Praćenje sesije

HTTP komunikacija

- HTTP je stateless protokol koji ne zateva od servera čuvanje statusa klijenta ili korisničke sesije klijenta tj. niza zahteva upućenih od strane istog klijenta
  - HTTP serveri prevazilaze prethodno tako što implementiraju različite metode za održavanje i upravljanje sesijom, tipično se oslanjajući na jedinstveni identifikator *cookie* ili neki drugi parametar koji omogućava praćenje zahteva koji originiraju od istog klijenta (npr. URL Rewriting mehanizam), kreirajući stateful protokol iznad HTTP protokola.

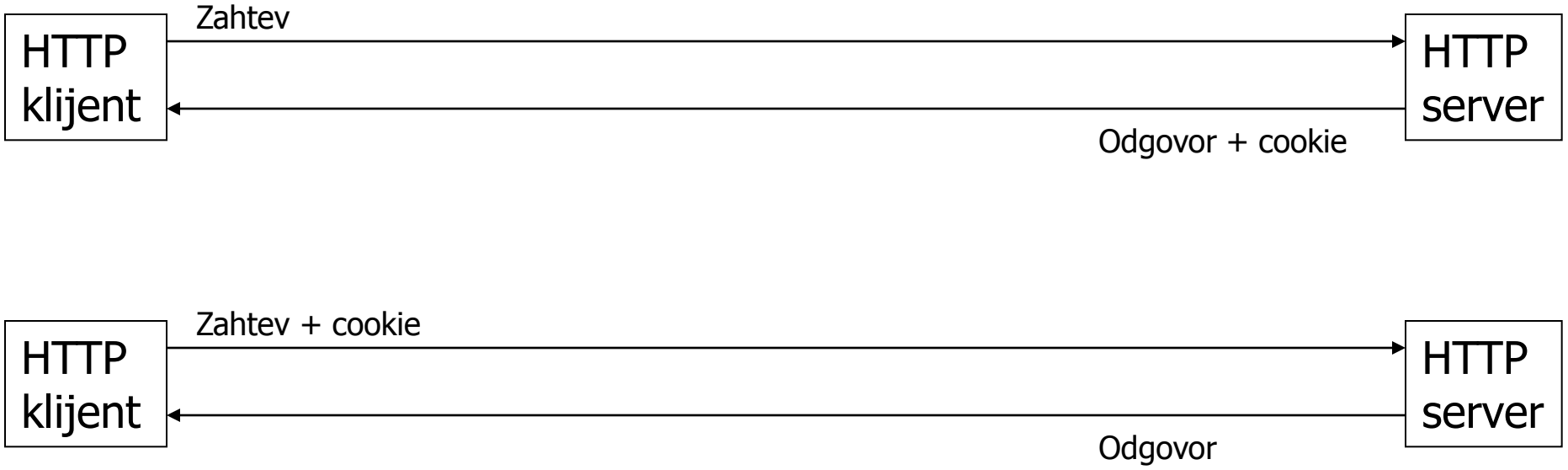
Praćenje sesije korisnika 1/3

- HTTP protokol ne prati sesiju – veza klijenta i servera se zatvara po isporuci resursa.
- Koristi se *cookie* mehanizam:
  - Server šalje *cookie* klijentu u okviru http response
  - klijent čuva primljeni *cookie* i šalje ga uz svaki http request.
- Ako navigator ne prihvata *cookie-je*, koristi se URL Rewriting mehanizam:

```
<a href="http://www.myserver.com/catalog/index.html;jsessionid=1234">
link
</a>
```

