspešna, JSF ciklus se nastavlja normalno. Kada se desi greška JSF implementacija poziva *Render Response* fazu direktno, prikazujući ponovo traženu stranicu, tako da se korisniku omogući korektan unos. Može se prikazati i odgoavarjuća poruka, koja korisniku objašnjava zašto ponovo vidi iste podatke.
- Kada se izvrše konverzije i validacije, pretpostavlja se da je bezbedno promeniti model podataka. Tokom *Update Model Values* faze, lokalne vrednosti se koriste da bi se promenile vrednosti u Bean-ovima povezanim sa komponentama.
- U okviru *Invoke Application* faze izvršava se metod action() dugmeta ili linka koji je doveo do slanja forme. U ovom metodu se mogu izvršiti dodatne aplikacione obrade. Takođe, tu se generiše izlazni string, koji se šalje delovima odgovornim za navigaciju, gde se izvršava poziv nove stranice.
- Na kraju, *Render Response* faza dekodira odgovor i šalje ga klijentu. Kada korisnik sa nove stranice pošalje novu formu, klikne na link ili na drugi način generiše novi zahtev, startuje se novi ciklus

Životni ciklus

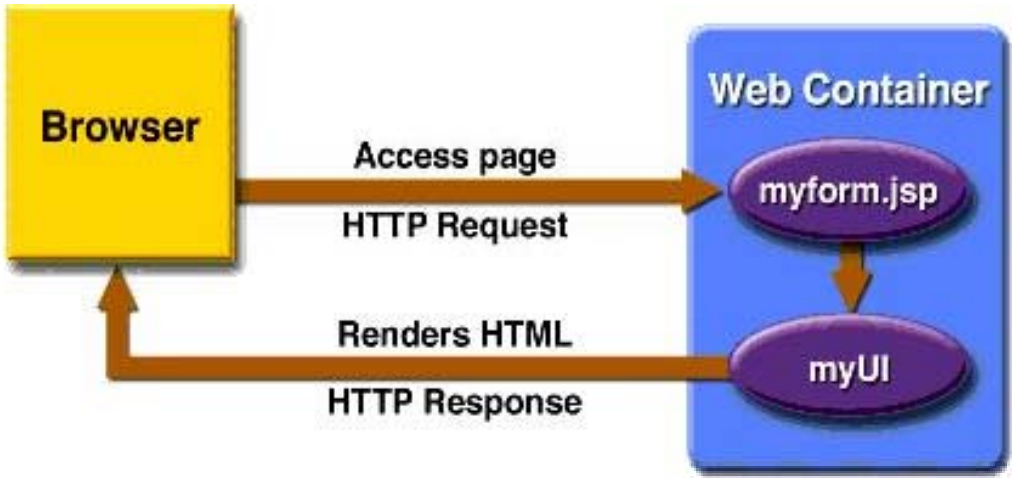


Životni ciklus

- **Za realizaciju web aplikacija JSF pojednostavljuje razvojni proces obezbeđujući komponentnocentričan pristup razvoju java web user intreface-a. Glavne komponente JSF-a su:**
- **API koji omogućava predstavljanje UI komponenti i manipulisanje njihovim stanjima, upravljanje događajima, validacije na strani servera, konverziju podataka, definisanje navigacije između stranica, internacionalizaciju.**
- **Dve JSP tag biblioteke za predstavljanje UI komponenti u okviru JSP strane i za povezivanje komponenti sa serverskim objektima.**

Životni ciklus

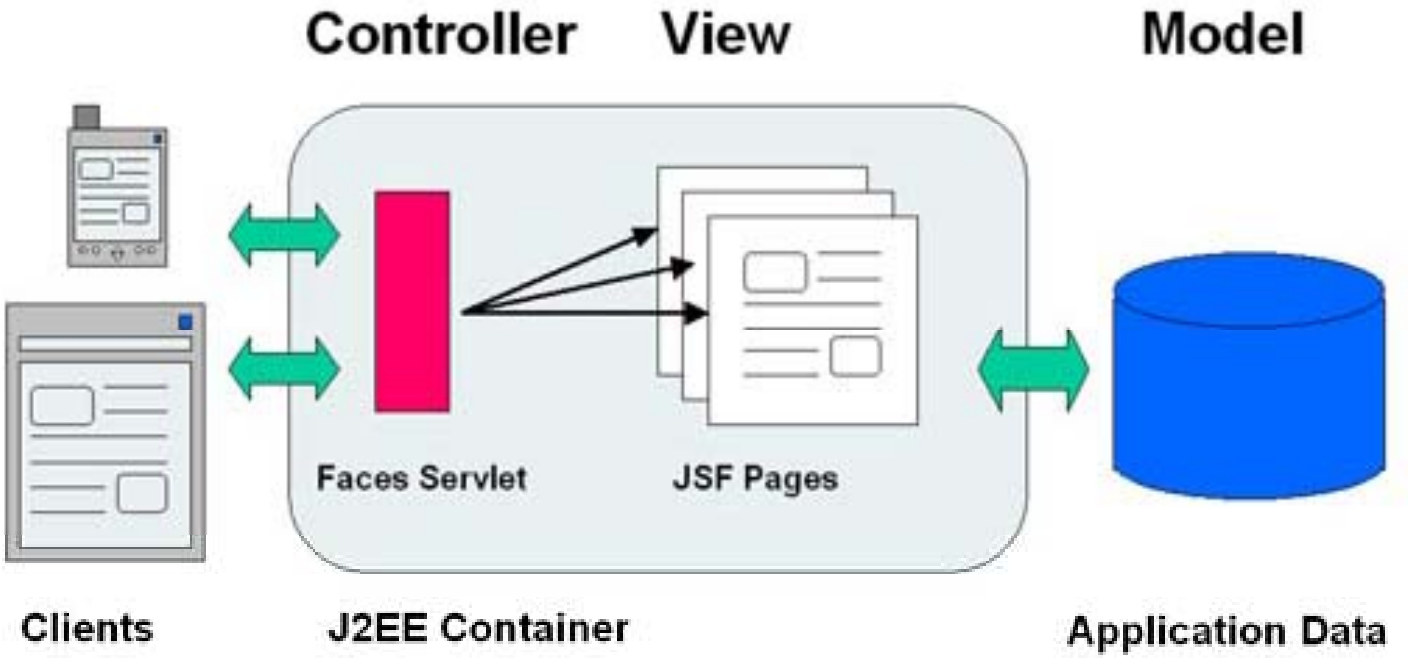
- **Korisnički interfejs koji je kreiran pomoću JSF tehnologije (*myUI*) se izvršava na serveru i renderuje nazad klijentu.**



- ***myform.jsp* je JSF stranica - JSP stranica koja u sebi sadrži JSF tagove. Ona prikazuje UI komponente koristeći tagove definisane JavaServer Faces tehnologijom. UI za web aplikaciju (*myUI*) manipuliše objektima koje koristi JSP strana. Ovi objekti uključuju:**
  - UI component objekte koji se mapiraju na tag-ove na JSP strani
  - Event lintener-e, validator-e, i converter-e koji su registrovani na komponente
  - Objekte koji enkapsuliraju podatke i funkcionalnosti komponente specifične za aplikaciju

Životni ciklus

- **JSF framework prati Model-View-Controller (MVC) pattern što JSF aplikacije čini dosta prilagodljivijm i pogodnijim za rad - user-interface kod (*View*) je razdvojen od aplikacionih podataka i logike (*Model*).**
- **JSF servlet (*Controller*) upravlja svom korisničkom interakcijom sa aplikacijom. On priprema JSF kontekst koji omogućava stranicama pristup aplikacionim podacima kao i zaštitu od neovlašćenog i neadekvatnog pristupa stranicama.**



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>

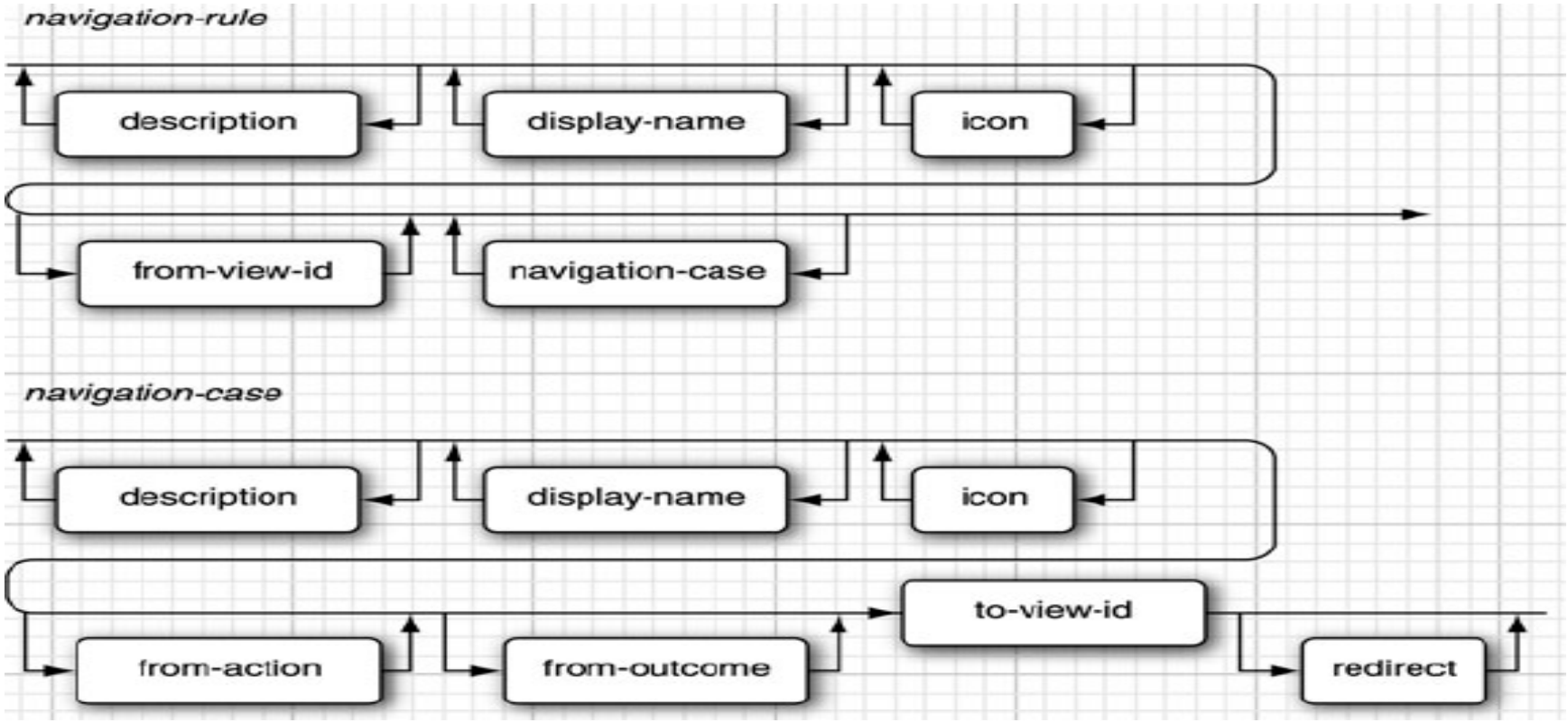
</navigation-case>

</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 –redirekcija

- Ako se unese redirect element posle to-view-id elementa, JSF prekida trenutni zahtev i šalje HTTP redirect klijentu, što govori klijentu koji URL da koristi za sledeću stranicu.
- Preusmerenje stranice je sporije u odnosu na prosleđivanje jer se aktivira još jedan ciklus sa browser-om, ali dopušta browser-u da izmeni vrednost svog adresnog polja.

<navigation-case>

<from-outcome>success</from-outcome>

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

<redirect/>

</navigation-case>

- Bez redirekcije, orginalni URL (localhost:8080/javaquiz/index.faces) ostaje nepromenjen kada se sa /index.jsp pređe na stranicu /success.jsp.
- Sa redirekcijom, browser prikazuje novi URL (localhost:8080/javaquiz/success.faces).

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

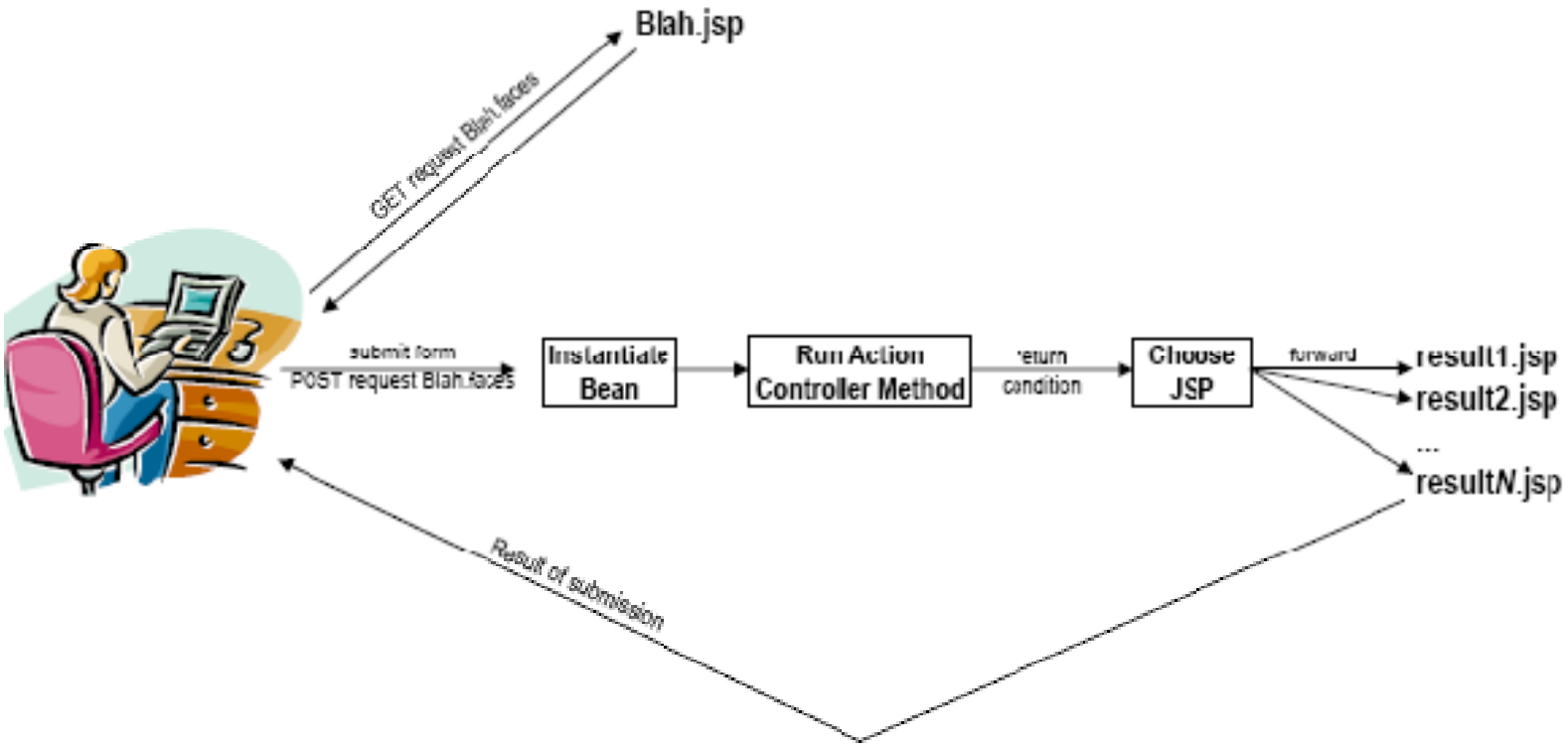
- Posle implementacije bean-ova koji će se koristiti u aplikaciji potrebno je konfigurisati ih u faces-config.xml konfiguracionom fajlu.
- To je potrebno da bi JSF mogao da automatski kreira nove instance bean-ova kad god se ukaže potreba:

```
<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-property>
    <property-name>personName</property-name>
    <property-class>java.lang.String</property-class>
    <value>Bosko Nikolic</value>
  </managed-property>
</managed-bean>
```

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 (pojednostavljena)



JSF kontrola toka (pojednostavljena)

- **Forma se prikaže**
  – Forma koristi f:view i h:form
- **Izvrši se submit forme**
  – Originanil URL i ACTION URL su http://…/*blah*.faces
- **Napravi se instanca bean-a**
  – Navedeni u managed-beans delu faces-config.xml
- **Poziva se action controller metod**
  – Naveden kao action atribut h:commandButton
- **action metod vraća uslov**
  – String koji se poklapa sa from-outcome u okviru navigacionih pravila u faces-config.xml
- **Rezultujuća stranica se prikazuje**
  – Stranica je specificirana sa to-view-id u okviru navigacionih pravila u faces-config.xml

Osnovni koraci upotrebe JSF

1. **Kreirati bean sa propertie-ijim 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

- **Kopirati jsf-blank**
  - Preinemovati u jsf-test
  - Identičan rad sa Apache MyFaces i Sun RI
- **Originalni URL:**
  - http://*hostname*/jsf-test/register1.faces
- **Kada se forma submituje:**
  - Statička stranica (result1.jsp) se prikazuje
- **Statički rezultat:**
  - Nema poslovne logike, bean-ova, ili Java koda
- **Glavni ciljevi**
  - Format originalne forme
  - Upotreba faces-config.xml fajla

Ciljevi primera

- **Početna forma je u sledećem formatu:**

<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
HTML markup
<h:form>
HTML markup and h:blah tags
</h:form>
HTML markup
</f:view>

- **Deklarišu se mapiranja u faces-config.xml**

<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE faces-config PUBLIC …>
<faces-config>
<navigation-rule>
<from-view-id>/*blah*.jsp</from-view-id>
<navigation-case>
<from-outcome>*some string*</from-outcome>
<to-view-id>/WEB-INF/results/*something*.jsp</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>

Korak 2: kreiranje početne forme

- **Osnovni format**

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

- **Poziv stranice**
  - Fajl je *blah*.jsp
  - URL je *blah*.faces

Korak 2: kreiranje početne forme

- **h:form element**
  - ACTION je automatski isti fajl (trenutni URL)
  - METHOD je automatski POST
- **Elementi unutar h:form**
  - Koristiti specijalne tagove da bi se prikazali ulazni elementi
    - h:inputText odgovara <INPUT TYPE="TEXT">
    - h:inputSecret odgovara <INPUT TYPE="PASSWORD">
    - h:commandButton odgovara <INPUT TYPE="SUBMIT">
  - Kasnije, ulazni elementi će se povezivati sa propertie-ijima bean
  - Za statičku navigaciju, specificira se jednostavan string kao akcija za h:commandButton
    - String mora da odgovara pravilima navigacije iz faces-config.xml
- **Više informacija o h:*blah* elementima**
  - http://java.sun.com/j2ee/javaserverfaces/1.1/docs/tlddocs/

Korak 2: primer koda

```
<%@ 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>...</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">New Account Registration</TH></TR>
</TABLE>
<P>
<h:form>
Email address: <h:inputText/><BR>
Password: <h:inputSecret/><BR>
<h:commandButton value="Sign Me Up!" action="register"/>
</h:form>
</CENTER></BODY></HTML>
</f:view>
```