Praćenje sesije korisnika 2/3

- *cookie* mehanizam



Praćenje sesije korisnika 3/3

```
GET / HTTP/1.1
Accept-Language: sr
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
Host: localhost
Connection: Keep-Alive
```

```
HTTP/1.0 200 OK
Date: Tue, 04 May 02004 08:55:09 GMT
Status: 200
Content-Type: text/html
Content-Length: 2524
Content-Language: en
Set-Cookie: ASPSESSIONIDGGQQQAEK=JKKPPFKCMNDNMEEOHAADJKPM

<html><head></head>
<body></body>
</html>
```

```
GET /Test HTTP/1.1
Accept-Language: sr
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
Host: localhost
Connection: Keep-Alive
Cookie: ASPSESSIONIDGGQQQAEK=JKKPPFKCMNDNMEEOHAADJKPM
```

Detaljnije o Cookie-ima i HTTP odgovoru

- Vraća se u atributu Set-Cookie
- Opcioni atributi
  - domain – domen u kome važi cookie
  - path – za koje URL-ove na sajtu važi
  - expires – datum isticanja
- Kada ističe cookie?
  - eksplicitan momenat (expires)
  - nedefinisano – kada se ugasi browser

Primer

HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html
Set-Cookie:
PREF=ID=3ff1557ca378ed16:TM=1151585607:LM=115
1585607:S=rVfoQD15sUOZajEt; expires=Sun, 17-Jan-
2038 19:14:07 GMT; path=/; domain=.google.com
Content-Encoding: gzip
Server: GWS/2.1
Content-Length: 1349
Date: Thu, 29 Jun 2006 12:53:27 GMT

Browseri

- Svaki browser ima svoje kukije i ne deli ih sa drugim browserima
  - za svaki sajt postoji poseban kuki
- Tabovi i stranice u browserima dele kukije

Alternativan način

- Ako navigator ne prihvata *cookie-je*, koristi se URL Rewriting mehanizam
- u hiperlink (<a href="...">) koji "gađa" naš server ugradimo id sesije:

<a href=http://www.myserver.com/catalog/index.html;jsessionid=1234>

- HTTPServletResponse.encodeURL() metoda
- HTTPServletResponse.encodeRedirectURL() metoda
- Da li uvek možemo da koristimo ovu tehniku?
- Gde se stavlja id sesije u formi?

Klasa HttpSession

- Reprezentuje sesiju
- Čuva cookie ili ID sesije za URL redirection
  - metoda getId()
- Čuva objekte vezane za sesiju
  - metode getAttribute(ime), setAttribute(ime, objekat), removeAttribute(ime)
- Invalidira sesiju i razvezuje sve objekte vezane za nju
  - metoda invalidate()
- podešava period neaktivnosti
  - metoda setMaxInactiveInterval(sekunde)

Praćenje sesije korisnika iz servleta 1/3

- Praćenje sesije se svodi na kreiranje objekata koji se vezuju na sesiju
  - objekat bi trebalo da je serijalizabilan
- Kada korisnik prozove neki servlet, u okviru njega se koristi objekat vezan za sesiju:
  - ako do tada nije postojao objekat, on se kreira i veže za sesiju;
  - ako postoji, koristi se.
- Session inactivity – obično 30 minuta
  - podešava se u web.xml
  - session.setMaxInactiveInterval

Praćenje sesije korisnika iz servleta 2/3

```
public void doGet(HttpServletRequest request, HttpServletResponse
response) throws java.io.IOException {
  ...
  HttpSession session = request.getSession();
  SessionCounter sc =  (SessionConter)session.getAttribute(
                      "brojac");
   if (sc != null) {
     sc.inc();
     out.println(", ukupno pristupa:" + sc.getCount() + ".<br>");
   } else {
     out.println(", prvi pristup.<br>");
     sc = new SessionCounter();
     sc.inc();
     session.setAttribute("brojac", sc);
   }

  ...

}
```

Klasa SessionCounter 3/3

```
class SessionCounter {

  private int count = 0;

  public int getCount() {
    return count;
  }

  public void setCount(int c) {
    count = c;
  }

  public void inc() {
    count++;
  }
}
```

Na šta se objekat može vezati?

- Na request:

```
request.setAttribute("ime", referenca);
```

- Na sesiju:

```
request.getSession().setAttribute("ime",
   referenca);
```

- Na aplikaciju:

```
getServletContext().setAttribute("ime",
   referenca);
```

Kako saznati gde se na serveru nalazi folder naše web aplikacije?

- getServletContext().getRealPath("");

Primer

- Importujte projekat *WebProgOrg* u Eclipse
- Napravite i eksportujte war u Apache Tomcat
- Porenite Server
- U web browser kucajte
- http://localhost:8080/WebProgOrg/WebShopServlet

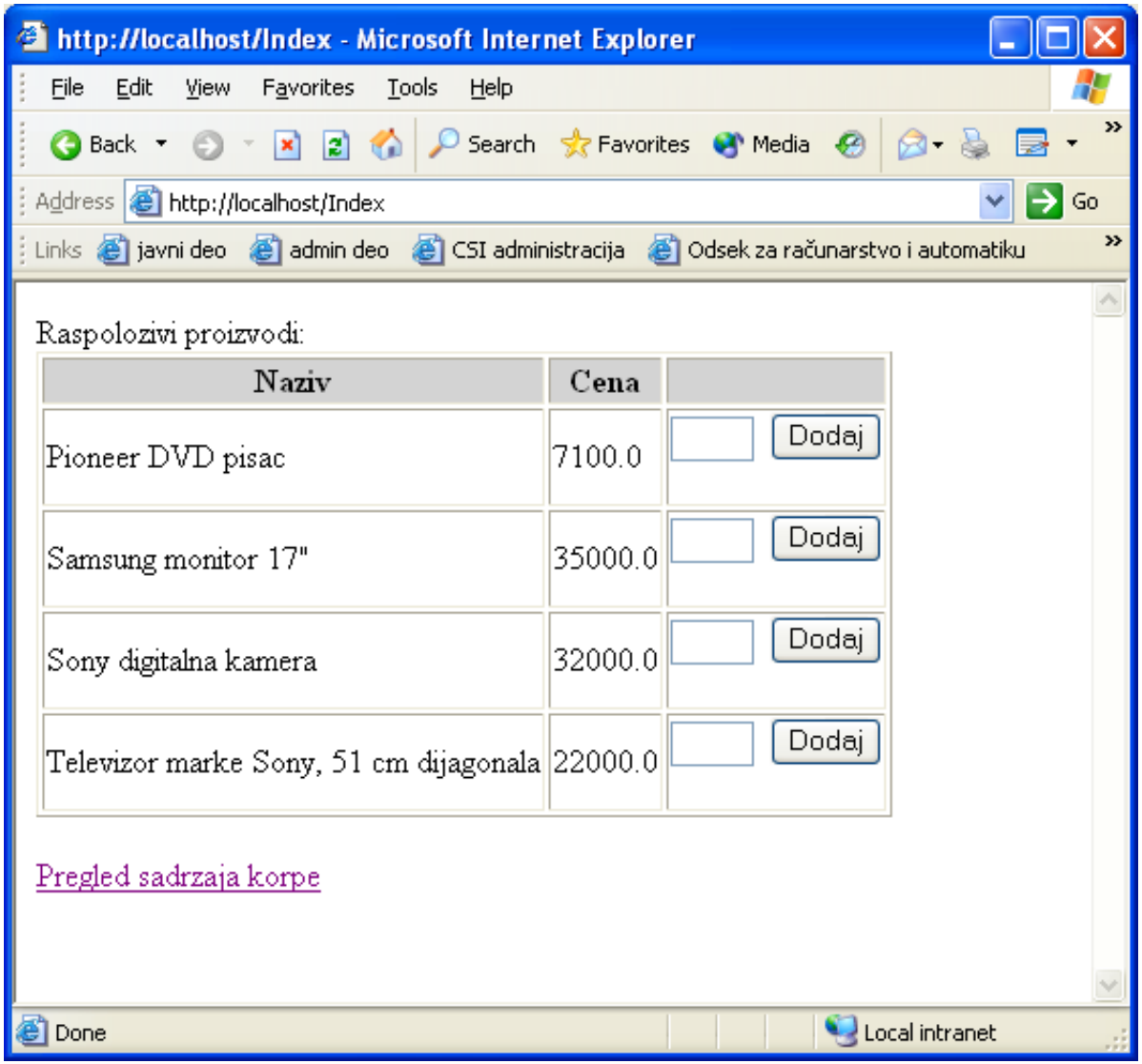
Case study – web shop

- Dve stranice web shop-a:
  - pregled svih raspoloživih proizvoda
  - dodavanje na spisak ili pregled spiska proizvoda odabranih za kupovinu