Korak 3: specificirati Action Controller

- **Koristiti action atribut h:commandButton**
  - U većini realnih slučajeva, specificira se metod koji se poziva ka da se forma prosledi
  - Metod vraća različite string-ove, i faces-config.xml mapira stringove u izlazne stranice
- **Za statičku navigaciju, specificira se jednostavan string**
  - faces-config.xml mapira string u izlaznu straniceu
- **Primer**

```
<h:form>
Email address: <h:inputText/><BR>
Password: <h:inputSecret/><BR>
<h:commandButton value="Sign Me Up!" action="register"/>
</h:form>
```

Koraci 2 i 3: rezultat

- **Fajl je *tomcat_dir*/webapps/jsf-test/register1.jsp**
- **URL he http://localhost/jsf-test/register1.faces**

Korak 5: Promeniti faces-config.xml

- **Osnovni format**

<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE faces-config PUBLIC …>
<faces-config>
…
</faces-config>

- **Specificirati pravila navigacije**

…
<faces-config>
<navigation-rule>
<from-view-id>/*the-input-form*.jsp</from-view-id>
<navigation-case>
<from-outcome>*string-from-action*</from-outcome>
<to-view-id>/WEB-INF/…/*something*.jsp</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>

Korak 5: Promeniti faces-config.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE faces-config PUBLIC ...>
<faces-config>
<navigation-rule>
<from-view-id>/register1.jsp</from-view-id>
<navigation-case>
<from-outcome>register</from-outcome>
<to-view-id>/WEB-INF/results/result1.jsp</to-view-
   id>
</navigation-case>
</navigation-rule>
</faces-config>
```

Korak 6: Kreirati izlaznu JSP stranu

- **Koristi se RequestDispatcher.forward**
  – Stranica može/treba da bude u WEB-INF
- **Primer koda:**
  – …/jsf-test/WEB-INF/results/result1.jsp

```
<!DOCTYPE ...>
<HTML>
<HEAD>...</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">Success</TH></TR>
</TABLE>
<H2>You have registered successfully.<BR>
(Version 1)</H2>
</CENTER>
</BODY></HTML>
```

Rezultat


Success - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/jsf-test/register1.faces
Go
Success
You have registered successfully.
(Version 1)
Done
Loca intranet

Korak 7: Zaštititi JSP strane

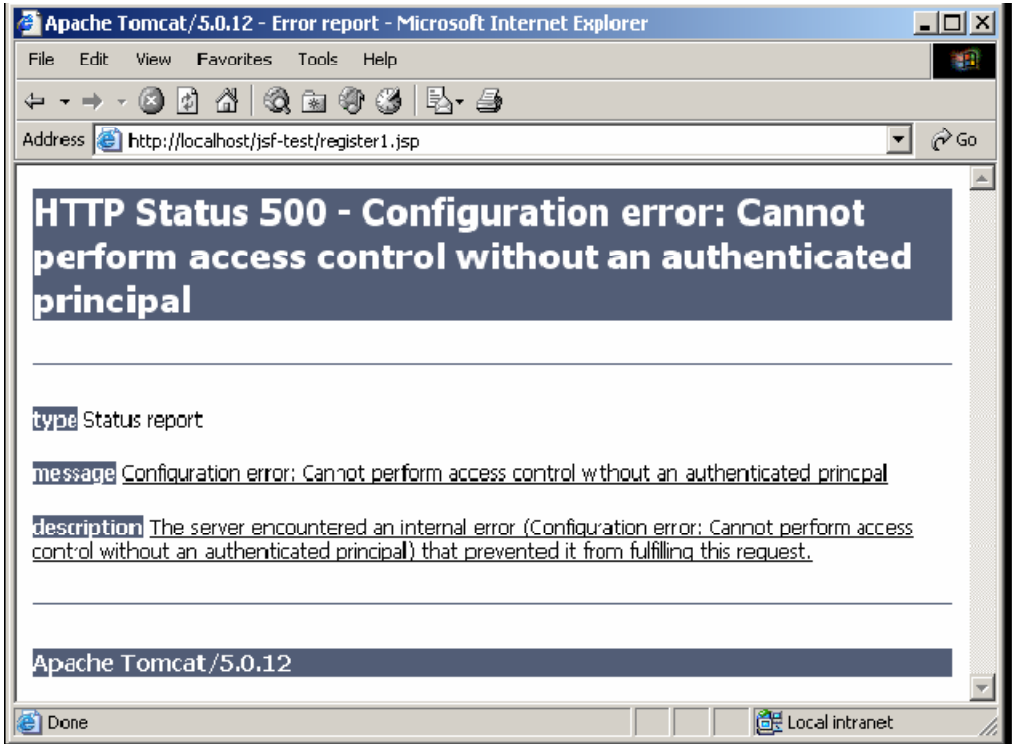
- **Veza imena fajlova i URL**
  - Trenutno fajlovi u formi *blah*.jsp
  - URL koji se koriste su u formi *blah*.faces
  - Treba zabraniti klijentima da direktno pristupaju JSP stranicama
- **Strategije**
  - Ne mogu se smestiti JSP stranice u WEB-INF
    - Zato što URL mora da direktno odgovara lokaciji fajla
  - Koristi se filter ili security-constraint u web.xml. Ali:
    - Nezgrapno
    - Lako se zaboravi
    - Rezultuje u konfuznum porukama o greškama
- **Jedan od najvećih nedostataka JSF dizajna**

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>/welcome.jsp</url-pattern>
<url-pattern>/register1.jsp</url-pattern>
…
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Rezultat

- **Tomcat rezultuje u zbunjujućim porukama o greškama**
- **Ne može se jednostavno koristiti error-page i error-code za bolji prikaz**
  - Status kod 500 se koristi i za druge Tomcat greške



Primer 2: Dinamička navigacija

- **Originalni URL:**
  - http://*hostname*/jsf-test/signup.faces
  - Prikupljaju se informacije o korisniku
- **Kada se forma pošalje, jedan od dva rezultat se prikazuje**
  - Korisnik je prihvaćen
  - Korisnik se odbija
- **Glavni ciljevi**
  - Specificirati action controller
  - Kreirati action controller
  - Koristiti faces-config.xml za
    - Deklaraciju controller-a
    - Mapirati rezultat u izlazne stranice

Glavni ciljevi

- **Specificirati controller sa #{*controllerName.methodName*}**

```
<h:commandButton value="Sign Me Up!“
    action="#{healthPlanController.register}"/>
```

- **Metod controller-a vraća stringove koji odgovaraju uslovima**
  - Ako se vrati null, forma se ponovo prikazuje
  - Za razliku od Struts-a, controller ne zahteva da nasledi neku posebnu klasu
- **Pomoću faces-config.xml deklarisati controller**

```
<faces-config>
<managed-bean>
<managed-bean-name>controller name</managed-bean-name>
<managed-bean-class>controller class</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
</faces-config>
```

- **Dodati više ulaza navigation-rule entries u faces-config.xml**
  - Jedan za svaki mogući string koji vraća controller
  - Ako se ne pronađe odgovarajuće poklapanje, form se ponovo prikazuje

Korak 2: kreiranje početne forme

- **Ista sintaksa kao u Primeru 1**
  - Izuzetak akcija kod commandButton

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
…
<h:form>
First name: <h:inputText/><BR>
Last name: <h:inputText/><BR>
…
<h:commandButton
value="Sign Me Up!"
action="#{healthPlanController.signup}"/>
</h:form>…
</f:view>
```

Korak 3: Specificirati akciju

- **Koristiti atribut action od h:commandButton**
  - Specificirati #{controllerName.methodName}
  - Dizajnirani metod ne zahteva argumente i vraća kao razultat string. faces-config.xml mapira stringove u izlazne stranice
  - Bean koji sadrži metod controller-a mora da bude deklarisan sa ulazom managed-beans entry u faces-config.xml
- **Primer**

```
<h:form>
First name: <h:inputText/><BR>
Last name: <h:inputText/><BR>...
<h:commandButton
value="Sign Me Up!"
action="#{healthPlanController.signup}"/>
</h:form>
```

Korak 3: Specificirati akciju

- **Fajl je *tomcat_dir*/webapps/jsf-test/signup.jsp**
- **URL je http://localhost/jsf-test/signup.faces**

Korak 4: Kreirati akciju

- **Ne treba nasleđivati nijednu posebnu klasu**
  – Razlika od Struts-a
- **Dizajnirani metod ne prihvata argumente i vraća String kao rezultat**
  – Svaki string mora da odgovara sa pravilom navigacije u faces-config.xml
  – Rezultat null dovodi do ponovnog prikaza iste forme
- **Većina controller-a pristupa parametrima request-a preko bean-ova**
  – O tome kasnije
  – Za sada, poslovna logika će biti nezavisna od ulaznih podataka

Korak 4: Kreirati akciju

```
package coreservlets;
public class HealthPlanController {
public String signup() {
if (Math.random() < 0.2) {
return("accepted");
} else {
return("rejected");
}
}
}
```

Korak 5: Promeniti faces-config.xml

- **Declarisati action controller**

```
...
<faces-config>
<managed-bean>
<managed-bean-name>
healthPlanController
</managed-bean-name>
<managed-bean-class>
coreservlets.HealthPlanController
</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
...
</faces-config>
```

Korak 5: Promeniti faces-config.xml

- **Declarisati action controller**

```
...
<faces-config>
<managed-bean>
<managed-bean-name>
healthPlanController
</managed-bean-name>
<managed-bean-class>
coreservlets.HealthPlanController
</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
...
</faces-config>
```

Korak 5: Promeniti faces-config.xml

- **Specificirati pravila navigacije**
  – Moraju se poklapati sa vrednostima koje šalje controller

```
…
<faces-config>
…
<navigation-rule>
<from-view-id>/signup.jsp</from-view-id>
<navigation-case>
<from-outcome>accepted</from-outcome>
<to-view-id>/WEB-INF/results/accepted.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>rejected</from-outcome>
<to-view-id>/WEB-INF/results/rejected.jsp</to-view-id>
</navigation-case>
</navigation-rule>
…
</faces-config>
```

Korak 6: Kreirati izlaznu JSP stranicu

- **…/jsf-test/WEB-INF/results/accepted.jsp**

```
<!DOCTYPE …>
<HTML>
<HEAD>…</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">Accepted!</TH></TR>
</TABLE>
<H2>You are accepted into our health plan.</H2>
Congratulations.
</CENTER>
</BODY></HTML>
```

Korak 6: Kreirati izlaznu JSP stranicu

- **…/jsf-test/WEB-INF/results/rejected.jsp**

```
<!DOCTYPE …>
<HTML>
<HEAD>…</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">Rejected!</TH></TR>
</TABLE>
<H2>You are rejected from our health plan.</H2>
Get lost.
</CENTER>
</BODY></HTML>
```

Rezultat


Rejected! - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/jsf-test/signup.faces
Rejected!
You are rejected from our health plan.
Get lost.
Done

Korak 7: Zaštiti JSP stranice

```
...
<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>/welcome.jsp</url-pattern>
<url-pattern>/register1.jsp</url-pattern>
<url-pattern>/signup.jsp</url-pattern>
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Primer: Čitanje podataka zahteva

- **Originalni URL:**
  – http://*hostname*/jsf-test/register2.faces
- **Kada se forma pošalje, dobije se jedan od tri rezultata**
  – Poruka o nelegalnoj email adresi
  – Poruka o nelegalnoj šifri
  – Uspešno
- **Glavni zadaci**
  – Čitanje podataka zahteva u okviru action controller-a
  • Kasnije, jednostavniji pristup koji se primenjuje u praksi. Ipak, na ovaj način se može pristupiti HttpServletRequest objektu da bi se pročitali cookie-iji, headeri, ...

Glavni zadaci

- **Većina konkretnih JSF aplikacija koristi form bean-ove**
  - Kasnije
- **U ovom primeru se čitanje obavlja ručno**
  - Zahteva određeni deo Java koda (ExternalContext)
  - ExternalContext se koristi i u konkretnim JSF stranicama
  - Za čitanje request header-a
  - Za čitanje cookie-ija
  - Za postavljanje response header-a ili status kodova
  - Za pristup promenljivama koje nisu povezane sa form bean-om

Korak 1: Kreirati bean propertie-ije

- **Razmatra se u sledećim lekcijama**

  – U ovom primeru podaci sa forme se čitaju na stari način: direktnim pozivom request.getParameter.

  – Da bi to moglo da bude moguće, treba pregledati ExternalContext, koji omogućava pristup standardnim servlet i JSP objektima (request, response, ServletContext, session, ...)

ExternalContext context =

FacesContext.getCurrentInstance().getExternalContext();

HttpServletRequest request =

(HttpServletRequest)context.getRequest();

String param1 = request.getParameter("name1");

–ExternalContext ima i metode za pristup parametrima i drugim podacima iz heš tabele

• Request data

– getRequestParameterMap, getRequestCookieMap, getRequestHeaderMap, ...

• Promenljive

– getRequestMap, getSessionMap, getApplicationMap

Korak 2: Kreirati početnu formu

- **Isto kao u prethodnom primeru, osim**
  - h:form ima id atribut
  - h:inputText i h:inputSecret takođe imaju id
  - Trenutni request parametar je *formID*:*elementID*
    - Potrebno je samo u slučaju kada se ručno čitaju parametri zahteva!
- **Primer**

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
…
<h:form id="form2">
Email address: <h:inputText id="email"/><BR>
Password: <h:inputSecret id="password"/><BR>
<h:commandButton value="Sign Me Up!"
action="#{registrationController.register}"/>
</h:form>…
</f:view>
```

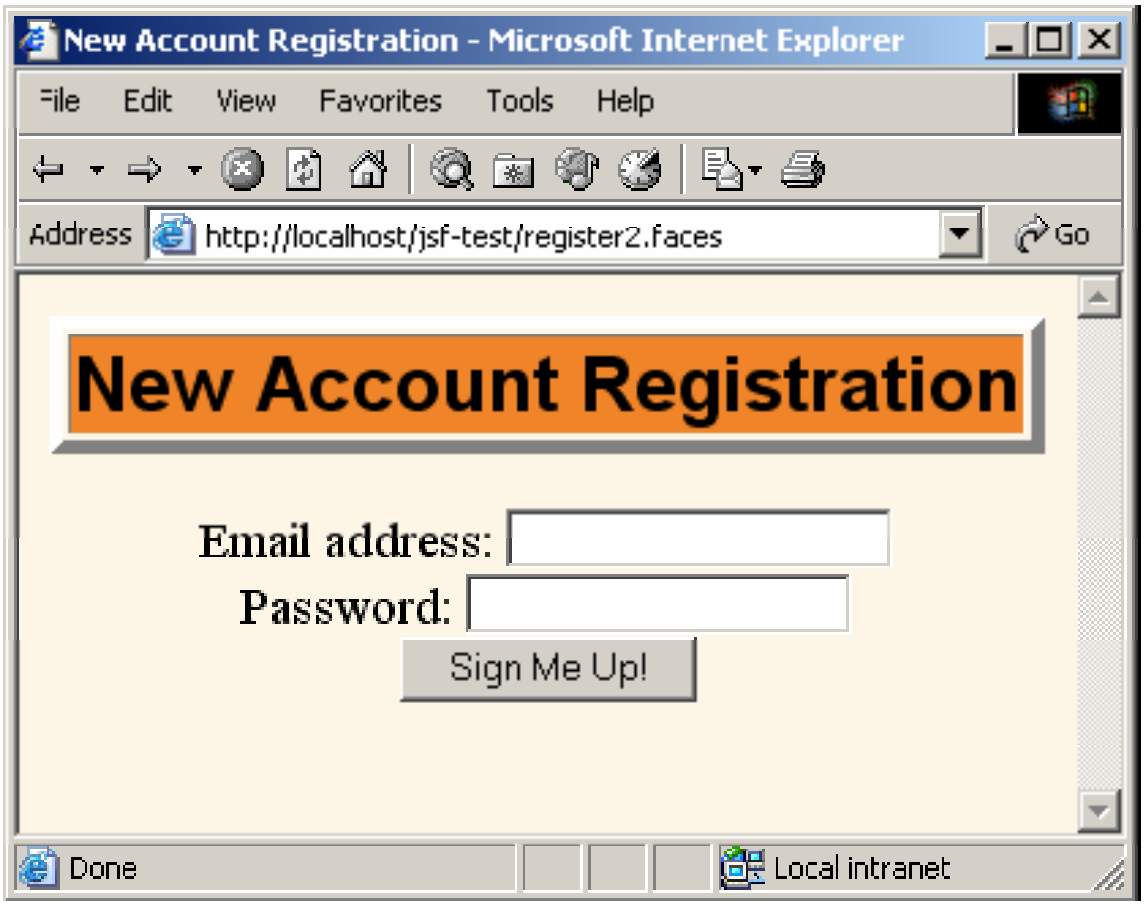
Korak 3: Specificirati akciju

- **Koristiti action atribut od h:commandButton**
  - Specificirati #{controllerName.methodName}
  - Isto kao u prošlom primeru
- Primer

```
<h:form id="form2">
Email address: <h:inputText id="email"/><BR>
Password: <h:inputSecret id="password"/><BR>
<h:commandButton value="Sign Me Up!"
action="#{registrationController.register}"/>
</h:form>
```

Korak 3: Specificirati akciju

- File je *tomcat_dir*/webapps/jsf-test/register2.jsp
- URL je http://localhost/jsf-test/register2.faces



Korak 4: Kreirati akciju

- **Ne treba nasleđivati nijednu posebnu klasu**
  – Razlika od Struts-a
- **Dizajnirani metod ne prihvata argumente i vraća String kao rezultat**
  – Svaki string mora da odgovara sa pravilom navigacije u faces-config.xml
  – Rezultat null dovodi do ponovnog prikaza iste forme
- **Većina controller-a preistupa parametrima request-a preko bean-ova**
  – O tome kasnije
  – Koristi se pristup preko ExternalContext
  – ExternalContext se može koristiti za pristup
  • Cookies, podacima sesije, request headers, ...