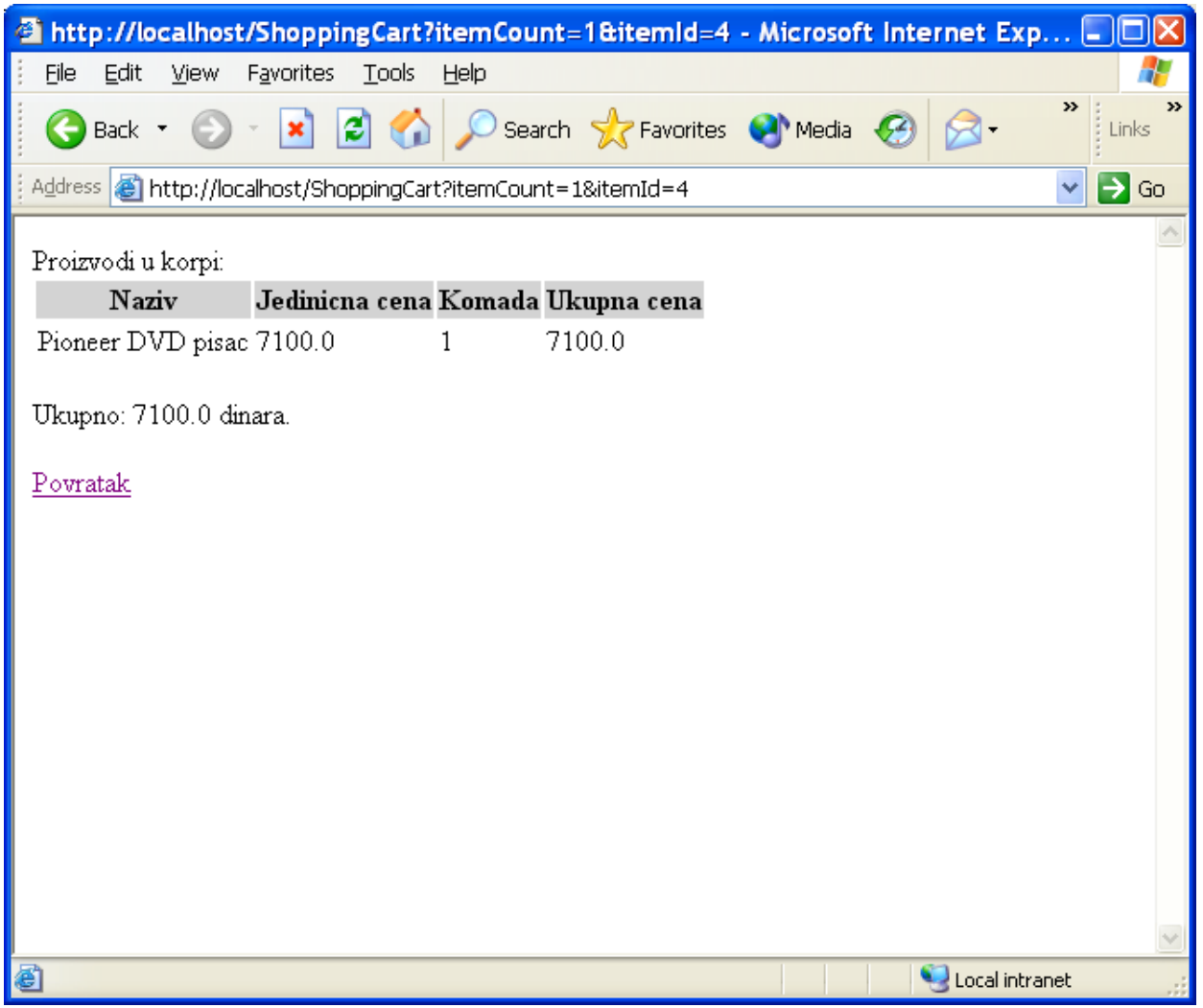
Case study – web shop

- Uzastopnim ponavljanjem dodaj nekoliko različitih proizvoda u korpu
- Klikni na link pregled *Pregled sadrzaja korpe*
- Ugasi web browser i pokreni ga opet
- Klikni na link pregled *Pregled sadrzaja korpe,* da li ima proizvoda u korpi?
- Otvori drugi web browser i klikni na link pregled *Pregled sadrzaja korpe,* da li ima proizvoda u korpi?