Korak 4: Kreirati akciju

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.faces.context.*;
public class RegistrationController {
public String register() {
ExternalContext context =
    FacesContext.getCurrentInstance().getExternalContext();
HttpServletRequest request = (HttpServletRequest)context.getRequest();
String email = request.getParameter("form2:email");
String password = request.getParameter("form2:password");
if ((email == null) || (email.trim().length() < 3) ||
                                (email.indexOf("@") == -1)) {
          return("bad-address");
    } else if ((password == null) || (password.trim().length() < 6)) {
return("bad-password");
} else { return("success"); } } }
```

Korak 5: Promeniti faces-config.xml

- **Delarisati action controller**

```
...
<faces-config>
<managed-bean>
<managed-bean-name>
registrationController
</managed-bean-name>
<managed-bean-class>
coreservlets.RegistrationController
</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
...
</faces-config>
```

Korak 5: Promeniti faces-config.xml

- **Specifirati pravila navigacije**

```
...
<faces-config>
...
<navigation-rule>
<from-view-id>/register2.jsp</from-view-id>
<navigation-case>
<from-outcome>bad-address</from-outcome>
<to-view-id>/WEB-INF/results/bad-address2.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>bad-password</from-outcome>
<to-view-id>/WEB-INF/results/bad-password2.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>success</from-outcome>
<to-view-id>/WEB-INF/results/result2.jsp</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>
```

Korak 6: Kreirati izlazne JSP stranice

- **…/jsf-test/WEB-INF/results/bad-address2.jsp**

```
<!DOCTYPE …>
<HTML>
<HEAD>…</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">Illegal Email Address</TH></TR>
</TABLE>
<P>
Address must be of the form username@host.
Please <A HREF="register2.faces">try again</A>.
</CENTER>
</BODY></HTML>
```

Rezultat


Illegal Email Address - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/jsf-test/register2.faces
Go
Illegal Email Address
Address must be of the form username@host.
Please try again.
Done
Local intranet

Korak 6: Kreirati izlazne JSP stranice

- **…/jsf-test/WEB-INF/results/badpassword2.jsp**

```
<!DOCTYPE …>
<HTML>
<HEAD>…</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">Illegal Password</TH></TR>
</TABLE>
<P>
Password must contain at least six characters.
Please <A HREF="register2.faces">try again</A>.
</CENTER>
</BODY></HTML>
```

Korak 6: Kreirati izlazne JSP stranice

Korak 6: Kreirati izlazne JSP stranice

- **…/jsf-test/WEB-INF/results/result2.jsp**

```
<!DOCTYPE …>
<HTML>
<HEAD>…</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">Success</TH></TR>
</TABLE>
<H2>You have registered successfully.<BR>
(Version 2)</H2>
</CENTER>
</BODY></HTML>
```

Korak 6: Kreirati izlazne JSP stranice

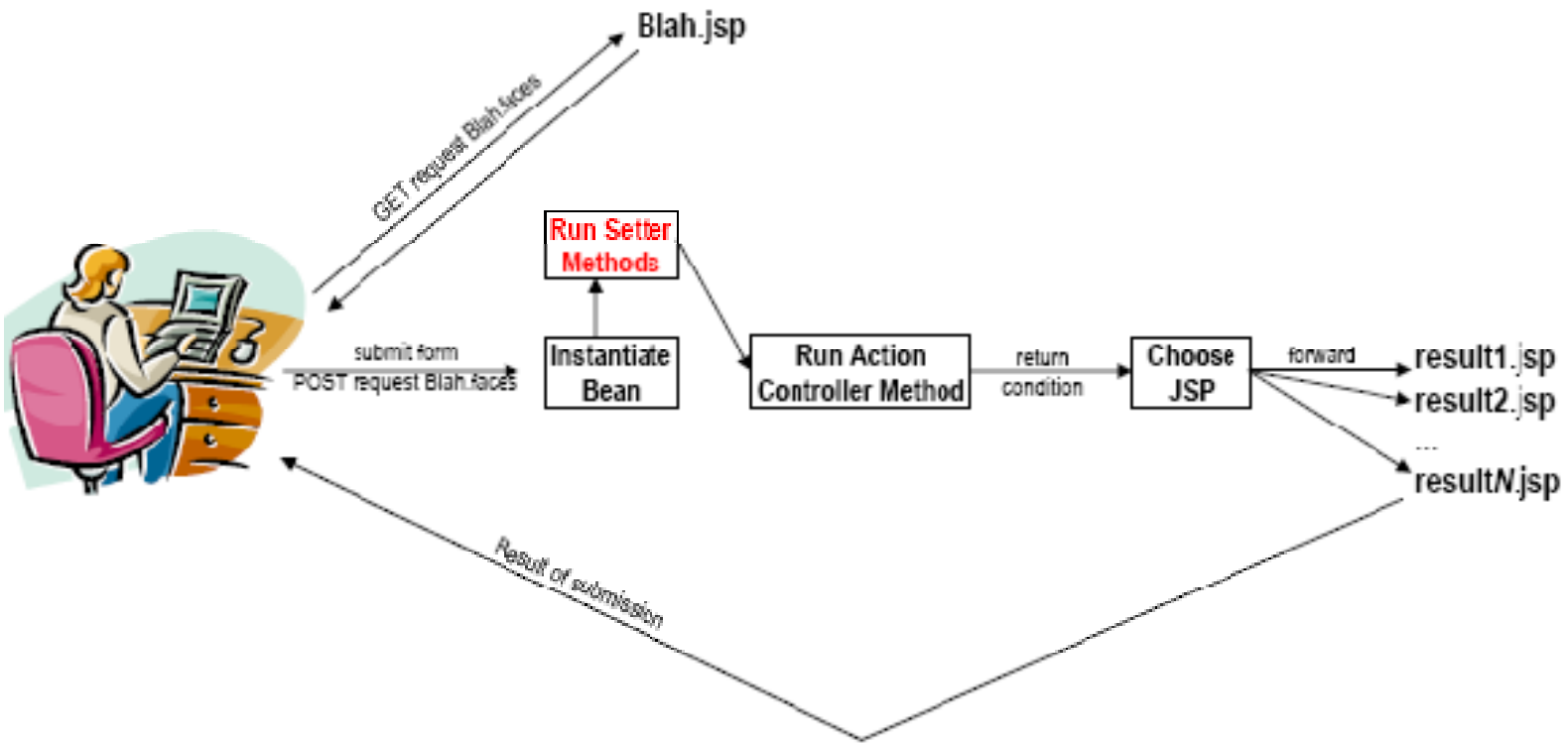
Korak 7: Zaštiti JSP stranice

```
<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>/welcome.jsp</url-pattern>
<url-pattern>/register1.jsp</url-pattern>
<url-pattern>/signup.jsp</url-pattern>
<url-pattern>/register2.jsp</url-pattern>
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Koristiti bean-ove

- **Upotrebiti bean-ove za prikazivanje parametara zahteva**
- **Deklarisat bean-ove u faces-config.xml**
- **Prikaz propertie-ija bean-a**
  - Standardni JSF pristup
  - JSP 2.0 expression jezik
- **Koristiti properties fajlove**
  - Standardni upiti/stringovi
  - Internacionalizacija sadržaja

JSF kontrola toka



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>
```

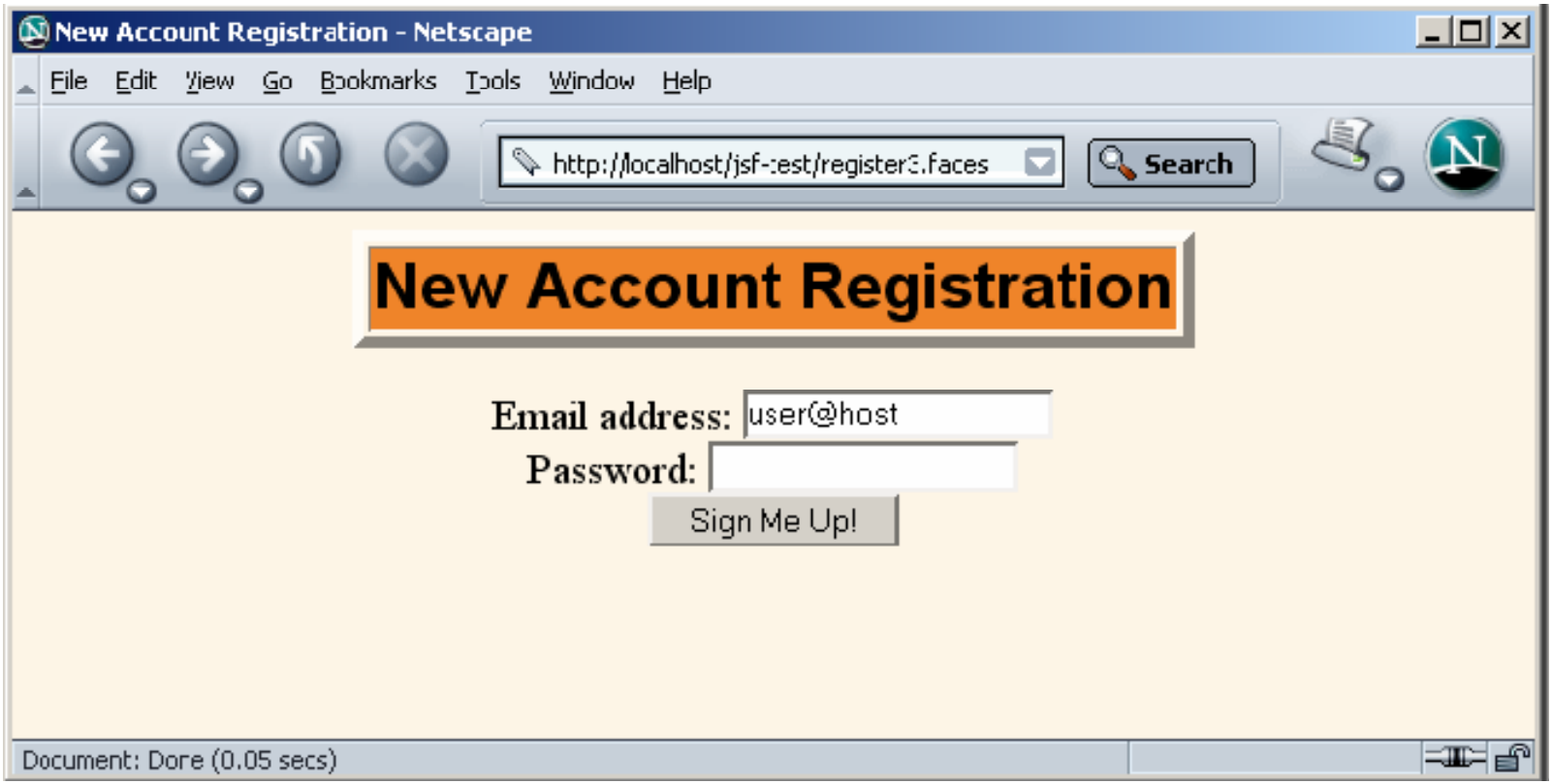
Korak 3: definisati akciju

- **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)

Korak 6: Kreiranje izlaznih JSP strana

- .../jsf-test/WEB-INF/results/result3.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">Success</TH></TR>
</TABLE>
<H2>You have registered successfully.</H2>
<UL>
<LI>Email Address:
<h:outputText value="#{registrationBean.email}"/>
<LI>Password:
<h:outputText value="#{registrationBean.password}"/>
</UL>
(Version 3)
...
</HTML>
</f:view>
```

Rezultat


Success - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost/jsf-test/register3.faces
Search
Success
You have registered successfully.
• Email Address: palmisano@ibm.com
• Password: jhugrad
(Version 3)
Document: Done (0.1 secs)

Korak 7: Zaštita JSP strana

```
<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>/welcome.jsp</url-pattern>
<url-pattern>/register1.jsp</url-pattern>
<url-pattern>/register2.jsp</url-pattern>
<url-pattern>/register3.jsp</url-pattern>
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Moguće alternative

- **Koristiti JSP 2.0 EL**

  – Ako izlazne stranice samo prikazuju property-ije bean-a (ne obavljaju operacije sa formom ili koriste elemente forme):

  • Zašto i f:view i povezana taglib deklaracija?

  • Zašto koristiti h:outputText i odgovarajuću taglib deklaraciju, kada je upotreba JSP 2.0 EL jednostavnija?

  – Ako se koristi JSP 2.0 EL, mora se koristiti:

  • JSP 2.0 okruženje (na primer, Tomcat 5, a ne Tomcat 4)

  • Koristiti JSP 2.0 deklaracija u okviru web.xml (kasnije)

Upotreba JSP 2.0 EL

- **Standardni JSF pristup**

```
<%@ 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">Success</TH></TR>
</TABLE>
<H2>You have registered successfully.</H2>
<UL>
<LI>Email Address:
<h:outputText value="#{registrationBean.email}"/>
<LI>Password:
<h:outputText value="#{registrationBean.password}"/>
</UL>
(Version 3)
...
</HTML>
</f:view>
```

Upotreba JSP 2.0 EL

- **JSP 2.0 pristup**
  - Izbegava se taglib deklaracija i f:view tagovi
  - Kraći izrazi koji prikazuju property-ije bean-a

```
<!DOCTYPE ...>
<HTML>
...
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">Success</TH></TR>
</TABLE>
<H2>You have registered successfully.</H2>
<UL>
<LI>Email Address: ${registrationBean.email}
<LI>Password: ${registrationBean.password}
</UL>
(Version 3)
...
</HTML>
```

Prednosti upotrebe JSP 2.0 EL

- **Skraćena notacija za property-ije bean-a.**

  – Da bi se pristupilo property-iju companyName (rezultat metoda getCompanyName) odgovarajuće promenljive company (objekat smešten u okviru zahteva, sesije ili aplikacije) dovoljno je koristiti #{company.companyName}. Da bi se prisupilo property-iju firstName, koji je definisan u okviru property-ija president promenljive company, koristi se #{company.president.firstName}.
- **Jednostavan pristup elementima kolekcije.**

  – Da bi se pristupilo elementu niza, liste ili mape, koristi se #{variable[indexOrKey]}.

Prednosti upotrebe JSP 2.0 EL

- **Jednostavan pristup parametrima zahteva, cooky-ijima, i drugim podacima zahteva.**

  – Za pristup standardnim tipovima podataka zahteva, može se koristiti jedan od nekoliko predefinisanih implicitnih objekata.
- **Mali, ali koristan skup jednostavnih operatora.**

  – ZA obradu objekata pomoću EL izraza, može se koristiti neki od nekoliko aritmetičkih, relacionih, logičkih, ili drugih operatora.
- **Uslovni izlaz.**

  – Za izbor između više izlaznih opcija, ne moraju se koristiti Java scripleti, već #{test ? option1 : option2}.
- **Automatska konverzija tipova.**

  – EL uklanja potrebu za konverziju tipova.
- **Prazna vrednost umesto poruke o grešci.**

  – U većini slučajeva, greška ili NullPointerExceptions daju prazan string, a ne izuzetak

Prednosti upotrebe JSP 2.0 EL

- **JSF EL**
- **Može se koristiti samo u okviru atributa JSF tagova**
- **Zahtevaju taglib deklaraciju**
- **Koriste se na serverima koji podržavaju JSP 1.2+**
  - **–WebLogic 8.1, Tomcat 4, ...**
- **Koriste #{blah}**
- **Mogu da prikažu poslate podatke i izlazne vrednosti**
- **Pristupaju bean-ovima definisanim u okviru zahteva, sesije, aplikacije ili posebno definisanim**

- **JSP 2.0 EL**
- **Mogu se koristiti bilo gde u okviru JSP stranice**
- **Ne zahtevaju taglib deklaraciju**
- **Koriste se na serverima koji podržavaju JSP 2.0**
  - **–WebLogic 9, Tomcat 5, ...**
- **Koriste ${blah}**
- **Prikazuju samo izlazne vrednosti**
- **Pristupaju bean-ovima definisanim u okviru zahteva, sesije ili aplikacije**

Prikaz property-ija bean-a

- **#{varName.propertyName}**
  – Rezultat je pretraga objekata HttpServletRequest, HttpSession, ServletContext, *u datom poretku*, i prikaz odgovarajućeg property-ija bean-a
  – Mora se koristiti kao atribut u okviru JSF taga
- **Ekvivalentne forme**
  – <h:outputText value="#{customer.firstName}"/>
  • Moguće je u okviru svih JSF verzija.
  – ${customer.firstName}
  • Moguće je samo u okviru JSP 2.
  – <%@ page import="coreservlets.NameBean" %>
  <% NameBean person = (NameBean)pageContext.findAttribute("customer"); %>
  <%= person.getFirstName() %>
  • Ne preporučuje se pre-EL verzija.

Primer TestBean

```
package coreservlets;

import java.util.*;

public class TestBean {
  private Date creationTime = new Date();
  private String greeting = "Hello";

  public Date getCreationTime() {
    return(creationTime);
  }

  public String getGreeting() {
    return(greeting);
  }

  public double getRandomNumber() {
    return(Math.random());
  }
}
```

Primer faces-config.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE ...>
<faces-config>
<managed-bean>
<managed-bean-name>testBean</managed-bean-
   name>
<managed-bean-class>
coreservlets.TestBean
</managed-bean-class>
<managed-bean-scope>request</managed-bean-
   scope>
</managed-bean>
...
</faces-config>
```

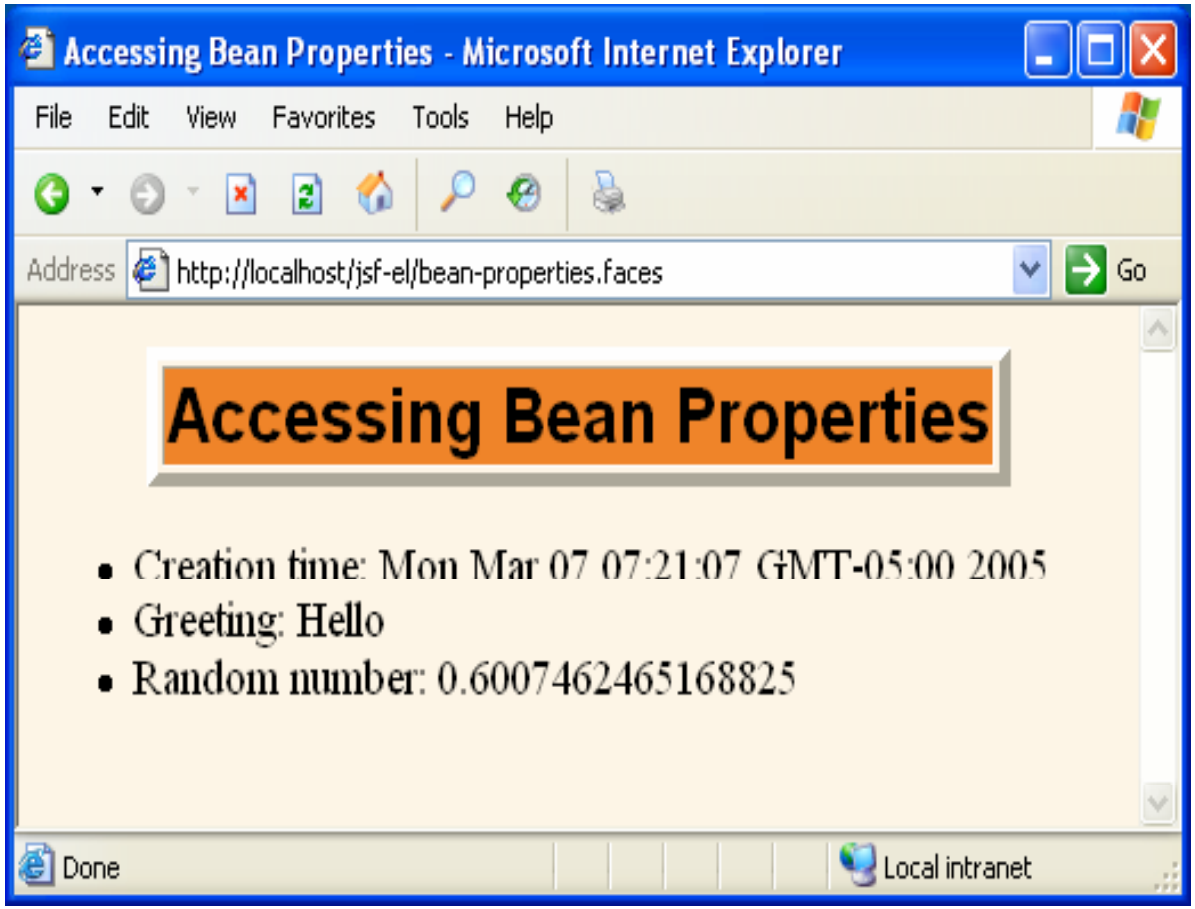
Primer bean-properties.jsp

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
...
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">Accessing Bean Properties</TH></TR>
</TABLE>
<UL>
<LI>Creation time:
<h:outputText value="#{testBean.creationTime}"/>
<LI>Greeting:
<h:outputText value="#{testBean.greeting}"/>
<LI>Random number:
<h:outputText value="#{testBean.randomNumber}"/>
</UL>
</BODY></HTML>
</f:view>
```

Primer web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
...
version="2.4">
...
<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>
<url-pattern>/bean-properties.jsp</url-pattern>
...
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Primer rezultat



Ugnežđeni property-iji bean-a

- **#{varName.prop1.prop2}**
  – Prvo se izvršava pretraga za odgovarajuću definiciju bean-a po imenu varName
  – Tada se pristupa property-iju prop1 (poziva se metod getProp1)
  – Tada se pristupa property-iju prop2 dobijenog rezultata (poziva se metod objekta koji je dobijen sa getProp1)

Primer NameBean

```
package coreservlets;

public class NameBean {
  private String firstName = "Missing first name";
  private String lastName = "Missing last name";

  public NameBean() {}

  public NameBean(String firstName, String lastName) {
    setFirstName(firstName);
    setLastName(lastName);
  }

  public String getFirstName() {
    return(firstName);
  }

  public void setFirstName(String newFirstName) {
    firstName = newFirstName;
  }

  ...
```

Primer CompanyBean

```
package coreservlets;

public class CompanyBean {
  private String companyName;
  private String business;

  public CompanyBean(String companyName,
                     String business) {
    setCompanyName(companyName);
    setBusiness(business);
  }

  public String getCompanyName() { return(companyName); }

  public void setCompanyName(String newCompanyName) {
    companyName = newCompanyName;
  }

  ...
}
```

Primer EmployeeBean

```
package coreservlets;

public class EmployeeBean {
  private NameBean name;
  private CompanyBean company;

  public EmployeeBean(NameBean name, CompanyBean company) {
    setName(name);
    setCompany(company);
  }

  public EmployeeBean() {
   this(new NameBean("Marty", "Hall"),
        new CompanyBean("coreservlets.com",
                        "J2EE Training and Consulting"));
  }

  public NameBean getName() { return(name); }

  public void setName(NameBean newName) {
    name = newName;
  }
```

Primer faces-config.xml

```
<faces-config>
...
<managed-bean>
<managed-bean-name>employee</managed-bean-name>
<managed-bean-class>
coreservlets.EmployeeBean
</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
...
</faces-config>
```

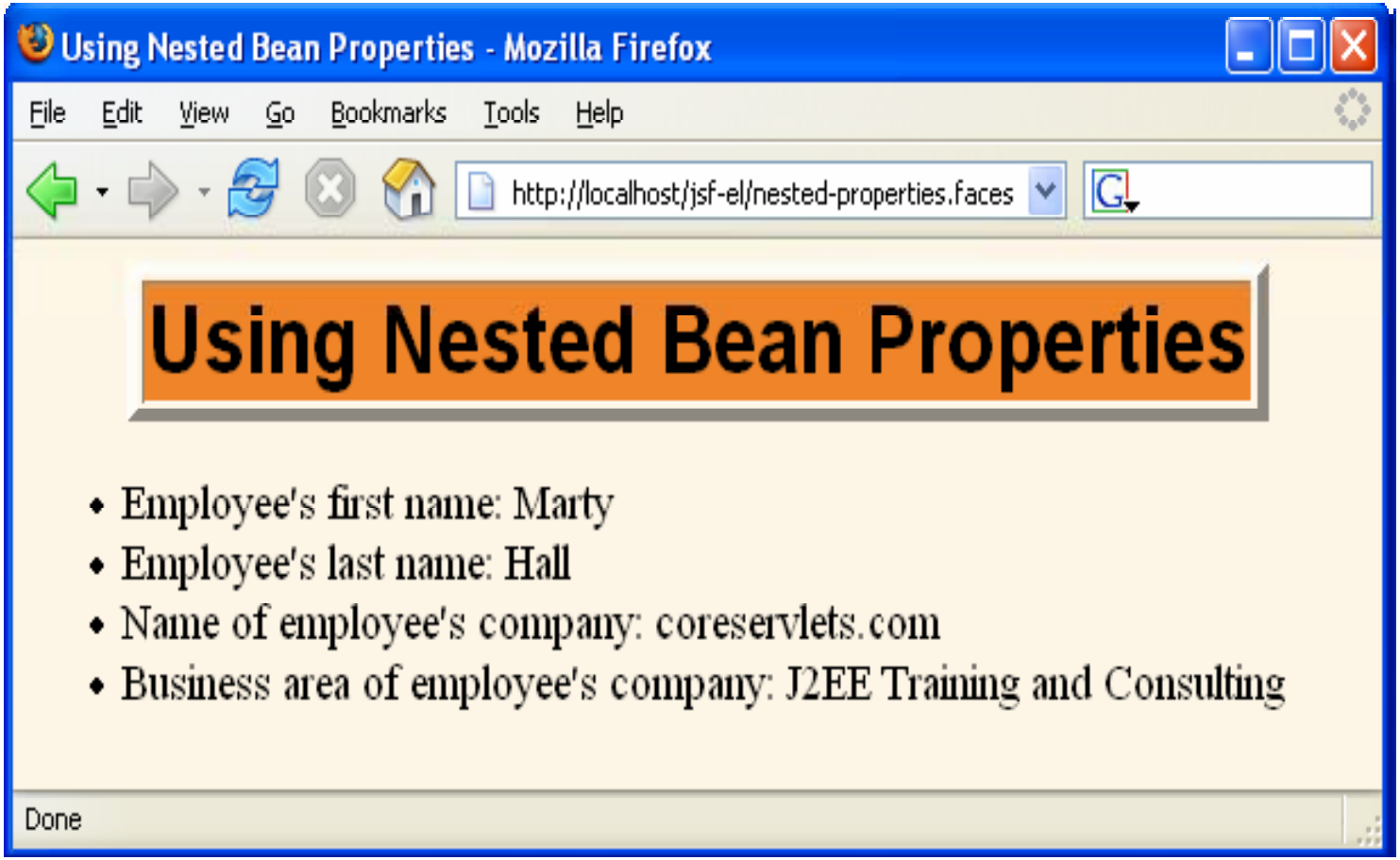
Primer nested-properties.jsp

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
...
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">Using Nested Bean Properties</TH></TR>
</TABLE>
<UL>
<LI>Employee's first name:
<h:outputText value="#{employee.name.firstName}"/>
<LI>Employee's last name:
<h:outputText value="#{employee.name.lastName}"/>
<LI>Name of employee's company:
<h:outputText value="#{employee.company.companyName}"/>
<LI>Business area of employee's company:
<h:outputText value="#{employee.company.business}"/>
</UL>
</BODY></HTML>
</f:view>
```

Primer web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
...
version="2.4">
...
<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>
<url-pattern>/nested-properties.jsp</url-pattern>
...
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Primer rezultat



Tri značenja #{...}

- **Dizajn izlaznih vrednosti**
  - – #{varName.propertyName} prikazuje vrednost property-ija date promenljive
  - – <h:outputText value="#{employee.address}"/>
  - • Bilo kada da se pristupa, podrazumeva izalzni tekst
  - – <h:inputText value="#{employee.address}"/>
  - • Kada se forma inicijalno prikazuje, prikazuje se predefinisana vrednost
- **Dizajn vrednosti koja se šalje**
  - – <h:inputText value="#{employee.address}"/>
  - • Kada se forma šalje, definiše se gde se smešta vrednost
- **Dizajn poziva metoda kod slanja**
  - – <h:commandButton value="Button Label" action="#{employee.processEmployee}"/>
  - • Kada se forma šalje, definiše se određena akcija

Primer EmployeeBean

```
package coreservlets;
public class EmployeeBean {
private NameBean name;
private CompanyBean company;
...
public String processEmployee() {
if (Math.random() < 0.5) {
return("accept");
} else {
return("reject");
}
}
}
```

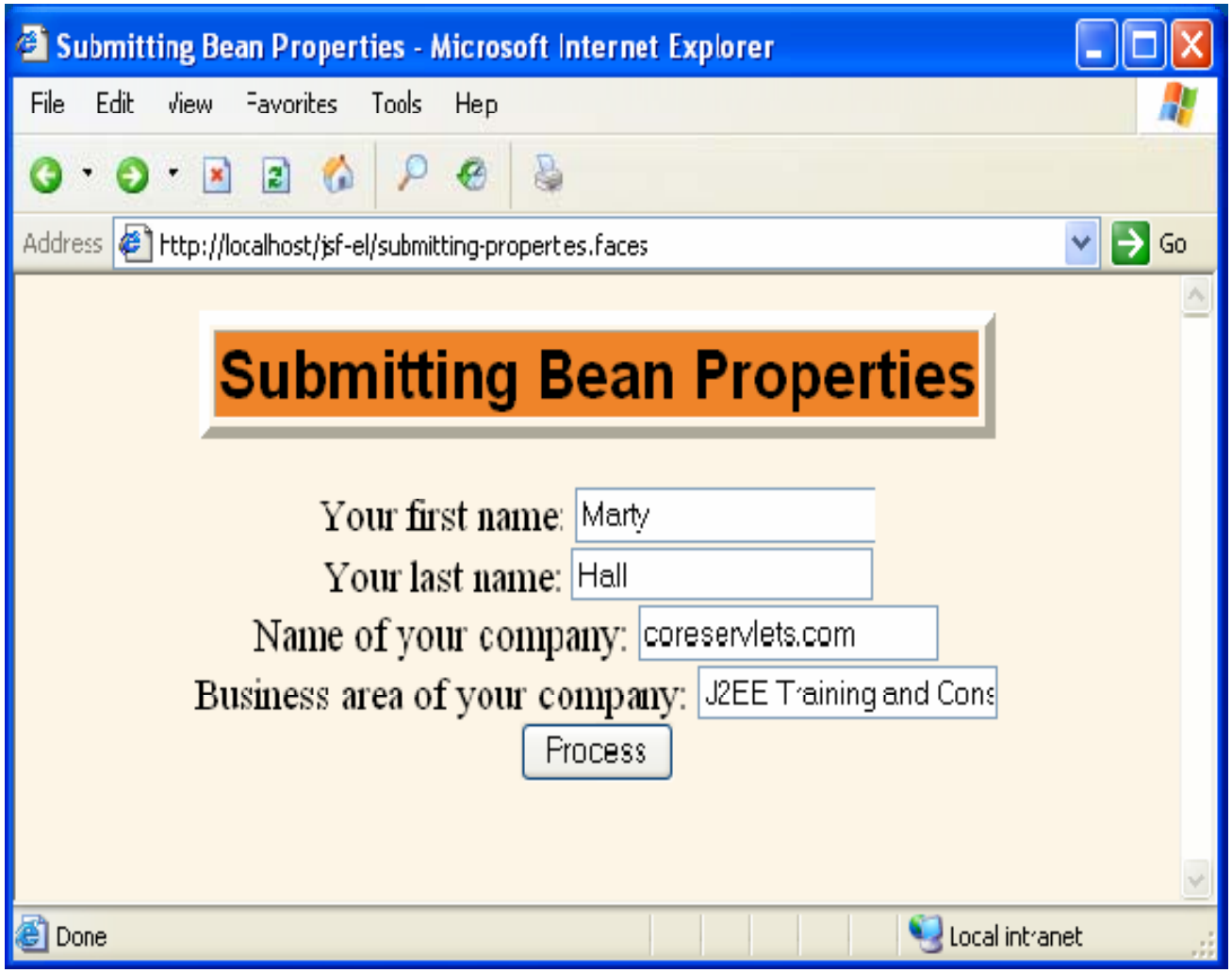
Primer faces-config.xml

```
<faces-config>
...
<managed-bean>
<managed-bean-name>employee</managed-bean-name>
<managed-bean-class>
coreservlets.EmployeeBean
</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
...
<navigation-rule>
<from-view-id>/submitting-properties.jsp</from-view-id>
<navigation-case>
<from-outcome>accept</from-outcome>
<to-view-id>/WEB-INF/results/accept.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>reject</from-outcome>
<to-view-id>/WEB-INF/results/reject.jsp</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>
```

Primer web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
...
version="2.4">
...
<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>
<url-pattern>/submitting-properties.jsp</url-pattern>
...
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Primer web.xml



Primer accept.jsp (JSF verzija)

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
...
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">Employee Accepted</TH></TR>
</TABLE>
<UL>
<LI>Employee's first name:
<h:outputText value="#{employee.name.firstName}"/>
<LI>Employee's last name:
<h:outputText value="#{employee.name.lastName}"/>
<LI>Name of employee's company:
<h:outputText value="#{employee.company.companyName}"/>
<LI>Business area of employee's company:
<h:outputText value="#{employee.company.business}"/>
</UL>
Congratulations.
</BODY></HTML>
</f:view>
```

Primer accept.jsp (JSP verzija)

```
...
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">Employee Accepted</TH></TR>
</TABLE>
<UL>
<LI>Employee's first name:
${employee.name.firstName}
<LI>Employee's last name:
${employee.name.lastName}
<LI>Name of employee's company:
${employee.company.companyName}
<LI>Business area of employee's company:
${employee.company.business}
</UL>
Congratulations.
</BODY></HTML>
```

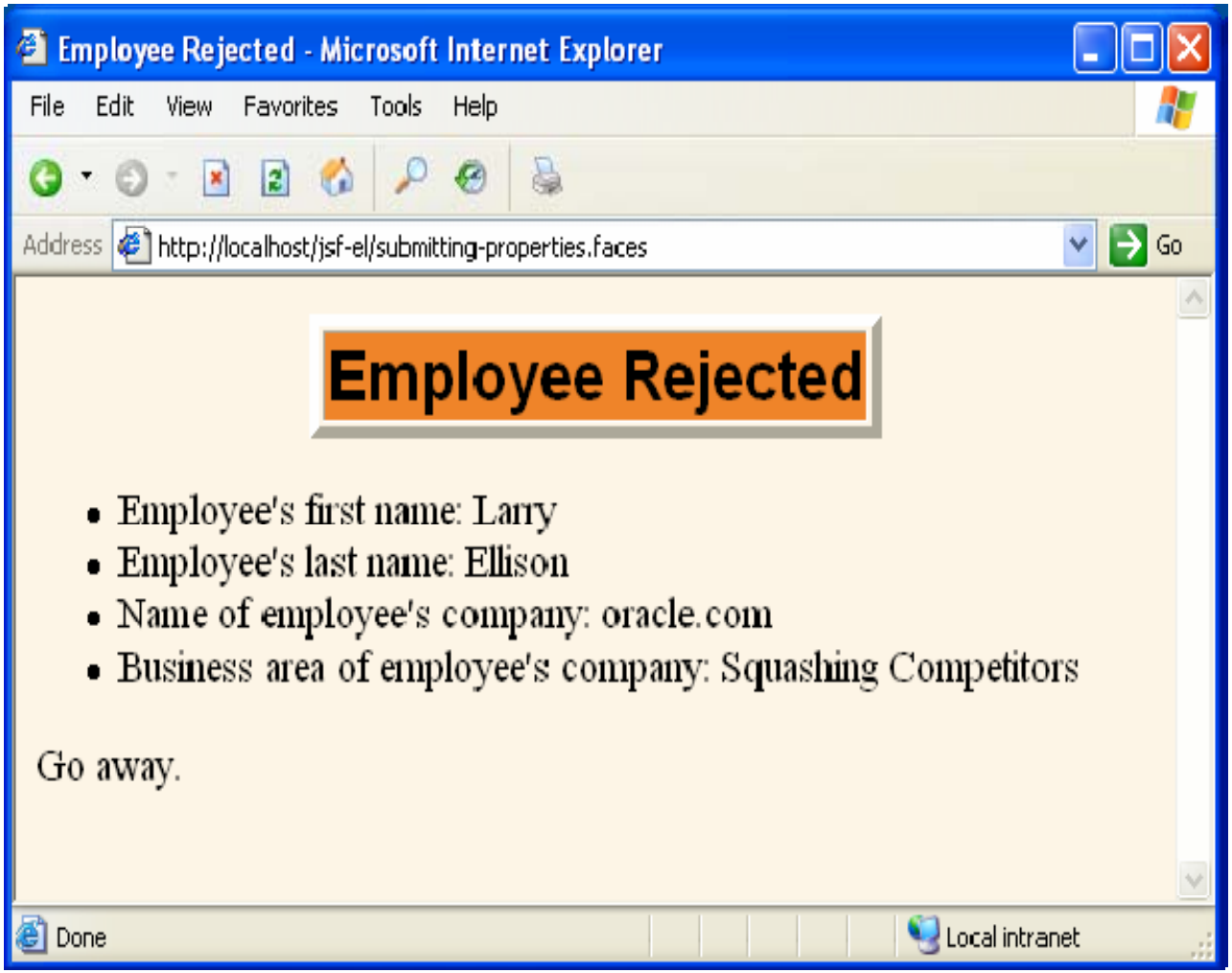
Primer reject.jsp (JSF verzija)

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
...
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">Employee Rejected</TH></TR>
</TABLE>
<UL>
<LI>Employee's first name:
<h:outputText value="#{employee.name.firstName}"/>
<LI>Employee's last name:
<h:outputText value="#{employee.name.lastName}"/>
<LI>Name of employee's company:
<h:outputText value="#{employee.company.companyName}"/>
<LI>Business area of employee's company:
<h:outputText value="#{employee.company.business}"/>
</UL>
Congratulations.
</BODY></HTML>
</f:view>
```

Primer reject.jsp (JSP verzija)

```
...
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">Employee Rejected</TH></TR>
</TABLE>
<UL>
<LI>Employee's first name:
${employee.name.firstName}
<LI>Employee's last name:
${employee.name.lastName}
<LI>Name of employee's company:
${employee.company.companyName}
<LI>Business area of employee's company:
${employee.company.business}
</UL>
Congratulations.
</BODY></HTML>
```

Primer rezultat



Pristup kolekcijama

- **Ekvivalentne forme**
  - #{name.property}
  - #{name["property"]}
- **Razlozi za upotrebu notacije nizova**
  - Za pristup nizovima, listama, i drugim kolekcijama
  - Da bi se izračunalo ime property-ija u trenutku zahteva.
    - #{name1[name2]} (nema navodnika oko name2)
  - Da bi se koristila imena koja su nelegalna kao imena Java promenljivih
    - #{foo["bar-baz"]}
    - #{foo["bar.baz"]}

Pristup kolekcijama

- **#{attributeName[entryName]}**
- **Moguće je sa**
  - Nizovima. Ekvivalentno sa
    - theArray[index]
  - Listama. Ekvivalentno sa
    - theList.get(index) ili theList.set(index, *submitted-val*)
  - Mapama. Ekvivalentno sa
    - theMap.get(key) ili theMap.put(key, *submitted-val*)
- **Ekvivalentne forme (za HashMap)**
  - #{stateCapitals["maryland"]}
  - #{stateCapitals.maryland}
  - Ali sledeći izraz nije dozvoljen, jer 2 nije legalno ime za promenljivu
    - #{listVar.2}