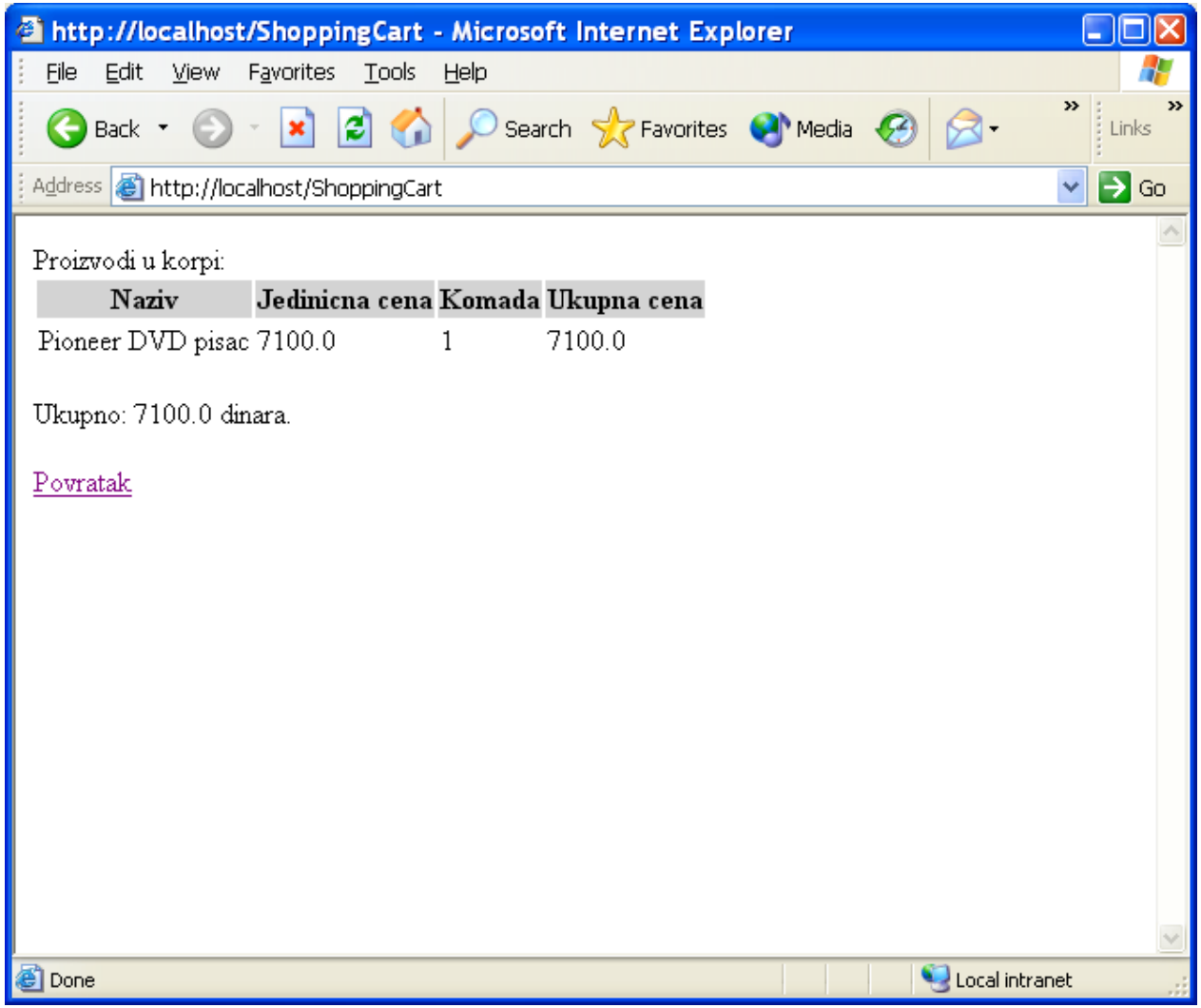
Pregled proizvoda



Dodavanje proizvoda na listu za kupovinu (shopping cart)



Pregled liste za kupovinu



Case study – web shop

- Pogledati strukturu projekta
  - *servlet1* paket sadrži servlete
  - *servlet1.webshop* paket klase koje predstavljaju model podataka
- Otvorite kod servleta *WebShopServlet* i otvorite source kod HTML stranice u web browser
- Koja HTTP metoda se poziva get ili post?
  - pogledaj *doGet*
  - pogledaj *init*

Case study – web shop

- Otvori klasu Products
  - konstruktor
- Otvori klasu product.txt
- Razlika između konzolne aplikacije i web aplikacije ako se posmatra putanja do fajla?
- Gde se ispisuje syso u konstruktoru? Koja je to putanja?

Case study – web shop

- Da li kod nekog ko koristi **deploy** iz Eclipse piše nešto nalik **Modul2Web/.metadata/.plugins/…**
- Zašto postoji razlika u putanji u odnosu na ostale?
- Ako postoji razlika
- Metoda *readProducts*
  - # započinje komentar u products.txt pa preskoči red

Case study – web shop

- Povratak u metodu *init WebShopServlet*
  - Zašto je bilo bolje kreirati klasu *Products* umesto da se koristi HashMap za smeštanje objekata?