Primer PurchaseBean

```
public class PurchaseBean {
  private String[] cheapItems =
    { "Gum", "Yo-yo", "Pencil" };
  private ArrayList mediumItems =
    new ArrayList();
  private HashMap valuableItems =
    new HashMap();
  private boolean isEverythingOK = true;

  public PurchaseBean() {
    mediumItems.add("iPod");
    mediumItems.add("GameBoy");
    mediumItems.add("Cell Phone");
    valuableItems.put("low", "Lamborghini");
    valuableItems.put("medium", "Yacht");
    valuableItems.put("high", "Chalet");
  }

  public String[] getCheapItems() {
    return(cheapItems);
  }
  public List getMediumItems() {
    return(mediumItems);
  }
  public Map getValuableItems() {
    return(valuableItems);
  }
```

Primer PurchaseBean

```
public String purchaseItems() {
isEverythingOK = Utils.doBusinessLogic(this);
isEverythingOK = Utils.doDataAccessLogic(this);
if (isEverythingOK) {
return("success");
} else {
return("failure");
}
}
}
```

Primer Utils

```
public class Utils {
public static boolean doBusinessLogic
(PurchaseBean bean) {
// Business logic omitted
return(Math.random() > 0.1);
}
public static boolean doDataAccessLogic
(PurchaseBean bean) {
// Database access omitted
return(Math.random() > 0.1);
}
}
```

Primer faces-config.xml

```
<faces-config>
...
<managed-bean>
<managed-bean-name>purchases</managed-bean-name>
<managed-bean-class>
coreservlets.PurchaseBean
</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
...
<navigation-rule>
<from-view-id>/using-collections.jsp</from-view-id>
<navigation-case>
<from-outcome>success</from-outcome>
<to-view-id>/WEB-INF/results/success.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>failure</from-outcome>
<to-view-id>/WEB-INF/results/failure.jsp</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>
```

Primer using-collections.jsp

```
...
<h:form>
<UL>
<LI><B>Cheap Items</B>
<OL>
<LI><h:inputText
value="#{purchases.cheapItems[0]}"/>
<LI><h:inputText
value="#{purchases.cheapItems[1]}"/>
<LI><h:inputText
value="#{purchases.cheapItems[2]}"/>
</OL>
<LI><B>Medium Items</B>
<OL>
<LI><h:inputText
value="#{purchases.mediumItems[0]}"/>
<LI><h:inputText
value="#{purchases.mediumItems[1]}"/>
<LI><h:inputText
value="#{purchases.mediumItems[2]}"/>
</OL>
```

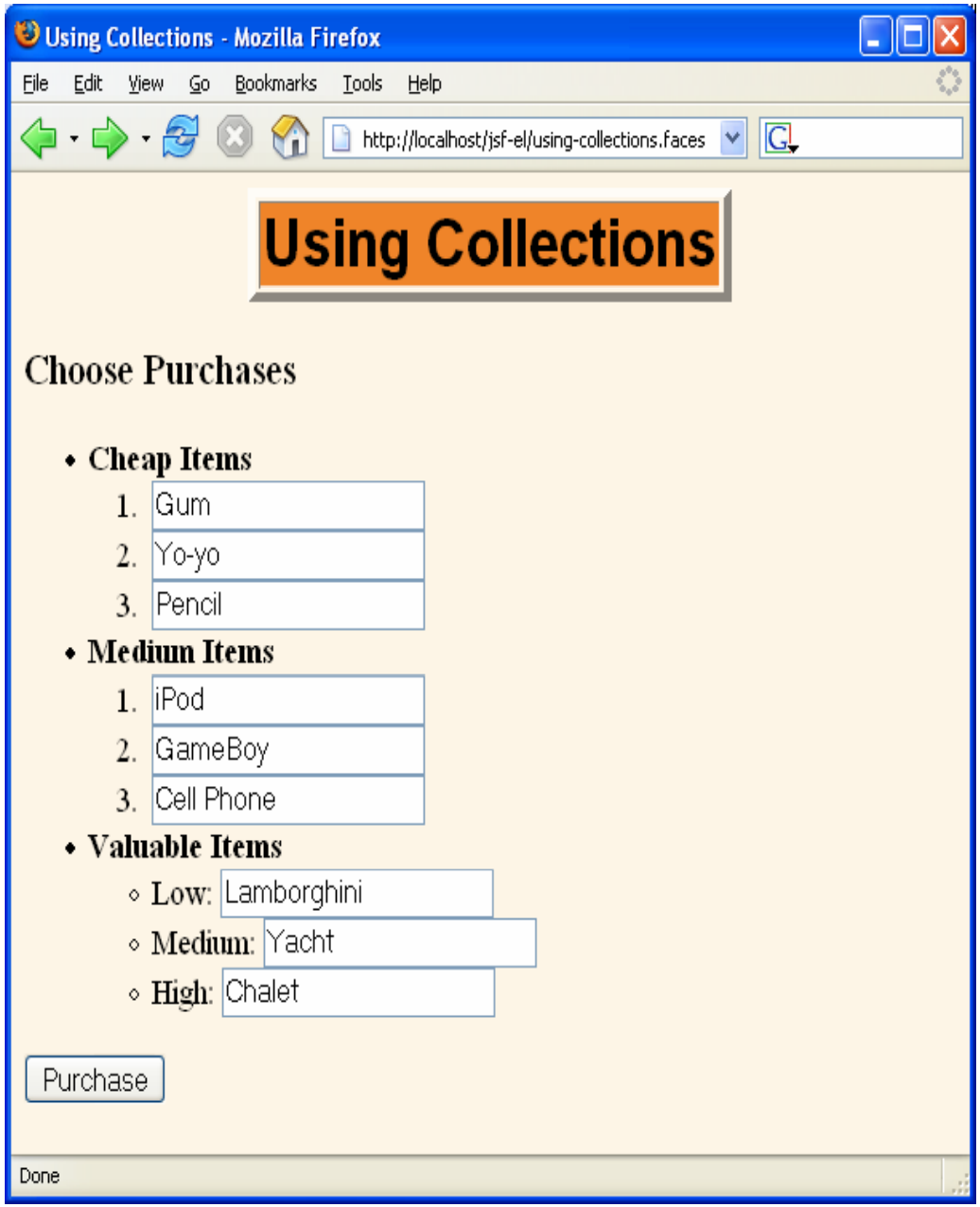
Primer using-collections.jsp

```
<LI><B>Valuable Items</B>
<UL>
<LI>Low:
<h:inputText
value='#{purchases.valuableItems["low"]}'/>
<LI>Medium:
<h:inputText
value='#{purchases.valuableItems["medium"]}'/>
<LI>High:
<h:inputText
value='#{purchases.valuableItems["high"]}'/>
</UL>
</UL>
<h:commandButton value="Purchase"
action="#{purchases.purchaseItems}"/>
</h:form>
...
```

Primer web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
...
version="2.4">
...
<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>
<url-pattern>/using-collections.jsp</url-pattern>
...
</web-resource-collection>
<auth-constraint>
<description>No roles, so no direct access</description>
</auth-constraint>
</security-constraint>
</web-app>
```

Primer rezultat



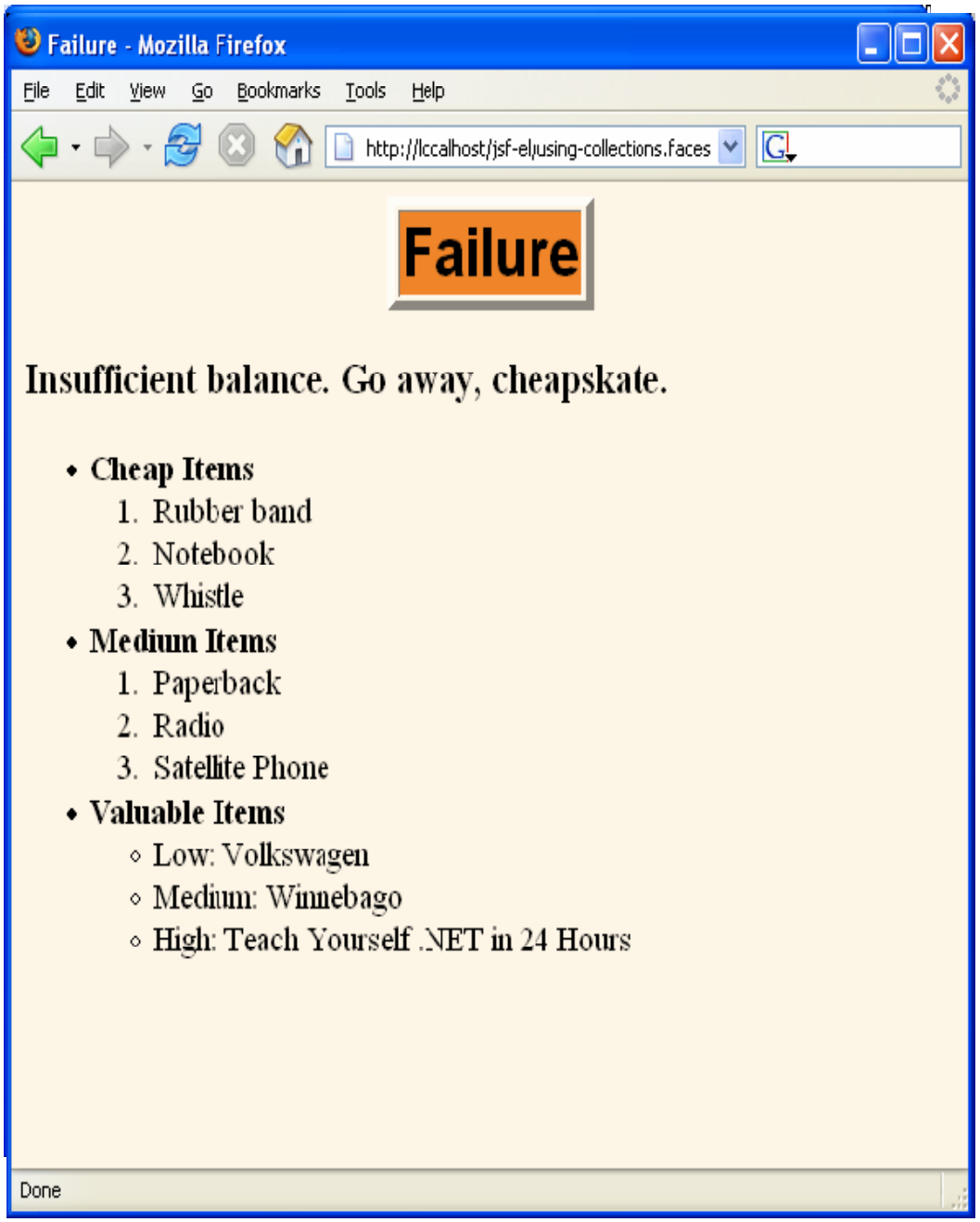
Primer success.jsp (JSF verzija)

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>...
<UL>
<LI><B>Cheap Items</B>
<OL>
<LI><h:outputText
value="#{purchases.cheapItems[0]}"/>
<LI><h:outputText
value="#{purchases.cheapItems[1]}"/>
<LI><h:outputText
value="#{purchases.cheapItems[2]}"/>
</OL>
<LI><B>Medium Items</B>
<OL>
<LI><h:outputText
value="#{purchases.mediumItems[0]}"/>
<LI><h:outputText
value="#{purchases.mediumItems[1]}"/>
<LI><h:outputText
value="#{purchases.mediumItems[2]}"/>
</OL>
...</f:view>
```

Primer success.jsp (JSP verzija)

```
...
<UL>
<LI><B>Cheap Items</B>
<OL>
<LI>${purchases.cheapItems[0]}
<LI>${purchases.cheapItems[1]}
<LI>${purchases.cheapItems[2]}
</OL>
<LI><B>Medium Items</B>
<OL>
<LI>${purchases.mediumItems[0]}
<LI>${purchases.mediumItems[1]}
<LI>${purchases.mediumItems[2]}
</OL>
...
```

Primer rezultat



Implicitni objekti

```
<!DOCTYPE …>
…
<P>
<UL>
<LI><B>test Request Parameter:</B>
${param.test}
<LI><B>User-Agent Header:</B>
${header["User-Agent"]}
<LI><B>JSESSIONID Cookie Value:</B>
${cookie.JSESSIONID.value}
<LI><B>Server:</B>
${pageContext.servletContext.serverInfo}
</UL>
</BODY></HTML>
```

Operatori

```
...
<TABLE BORDER=1 ALIGN="CENTER">
<TR><TH CLASS="COLORED" COLSPAN=2>Arithmetic Operators
<TH CLASS="COLORED" COLSPAN=2>Relational Operators
<TR><TH>Expression<TH>Result<TH>Expression<TH>Result
<TR ALIGN="CENTER">
<TD>\${3+2-1}<TD>${3+2-1}
<TD>\${1&lt;2}<TD>${1<2}
<TR ALIGN="CENTER">
<TD>\${"1"+2}<TD>${"1"+2}
<TD>\${"a"&lt;"b"}<TD>${"a"<"b"}
<TR ALIGN="CENTER">
<TD>\${1 + 2*3 + 3/4}<TD>${1 + 2*3 + 3/4}
<TD>\${2/3 &gt;= 3/2}<TD>${2/3 >= 3/2}
<TR ALIGN="CENTER">
<TD>\${3%2}<TD>${3%2}
<TD>\${3/4 == 0.75}<TD>${3/4 == 0.75}
```

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**

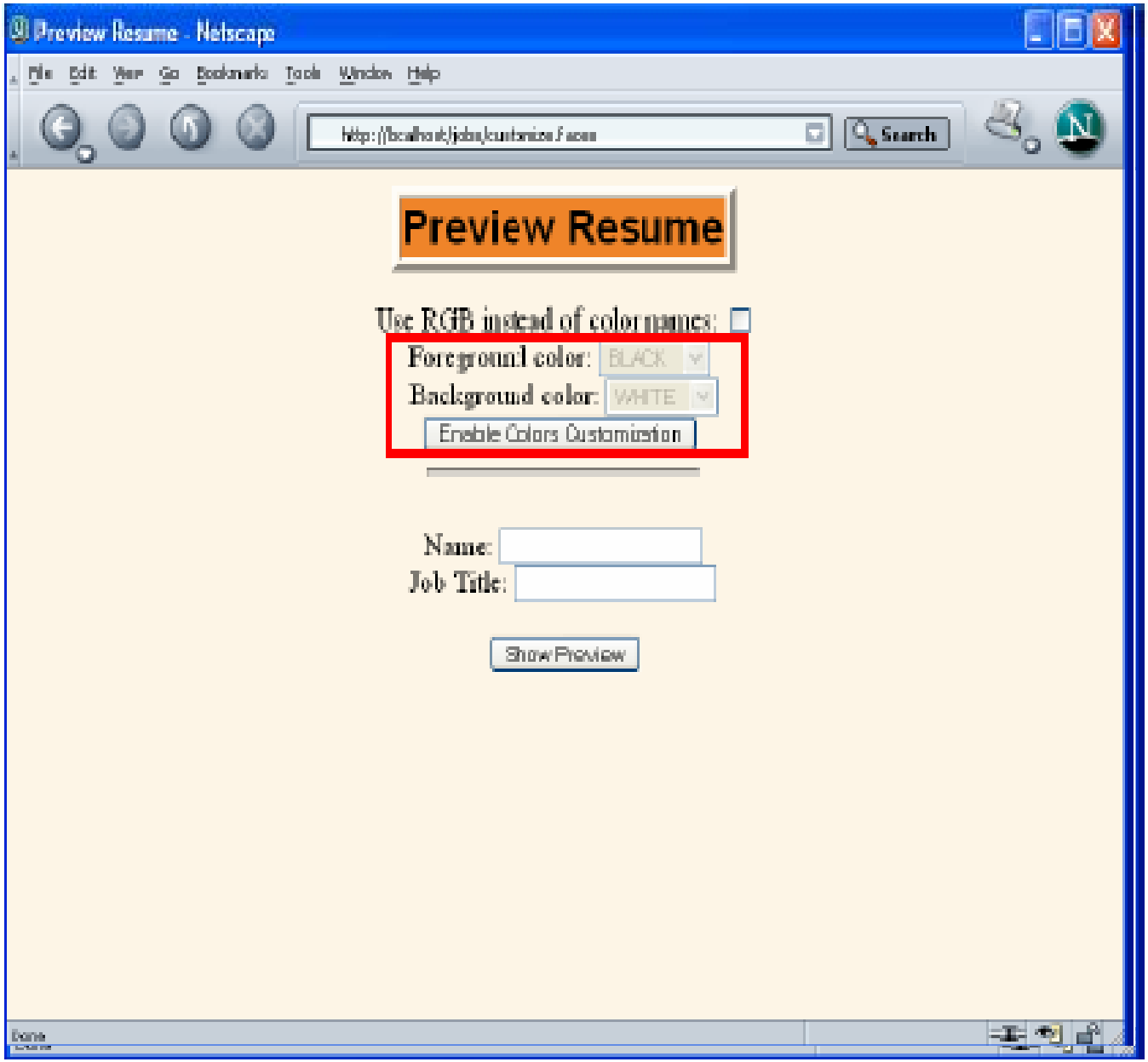
– Kod za kontrolu je unutar form bean-a (ResumeBean)

– 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**

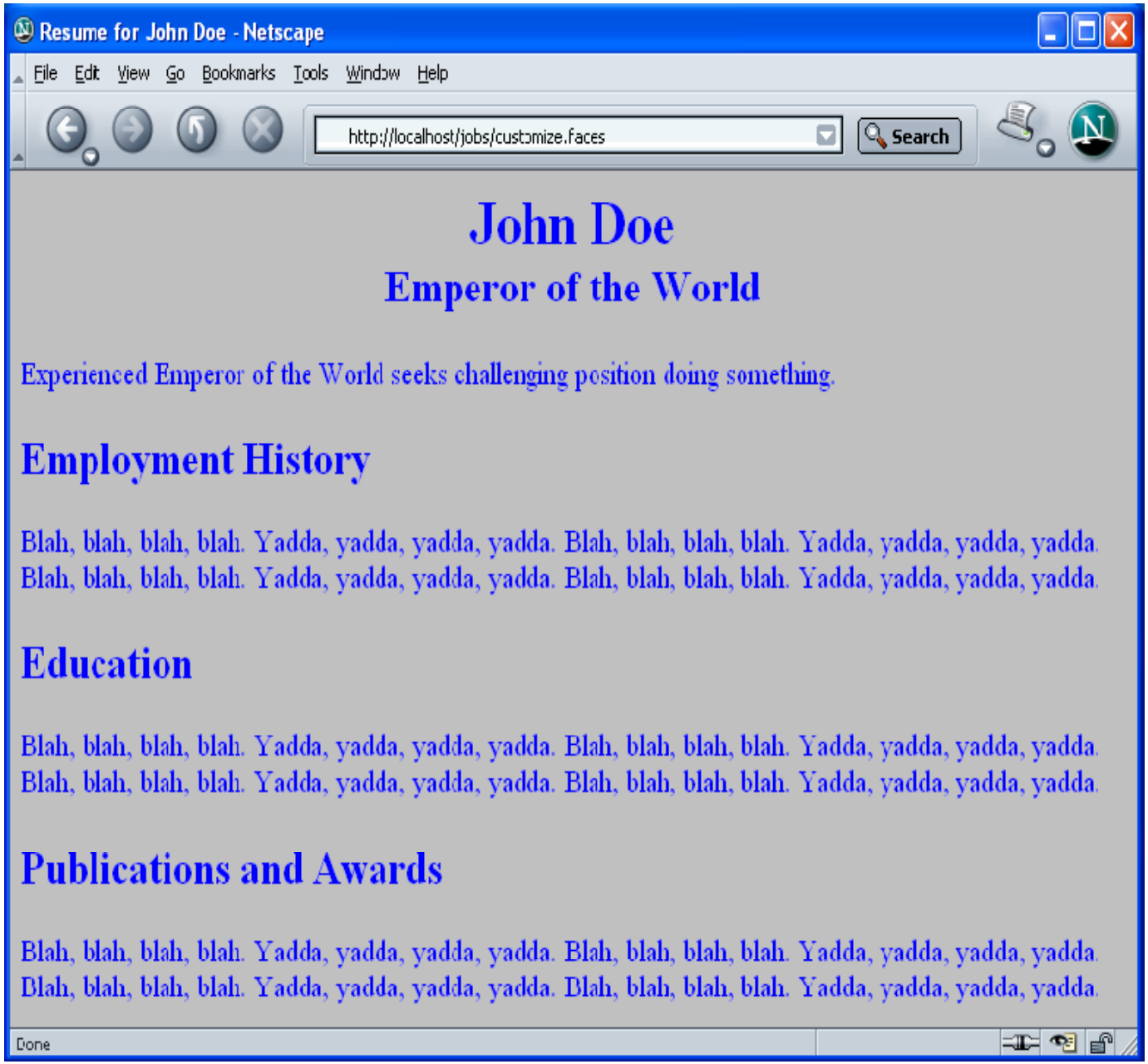
```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<!DOCTYPE ...>
<HTML>
...
You chose
"<h:outputText value="#{resumeBean.fgColor}"/>"
as both the foreground and background color.
<P>
You don't deserve to get a job, but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

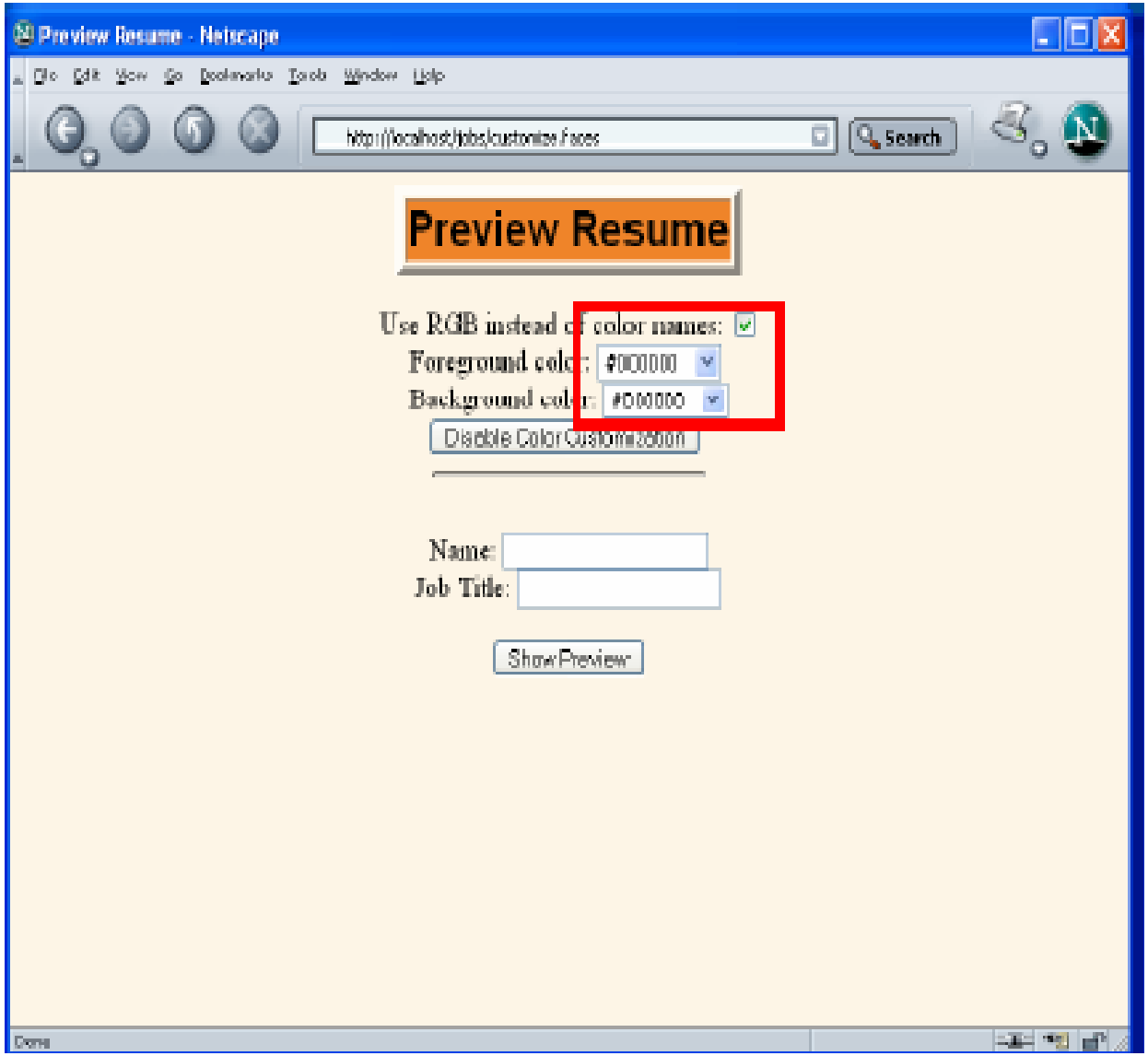
```
<h:selectBooleanCheckbox
valueChangeListener="#{resumeBean.changeColorMode}"
onclick="submit()"
immediate="true"/>
```

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
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

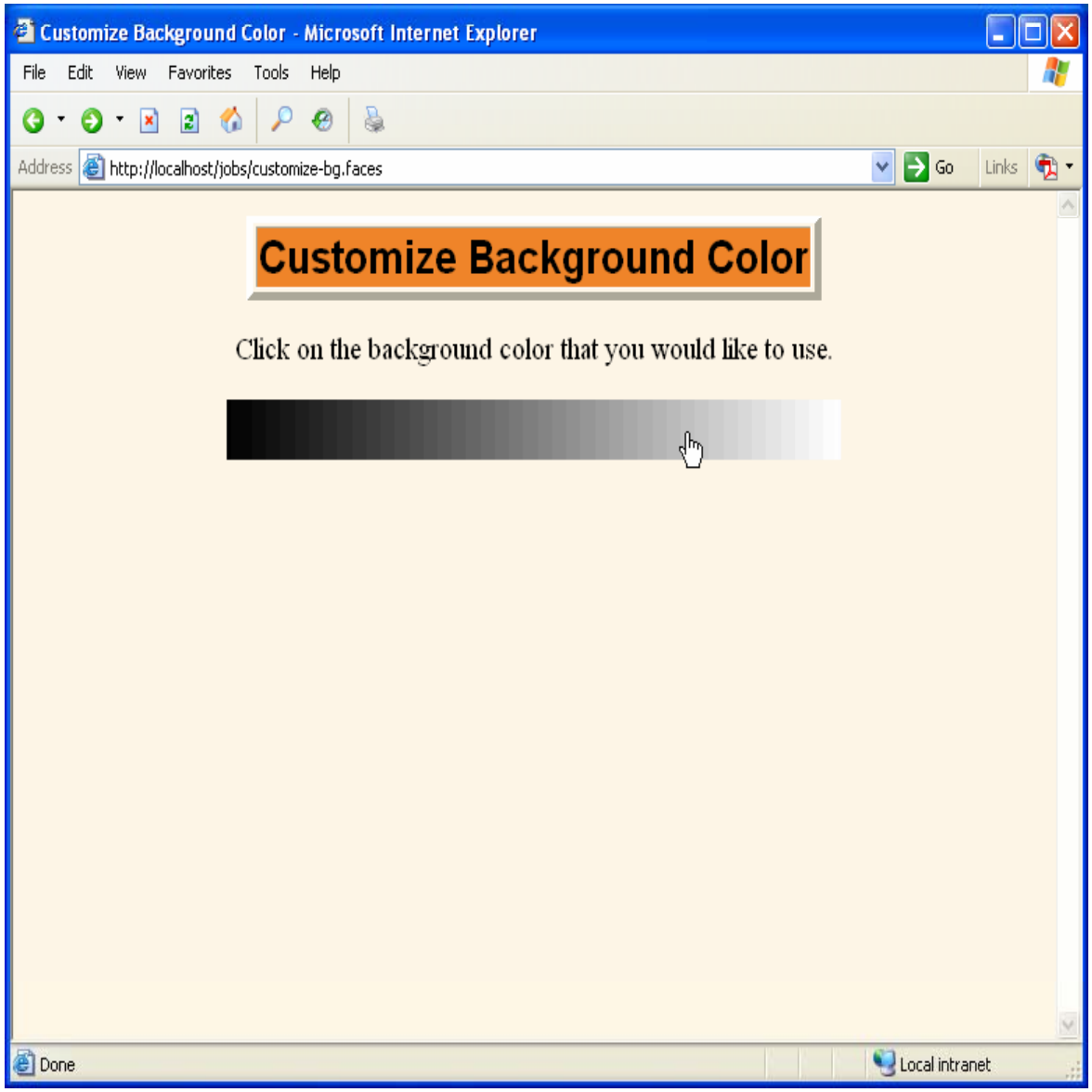
```
package coreservlets;
...
public class ColorBean {
private String bgColor = "WHITE";
public String getBgColor() { return(bgColor); }
public void setBgColor(String bgColor) {
this.bgColor = bgColor;
}
```

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**

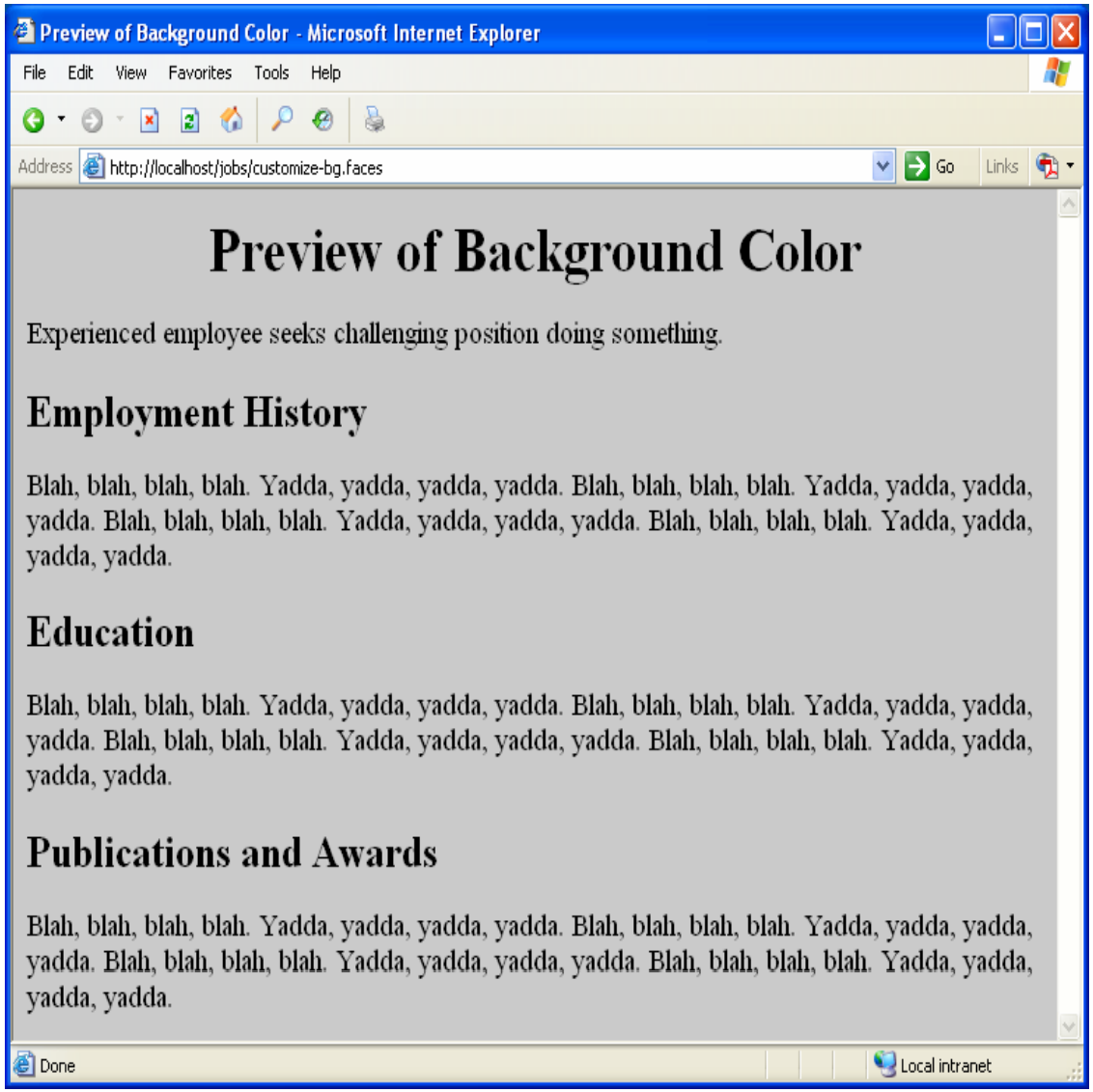
```
...
<faces-config>
...
<navigation-rule>
<from-view-id>/customize-bg.jsp</from-view-id>
<navigation-case>
<from-outcome>success</from-outcome>
<to-view-id>
/WEB-INF/results/show-preview2.jsp
</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>
```

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>
```

h biblioteka

- **h:form**
  - Ne specificira se ACTION specified
  - Mora da se koristi POST!
- **h:inputText**
  - NAME se generise automatski
  - VALUE je u formi #{beanName.propertyName}
- **h:inputSecret**
  - NAME se generise automatski
  - VALUE se primenjuje samo na izlaz, ne na ulaz
- **h:commandButton**
  - Akcija odgovara atributu ACTION taga forme
- **h:outputText**
  - Prikazuju se property-iji bean-a

h biblioteka

- **Deljeni atributi**

  – Ove atribute je moguće koristiti u skoro svim h:blah elementima

  – Neće se ponavljati prilikom opisa svakog elementa

- **Najviše korišćeni deljeni atributi**

  – **id**. Identifikator; koristi se da bi se dodelilo ime elementu, tako da može da mu se pristupi kasnije (n.p., za validaciju)

  – **onclick**, **ondblclick**, ... JavaScript obrada događaja.

  • Obratiti pažnju na velika i mala slova, onClick nije ispravno.

  – **styleClass**. CSS ime stila.

  • **Očekivani atributi**

  – Ovi atributi su oni koji odgovaraju određenom HTML elementu, n.p. size i maxlength za tekst polje

h:form

- **Odgovarajući HTML element**
  – <FORM ...>
- **Mogućnosti**
  – Ne specificira se akcija
- **Atribut**
  – **enctype** - Tip kodiranja.
  • application/x-www-form-urlencoded (default)
  • multipart/form-data
  – **target**. Frame deo u kome se prikazuje rezultat
  – **onsubmit**. JavaScript kod koji se izvršava pre slanja forme
  • Može da se izvrši validacija polja na klijentskoj strani

h:inputText

- **Odgovarajući HTML element**
  – <INPUT TYPE="TEXT" …>
- **Atributi**
  – **value** – Odgovarajuća vrednost bean-a.
  • value="#{beanName.propertyName}"
  • Koristi se i za inicijalnu vrednost i pri slanju serverskoj komponenti
  – **valueChangeListener**. Koji se metod izvršava kada se šalje forma i vrednost ztekst polja je promenjena.
  – **onchange**. JavaScript koji se izvršava pri promeni vrednosti
  – **immediate**. Koristi se da pozove validaciju listener-a
  – **required**. Flag koji definiše da li korisnik mora da unese vrednost.
  • Kao je ne unese, forma se ponovo prikazuje i prikazuje se poruka o grešci.
  – **validator**. Metod koji izvršava validaciju.

h:inputSecret

- **Odgovarajući HTML element**
  – <INPUT TYPE="PASSWORD" ...>
- **Atributi**
  – **value**. Odgovarajuća vrednost bean-a.
  • value="#{beanName.propertyName}"
  • Koristi se samo za slanje podataka
  – **valueChangeListener**. Koji se metod izvršava kada se šalje forma i vrednost ztekst polja je promenjena.
  – **onchange**. JavaScript koji se izvršava pri promeni vrednosti
  – **immediate**. Koristi se da pozove validaciju listener-a
  – **required**. Flag koji definiše da li korisnik mora da unese vrednost.
  • Kao je ne unese, forma se ponovo prikazuje i prikazuje se poruka o grešci.
  – **converter**, **validator**. Metodi koji izvršavaju konverziju stringa i validaciju podataka.

h:commandButton

- **Odgovarajući HTML element**
  – <INPUT TYPE="SUBMIT" …>
- **Atributi**
  – **value**. Natpis na dugmetu. Često je statički string, ali se može dinamički menjati
  – **action**. Metod action controller-a koji se izvršava kada se šalje forma. Izvršava se kada svi listener-i završe svoj posao.
  – **actionListener**. Metod listener-a koji se izvršava kada se šalje forma. Izvršava se pre action controller-a.

h:commandButton (sa slikom)

- **Odgovarajući HTML element**
  – <INPUT TYPE="IMAGE" ...>
  • Slika na serverskoj strani
- **Atributi**
  – Isti kao na prethodnom slajdu, samo što je actionListener neophodan
  • Sami listener može pristupiti ID klijenta
  • ID klijenta nam je potreban da bi se dobile x i y lokacije

h:commandLink

- **Odgovarajući HTML element**

  – <A HREF="..."> sa povezanim JavaScript kodom koji šalje formu kada se link aktivira

  – JavaScript se ubacuje automatski pomoću JSF

  • **Atributi**

  – **value**. Tekst linka. Često je statički string, ali se može dinamički menjati

  – **action**. Metod action controller-a koji se izvršava kada se šalje forma. Izvršava se kada svi listener-i završe svoj posao.

  – **actionListener**. Metod listener-a koji se izvršava kada se šalje forma. Izvršava se pre action controller-a.