- Metoda servleta bi teoretski mogla da bude *readProduct* ali se to izbegava jer servlet treba da ima samo metode koje rade sa HTTP protokolom
  - dekompozicija problema
- U web browseru dodaj proizvod u korpu
  - sadržaj URL je

Lista raspoloživih proizvoda (WebShopServlet)

```
ServletContext ctx = getServletContext();
Products products = new Products(ctx.getRealPath(""));
ctx.setAttribute("products", products);
...
pout.println("Raspolozivi proizvodi:");
pout.println("<table border=\"1\"><tr
   bgcolor=\"lightgrey\"><th>Naziv</th><th>Cena</th><th> </th></tr>");
for (Product p : products.values()) {
  pout.println("<tr>");
  pout.println("<form method=\"get\" action=\"ShoppingCartServlet\">");
  pout.println("<td>" + p.getName() + "</td>");
  pout.println("<td>" + p.getPrice() + "</td>");
  pout.println("<td>");
  pout.println("<input type=\"text\" size=\"3\" name=\"itemCount\">");
  pout.println("<input type=\"hidden\" name=\"itemId\" value=\""
                          + p.getId() + "\">");
  pout.println("<input type=\"submit\" value=\"Dodaj\">");
  pout.println("</form>");
  pout.println("</td>");
  pout.println("</tr>");
}
pout.println("</table>");
```

Datoteka products.txt

1;Televizor marke Sony, 51 cm dijagonala;22000
2;Sony digitalna kamera;32000
3;Samsung monitor 17";35000
4;Pioneer DVD pisac;7100

Paket webshop – klasa Products 1/4

- Reprezentuje spisak proizvoda
- Iz datoteke products.txt učitava spisak proizvoda i smešta u asocijativnu listu (ključ je id proizvoda, a objekat koji se smešta u listu je klase *Product*)

Paket webshop – klasa Products 2/4

```
public Products(String path) {
  BufferedReader in = null;
  try {
    in = new BufferedReader(new FileReader(
                              path + "/products.txt"));
    readProducts(in);
    in.close();
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

Paket webshop – klasa Products 3/4

```
private void readProducts(BufferedReader in) {
  String line, id = "", name = "", price = "";
  StringTokenizer st;
  int pos;
  try {
    while ((line = in.readLine()) != null) {
      line = line.trim();
      if (line.equals("") || line.indexOf('#') == 0)
        continue;
      st = new StringTokenizer(line, ";");
      while (st.hasMoreTokens()) {
        id = st.nextToken().trim();
        name = st.nextToken().trim();
        price = st.nextToken().trim();
      }
      products.put(id, new Product(id, name,
   Double.parseDouble(price)));
    }
  } catch (Exception ex) {
    ex.printStackTrace();
  }
}
```

Paket webshop – klasa Products 4/4

```
private HashMap<String, Product> products =
new HashMap<String, Product>();

/** Vraca kolekciju proizvoda. */
public Collection<Product> values() {
    return products.values();
}

/** Vraca proizvod na osnovu njegovog id-a.
*/
public Product getProduct(String id) {
    return products.get(id);
}
```

Paket webshop – klasa Product 1/2

- Reprezentuje pojedinačan proizvod
- Karakterišu je:
  - id,
  - naziv i
  - cena.

Paket webshop – klasa Product$_{2/2}$

```
public class Product {
  private String id;
  public void setId(String i) {
    id = i;
  }
  public String getId() {
    return id;
  }
  private String name;
  public void setName(String n) {
    name = n;
  }
  public String getName() {
    return name;
  }
  private double price;
  public void setPrice(double p) {
    price = p;
  }
  public double getPrice() {
    return price;
  }
  public Product(String id, String name, double price) {
    this.id = id;
    this.name = name;
    this.price = price;
  }
}
```

Case study – web shop

- Otvorite kod servleta *ShoppingCartServlet* i metoda *doGet,* idite na deo kreiranja tabele u HTML
- Otvorite kod klase *ShoppingCart*
- Otvorite kod klase *ShoppingCartItem*
- Pogledajte kod punjenje korpe sa proizvodima
- Pogledajte kod koji radi sa sesijom sesije
- Svaki browser ima svoju memoriju za sesiju
- U dva browsera otvori developer tools, network kartica, pokreni dodavanje produkta u sesiju
  - Da li je cookie vredost ista?

Dodavanje/pregled proizvoda u listu za kupovinu (ShoppingCartServlet)