  • **Napomena**

  – Mora da se navede u okviru h:form (u ranijim verzijama MyFaces se javljala greška)

Elementi sa valueChangeListener atributima

- **Check boxes**

– h:selectBooleanCheckbox

  • Vrednost se dodeljuje na osnovu logičke bean property. Vrednost se postavlja na true ili false

- **Comboboxes**

– h:selectOneMenu

– h:selectManyMenu

- **List boxes**

– h:selectOneListbox

– h:selectManyListbox

- **Radio Buttons**

– h:selectOneRadio

- **Textfields**

– h:inputText

Checkbox i list box

- Specificira jedna opcija u jednom trenutku

```
<h:selectOneMenu ...>
<f:selectItem itemValue="..."
itemLabel="..."/>
<f:selectItem itemValue="..."
itemLabel="..."/>
<f:selectItem itemValue="..."
itemLabel="..."/>
</h:selectOneMenu>
```

- Specificira se početna lista

```
<h:selectOneMenu ...>
<f:selectItems value="..."/>
</h:selectOneMenu>
```

Checkbox i list box

- **Vrednost za f:selectItems mora biti List ili niz elemenata tipa java.faces.model.SelectItem**
- **SelectItem ima dva glavna konstruktora**
  - Jedan samo specificira String
    - Vrednost koja se prikazuje i vrednost koja se šalje setter metodu bean-a su isti
  - Drugi specificira Java objekat i String
    - String se prikazuje, ali se Java objekat kao vrednost šalje setter metodu bean-a (mapiranje se izvršava automatski pomoću JSF)

```
SelectItem[] listboxItems =
{ new SelectItem(realValue1, "Choice 1"),
new SelectItem(realValue2, "Choice 2"), ...
```

Mogućnosti prikaza

- **h:outputText**
  - Prikazuje se bean property ili element kolekcije
  - Vrednosti iz properties fajla se upisuju u kolekciju pomoću f:loadBundle
  - Slično bean:write iz Struts-a
  - Atributi
  - styleClass ili style: rezultat je SPAN element
  - escape: vrednost false je ne korišćenje filtera za < i >
- • **h:outputFormat**
  - Prikazuje poruke o grešci sa paramtrima
  - Parametri se prosleđuju sa f:param

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**

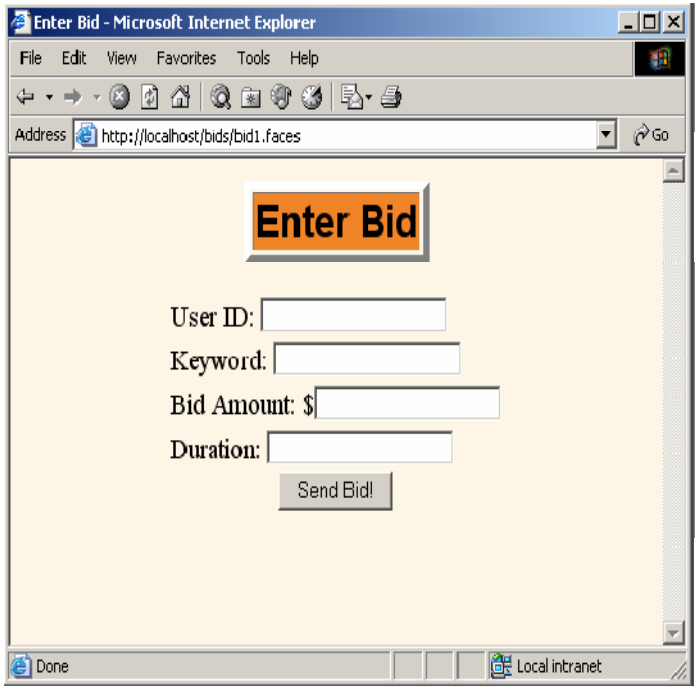
– Kreira se FacesMessage, u okviru ValidatorException, throw

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 dabude 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);
}
```

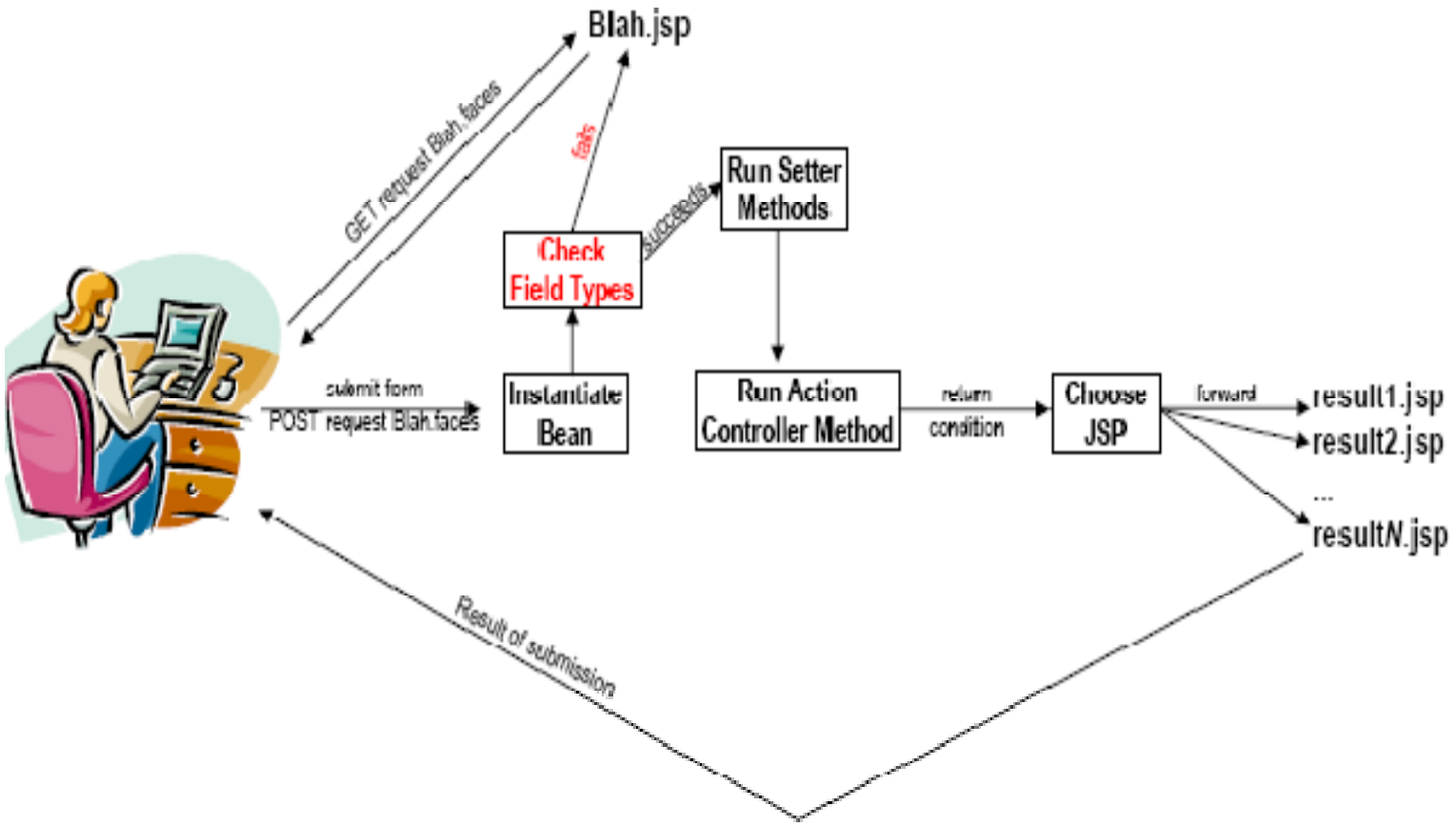
bid1.jsp

```
<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");
}
```

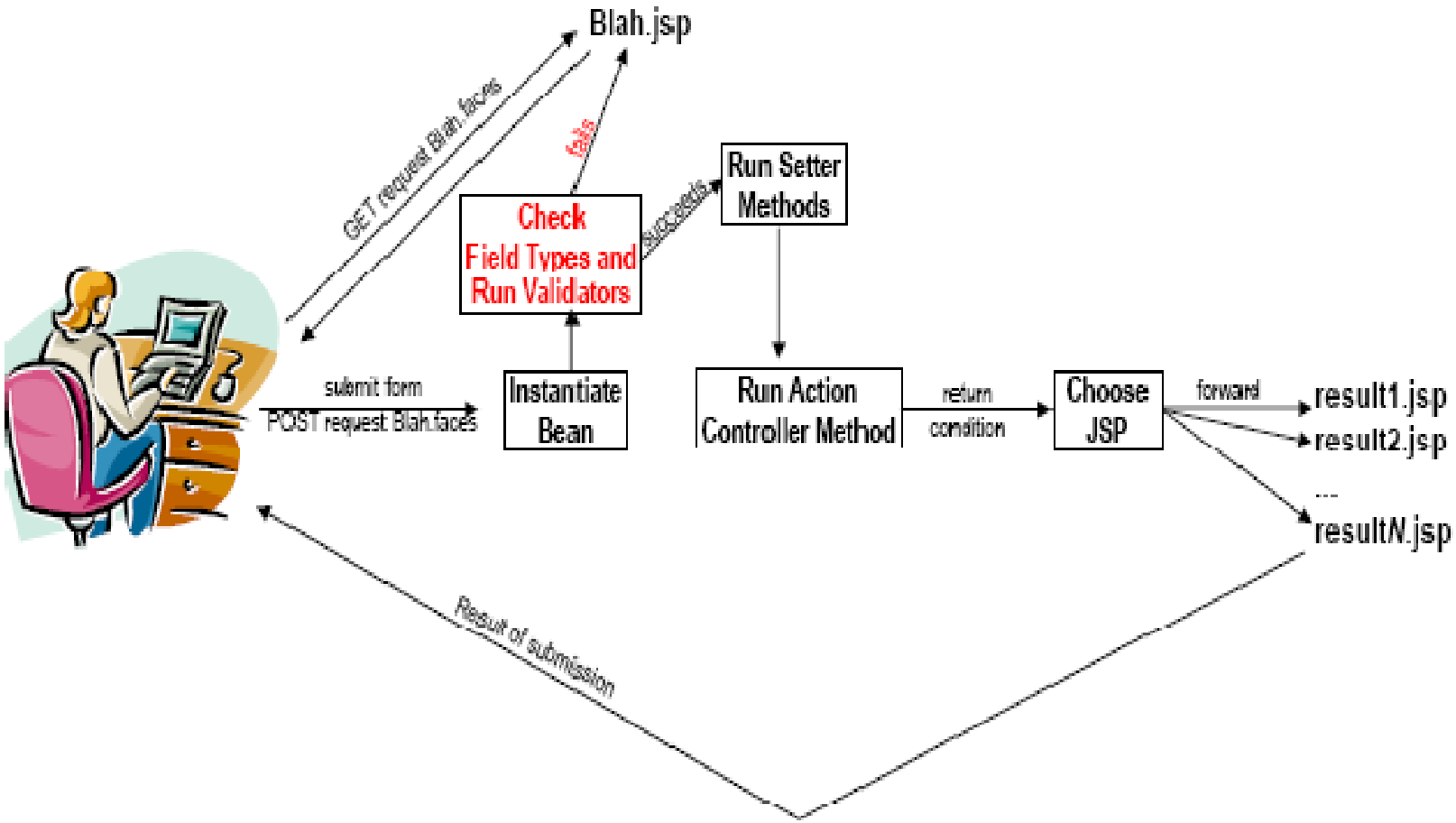
Ulazna forma

```
<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

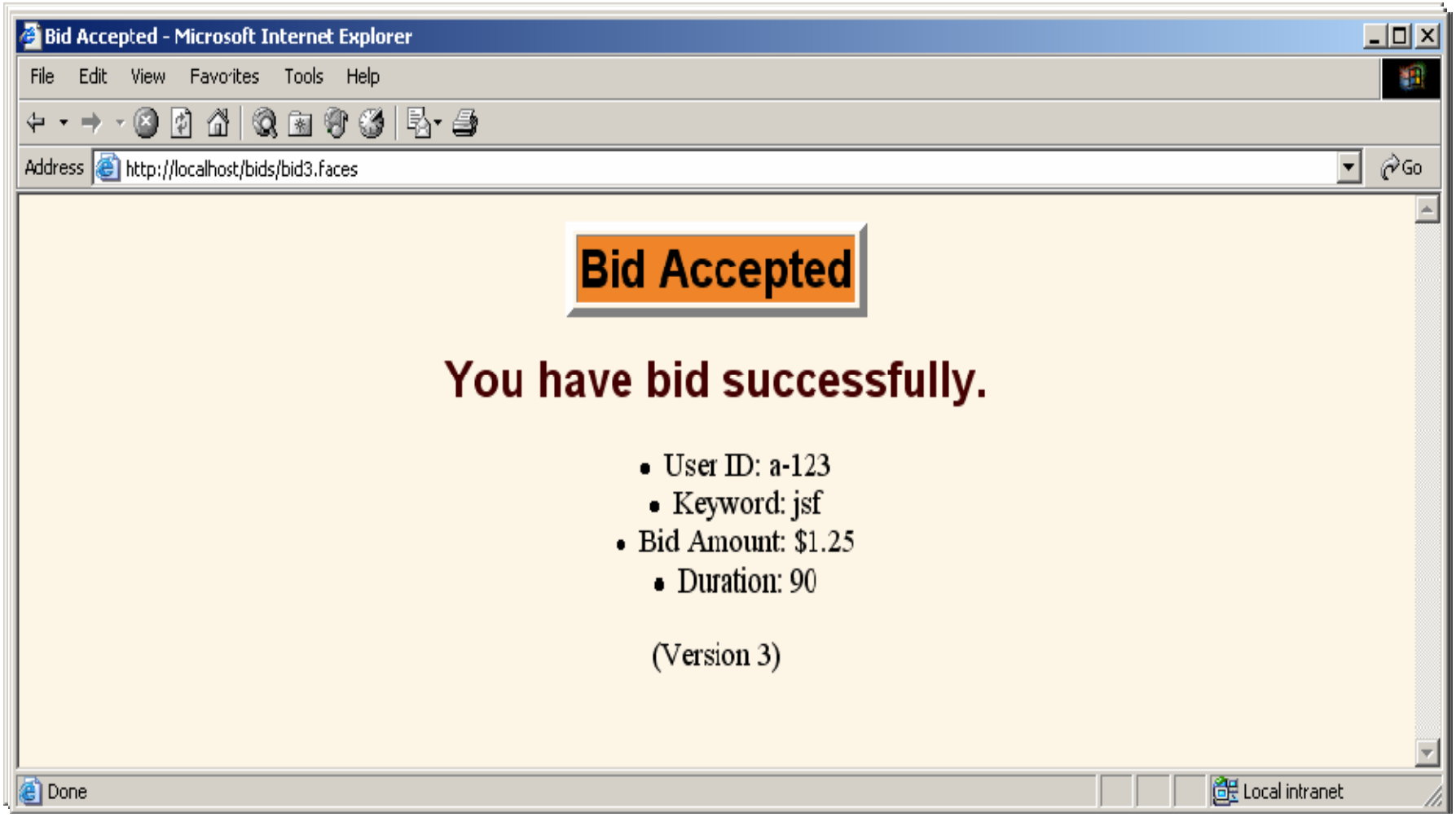
Početna forma

```
<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)

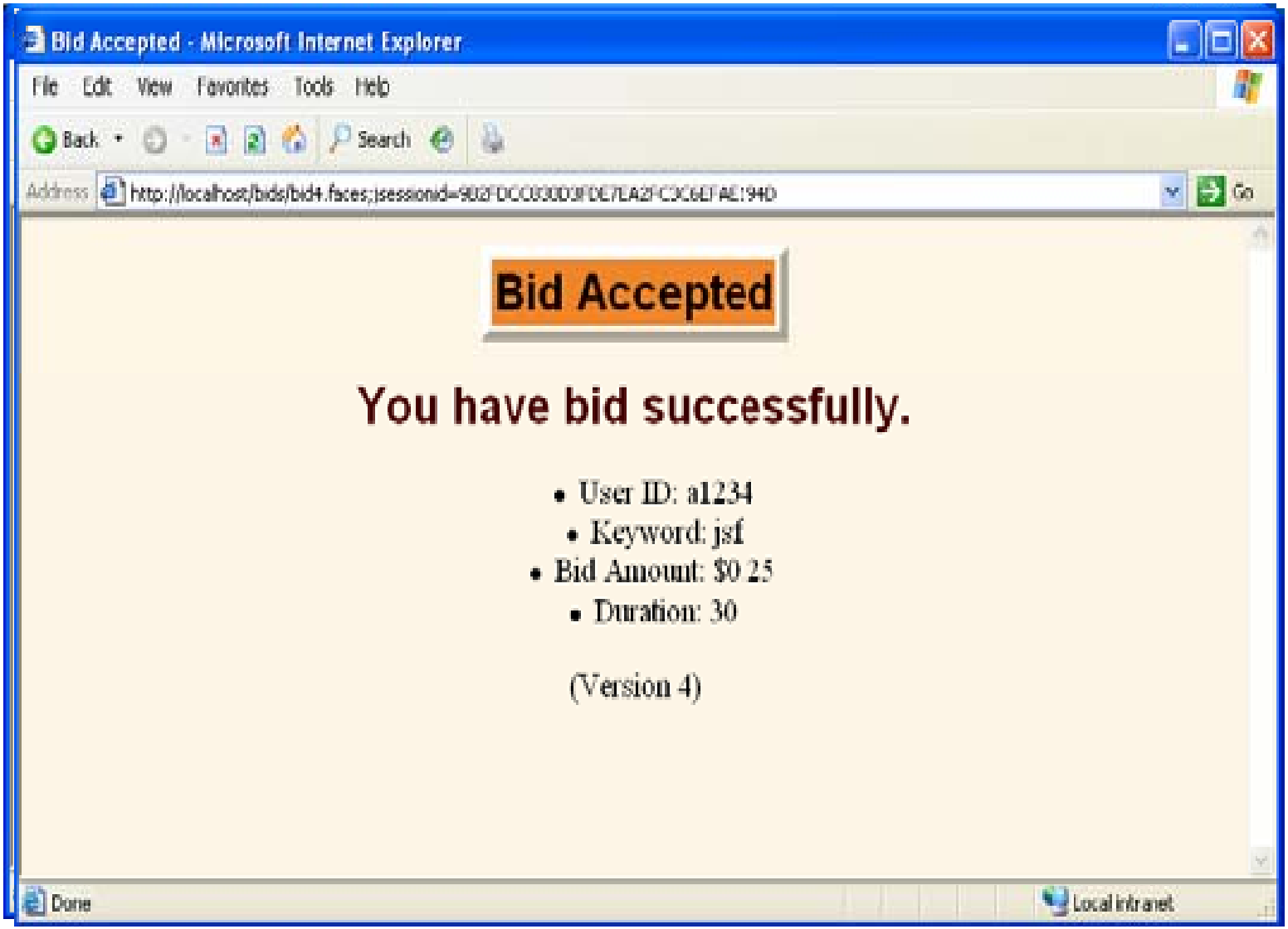
Pisanje sopstvenih metoda validacije - JSP

```
...
<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
}
```

Rezultat


Bid Accepted - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search
Address http://localhost/bids/bid4.faces;jsessionid=902FDCC0300F0E7EA2FC2C6E7AE1940
Go
Bid Accepted
You have bid successfully.
• User ID: a1234
• Keyword: jsf
• Bid Amount: $0.25
• Duration: 30
(Version 4)
Done
Local intranet

Rad sa podacima nepoznate dužine

- **Upotreba**

  – Šta se dešava ako se na serverskoj strani kreira rezultat sa nepoznatim brojem elemenata? Kako prikazati taj rezultat na JSP stranici, bez rušenja MVC modela?

- **Alternative**

  – Bez petlji

```
<h:outputText value="#{bankCustomer.depositTable}"/>
```

  <mytags:showDepositTable custID="..." month="..." styleClasses="..."/>

  – Petlje

```
<% for(...) { ... %>
```

  HTML kod

```
<% } %>
```

  • JSTL

Kreiranje HTML tabele

- **Nekada je potrebno koristiti HTML tagove, pa nije moguće koristiti rešenja bez petlji**
  - Upotrebu JSP skripleta smo već ranije eliminisali
  - Upotreba JSTL u okviru petlji je opcija, ali i dalje složena
- **JSF omogućava h:dataTable**
  - Navodi se definicija za *jedan* red, i JSF ponavlja