```
HttpSession session = request.getSession();
ShoppingCart sc =  (ShoppingCart)session.getAttribute("ShoppingCart");
if (sc == null) {
  // ako ne postoji, kreiramo ga i dodelimo tekucoj sesiji.
  // Na ovaj nacin, objekat klase ShoppingCart ce pratiti sesiju.
  sc = new ShoppingCart();
  session.setAttribute("ShoppingCart", sc);
}
PrintWriter pout = response.getWriter();
response.setContentType("text/html");
if (request.getParameter("itemId") != null) {
  // ako smo pozvali ovaj servlet sa parametrima za dodavanje proizvoda u korpu
  try {
    Products products = (Products)getServletContext().getAttribute("products");
    // probamo da ga dodamo
    sc.addItem(
            products.getProduct(request.getParameter("itemId")),
            Integer.parseInt(request.getParameter("itemCount")));
    } catch (Exception ex) {
      ex.printStackTrace();
    }
}
pout.println("Proizvodi u korpi:");
pout.println("<table><tr bgcolor=\"lightgrey\"><th>Naziv</th><th>Jedinicna
  cena</th><th>Komada</th><th>Ukupna cena</th></tr>");
double total = 0;
for (ShoppingCartItem i : sc.getItems()) {
    pout.println("<tr>");
    pout.println("<td>" + i.getProduct().getName() + "</td>");
    ...
}
```

Paket webshop – klasa ShoppingCart 1/2

- Reprezentuje listu proizvoda za kupovinu (korpu, shoppping cart)
- Sadrži listu odabranih proizvoda za kupovinu (svaki odabrani proizvod je reprezentovan objektom klase *ShoppingCartItem*)

Paket webshop – klasa ShoppingCart 2/2

```
public class ShoppingCart {
   ArrayList <ShoppingCartItem> items;

  public ShoppingCart() {
    items = new ArrayList <ShoppingCartItem>();
  }

  public void addItem(Product product, int count) {
    items.add(new ShoppingCartItem(product, count));
  }

  public ArrayList <ShoppingCartItem> getItems() {
    return items;
  }
}
```

Paket webshop – klasa ShoppingCartItem 1/2

- Reprezentuje jednu stavku iz spiska proizvoda koje je mušterija odabrala za kupovinu
- Čuva:
  - referencu na proizvod (referencu na objekat klase *Product*),
  - količinu.

Paket webshop – klasa ShoppingCartItem 2/2

```
public class ShoppingCartItem {

 private Product product;
  public void setProduct(Product p) {
    product = p;
  }
  public Product getProduct() {
    return product;
  }

  private int count;
  public void setCount(int c) {
    count = c;
  }
  public int getCount() {
    return count;
  }

  public ShoppingCartItem(Product p, int count) {
    this.product = p;
    this.count = count;
  }
}
```

Prijava na sistem

- Klasa User:
  - reprezentuje korisnika
  - vezuje se na sesiju
    - prilikom prijavljivanja na sistem
  - sadrži Shopping Cart kao atribut
  - sadrži informaciju o tome da li je korisnik prijavljen

Prijava na sistem

- Modelujemo klasu *User* koja sadrži String *username*, String *password*, ShoppingCart *sc*; boolean *loggedIn*. Napiši toString i equals
- Kreira se HTML stranica za logovanje *login.html* koja sadrži formu za unosa korisničkog *username* i *password*. Na dugme pošlji gađa se servlet *LogIn*

Prijava na sistem

- Kreirati servlet *LogIn*
- U *init* metodi kreirati listu *users ,* i dodati je u *ServletContext*

```
users.add(new User("pera", "peric"));
users.add(new User("steva", "stevic"));
users.add(new User("jova", "jovic"));
users.add(new User("mitar", "miric"));
users.add(new User("peka", "peka"));
```

Prijava na sistem

- U doGet metodi
  - *Preuzmi parametre forme username* i *password*
  - *Preuzmi iz ServletContext listu users*
  - *Proveri da li takav user postoji u listi users*
    - *Ako postoji dodaj u sesiju user i redirektuj na servlet za kupovinu WebShopServlet*
    - *Ako ne postoji redirektuj na login.html*
- Izmeni ostale servlete tako da ne može im se pristupiti ako korisnik nije ulogovan
- Izmeni *ShoppingCartServlet* tako da se korpa koristi iz klase *User*

Prijava na sistem

- Sada vi pokušajte da uradite to
- Krenite od prvog slajda Prijava na sistem i radite korak po korak