```
<h:dataTable value="#{someBean.someCollection}"
  var="rowVar" border="1">
<h:column>
<h:outputText value="#{rowVar.col1Data}"/>
</h:column>
<h:column>
<h:outputText value="#{rowVar.col2Data}"/>
</h:column>
...
</h:dataTable>
```

h:dataTable

- **value: kolekcija podataka (obično lista bean-ova). Legalni tipovi kolekcija:**

– Array
– List (ArrayList, LinkedList)
– ResultSet
– Result (omotač ResultSet iz JSTL)
– DataModel (u javax.faces.model)

- **var: okružuje svaki ulaz kolekcije**
  - – Ovja ulaz treba da bude nešto što JSF EL može da prikaže
  - • Bean, Array, List, Map
- **Drugi atributi**
  - – Standardni TABLE atributi
  - • border, bgcolor, width, cellpadding, cellspacing, frame, ...
  - – Style sheet
  - • rowClasses, headerClass, footerClass

h:column

- **Obično okružuju h:outputText elemente**

```
<h:column>
   <h:outputText value="#{rowVar.colData}"/>
</h:column>
```

- **Mogu okruživati druge h:Xxxx elemente**
  - –h:inputText, ili bilo koje druge elemente
- **Regularni HTML sadržaj mora biti u okviru f:verbatim**

```
<h:column>
   <f:verbatim>First Name: </f:verbatim>
   <h:outputText value="#{rowVar.firstName}"/>
</h:column>
```

- **Heading i footer tabele se specificira pomoću f:facet**

Primer

- **Prikaz tabele prodaje baziranoj na podacima tipa Array**
- **Klasa SalesBean prezentuje prodaju jabuka i pomorandži**
- **Niz SalesBean objekata prezentuje kvartalnu prodaju u godini**
- **Sve vrednosti će se prikazati u okviru HTML tabele**

SalesBean

```
public class SalesBean {
  private double apples = 0.0, oranges = 0.0;

  public SalesBean() {}

  public SalesBean(double apples, double oranges) {
    setApples(apples);
    setOranges(oranges);
  }

  public double getApples() { return(apples); }

  public void setApples(double apples) {
    this.apples = apples;
  }

  public double getOranges() { return(oranges); }

  public void setOranges(double oranges) {
    this.oranges = oranges;
```

SalesBean

```
public SalesBean[] getYearlySales() {
SalesBean[] yearlySales =
{ new SalesBean(100.22, 200.32),
new SalesBean(300.44, 400.55),
new SalesBean(500.66, 600.77),
new SalesBean(700.88, 800.99) };
return(yearlySales);
}
```

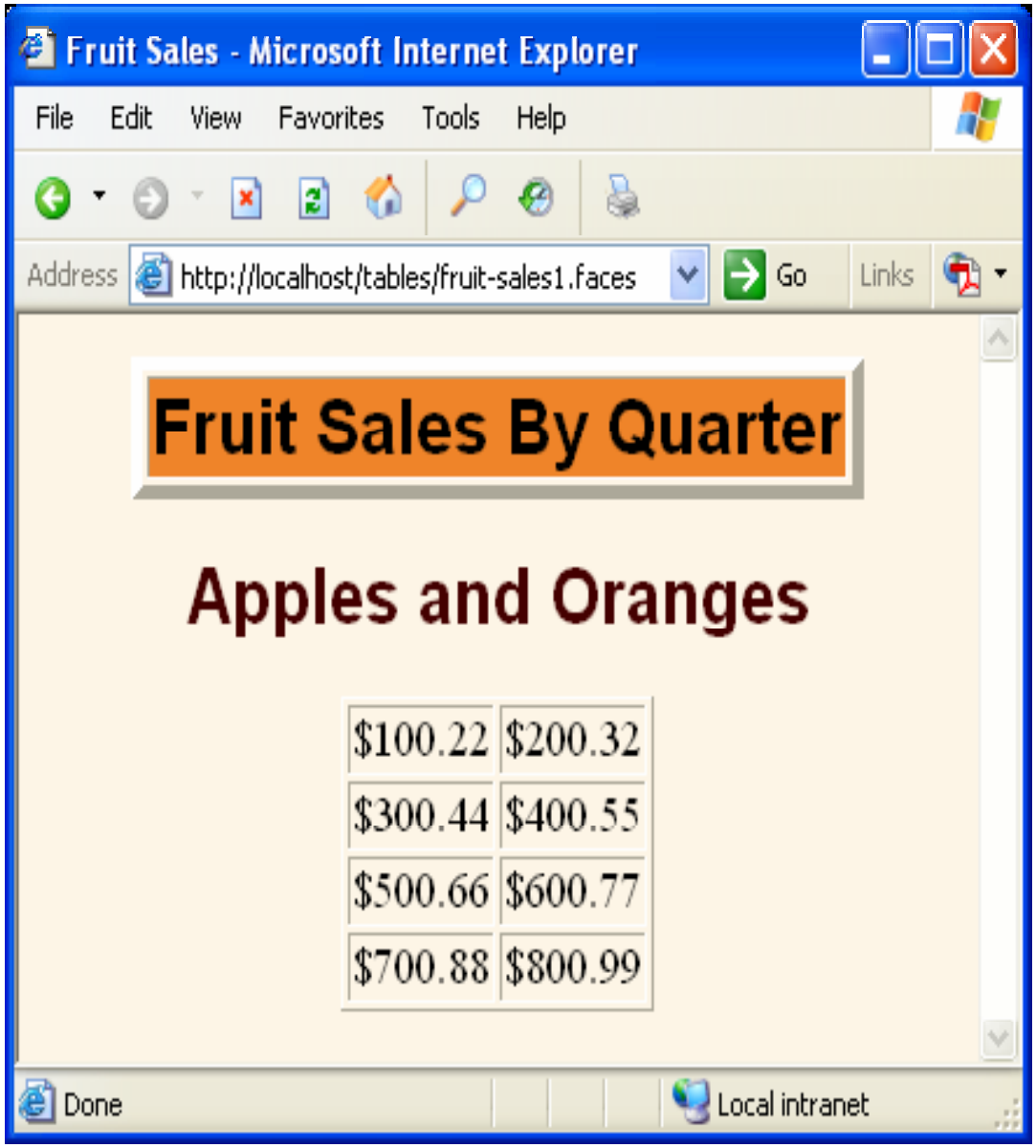
faces-config.xml

```
...
<faces-config>
<managed-bean>
<managed-bean-name>salesBean</managed-bean-
   name>
<managed-bean-class>
coreservlets.SalesBean
</managed-bean-class>
<managed-bean-scope>session</managed-bean-
   scope>
</managed-bean>
...
</faces-config>
```

Jsp strana

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
...
<H2>Apples and Oranges</H2>
<h:dataTable value="#{salesBean.yearlySales}‟ var="quarterlySales"
border="1">
<h:column>
<f:verbatim>$</f:verbatim>
<h:outputText value="#{quarterlySales.apples}"/>
</h:column>
<h:column>
<f:verbatim>$</f:verbatim>
<h:outputText value="#{quarterlySales.oranges}"/>
</h:column>
</h:dataTable>
...
</f:view>
```

Rezultat


Fruit Sales - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/tables/fruit-sales1.faces
Go
Links
Fruit Sales By Quarter
Apples and Oranges
$100.22 $200.32
$300.44 $400.55
$500.66 $600.77
$700.88 $800.99
Done
Local intranet

Header i footer

- **Header**
  - Koristi se f:facet sa name="header"
  - Vrednost može da bude f:verbatim ili h:outputText
  - I dalje je potrebno h:outputText za ne-heading vrednosti
- **Footer**
  - Koristi se f:facet sa name="footer"
  - Vrednost može da bude f:verbatim ili h:outputText
- **Primer koda**

```
<h:column>
    <f:facet name="header">
    <f:verbatim>...</f:verbatim>
    </f:facet>
<h:outputText value="#{rowVar.colVal}"/>
</h:column>
```

Primer: prikaz mesečne prodaje

- **Klasa MonthlySalesBean predstavlja ime meseca i SalesBean za prodaju jabuka (pomorandži) za taj mesec**
- **ArrayList predstavlja prodaju od Januara do Decembra**
- **Headings treba da budu "Month", "Apples", i "Oranges"**
- **Takođe treba koristiti tekst polja da bi se prikazali rezultati prodaje i da mogu da se promene**

– Primer ilustruje da tabele mogu da sadrže i ulazne elemente

– U primeru se ne prikazuje backend kod koji dovodi do rezultata

Primer: bean

```
public class MonthlySalesBean {
  private String month;
  private SalesBean salesBean;

  public MonthlySalesBean() {}

  public MonthlySalesBean(String month,
                          double apples,
                          double oranges) {
    setMonth(month);
    setSalesBean(new SalesBean(apples, oranges));
  }

  public String getMonth() { return(month); }

  public void setMonth(String month) {
    this.month = month; }

  public SalesBean getSalesBean() {
    return(salesBean);
  }
  public void setSalesBean(SalesBean salesBean) {
```

Primer: bean

```
public ArrayList getYearlySales() {
ArrayList yearlySales = new ArrayList();
yearlySales.add
(new MonthlySalesBean("January", 101.11, 501.11));
yearlySales.add
(new MonthlySalesBean("February", 102.22, 502.22));
yearlySales.add
(new MonthlySalesBean("March", 103.33, 503.33));
...
yearlySales.add
(new MonthlySalesBean("November", 111.11, 511.11));
yearlySales.add
(new MonthlySalesBean("December", 112.22, 512.22));
return(yearlySales);
}
```

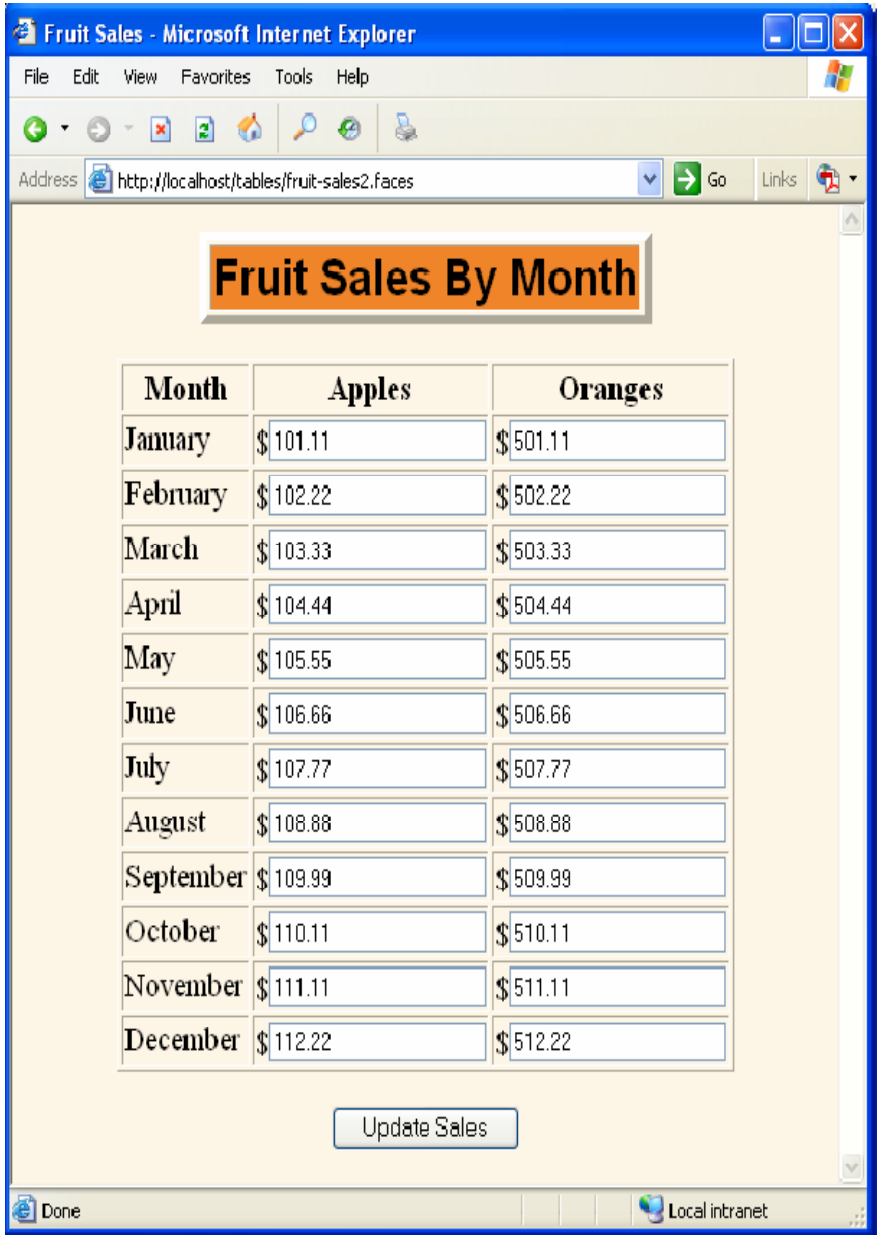
faces-config.xml

```
...
<faces-config>
<managed-bean>
<managed-bean-name>
monthlySalesBean
</managed-bean-name>
<managed-bean-class>
coreservlets.MonthlySalesBean
</managed-bean-class>
<managed-bean-scope>session</managed-bean-
   scope>
</managed-bean>
...
</faces-config>
```

Jsp strana

```
<h:form>
<h:dataTable value="#{monthlySalesBean.yearlySales}"
var="monthlySales‟ border="1">
<h:column>
<f:facet name="header">
<f:verbatim>Month</f:verbatim>
</f:facet>
<h:outputText value="#{monthlySales.month}"/>
</h:column>
<h:column>
<f:facet name="header">
<f:verbatim>Apples</f:verbatim>
</f:facet>
<f:verbatim>$</f:verbatim>
<h:inputText value="#{monthlySales.salesBean.apples}"/>
</h:column>
... Oranges ...
</h:dataTable>
```

Rezultat


Fruit Sales - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/tables/fruit-sales2.faces
Go
Links
Fruit Sales By Month
Month
Apples
Oranges
January $101.11 $501.11
February $102.22 $502.22
March $103.33 $503.33
April $104.44 $504.44
May $105.55 $505.55
June $106.66 $506.66
July $107.77 $507.77
August $108.88 $508.88
September $109.99 $509.99
October $110.11 $510.11
November $111.11 $511.11
December $112.22 $512.22
Update Sales
Done
Local intranet

CSS

```
.HEADING {
font-family: Arial, Helvetica, sans-serif;
font-weight: bold;
font-size: 20px;
color: black;
background-color: silver;
text-align: center;
}
```

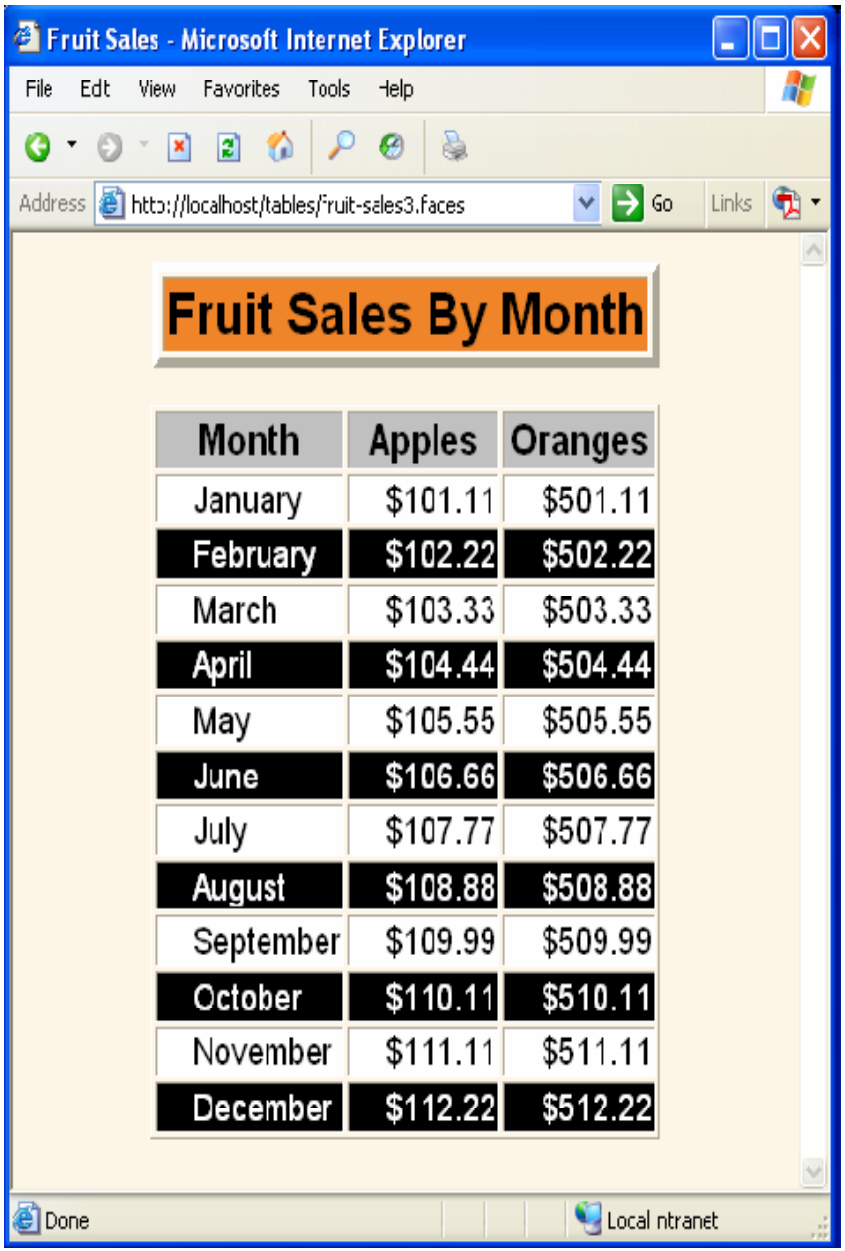
CSS

```
.ROW1 {
font-family: Arial, Helvetica, sans-serif;
font-size: 18px;
color: black;
background-color: white;
text-indent: 20px;
}
.ROW2 {
font-family: Arial, Helvetica, sans-serif;
font-size: 18px;
color: white;
background-color: black;
text-indent: 20px;
}
```

JSP

```
<h:dataTable value="#{monthlySalesBean.yearlySales}"
var="monthlySales‟ border="1"
headerClass="HEADING‟ rowClasses="ROW1,ROW2">
<h:column>
<f:facet name=‟header">
<f:verbatim>Month</f:verbatim>
</f:facet>
<h:outputText value="#{monthlySales.month}"/>
</h:column>
<h:column>
... Apples ...
</h:column>
<h:column>
... Oranges ...
</h:column>
</h:dataTable>
```

Rezultat


Fruit Sales - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/tables/fruit-sales3.faces
Go
Links
Fruit Sales By Month
Month
Apples
Oranges
January $101.11 $501.11
February $102.22 $502.22
March $103.33 $503.33
April $104.44 $504.44
May $105.55 $505.55
June $106.66 $506.66
July $107.77 $507.77
August $108.88 $508.88
September $109.99 $509.99
October $110.11 $510.11
November $111.11 $511.11
December $112.22 $512.22
Done
Local ntranet