Orbit Hotel | 8H2

SEBASTIAN CHLUP
KORBINIAN CHRISTL
MICHAEL LAZARUS
VERONIKA LOMASOW

Reflexion des Projekts (1)

Orbit Hotel

- Bietet einem Hotelunternehmen die Möglichkeit, Zimmerübernachtungen an Kunden zu verkaufen
- Hotelmitarbeiter (Hotelier) offerieren neue Zimmer
- Hoteliers sehen Zimmerbewertungen

- Bietet Kunden die Möglichkeit nach Zimmern zu suchen
- Können von Kunden gebucht werden
- Kunden können Zimmer bewerten

Reflexion des Projekts (2)

- „Analysten“ → Für die Buchhaltung und die Preisgestaltung
- Analysten sehen Bewertungen
- Unterstützen Hotelier in der optimalen Preisgestaltung
- Können 3 verschiedene Statistiken erfassen

ServletKlasse - MasterServlet

- Von JSPs aufgerufen
- Erstellt Session
- Holt sich Parameter aus JSP
- Ruft Methoden aus Hotelmanagement mit Parametern auf
- SendRedirect

Verwaltungsklasse Hotelmanagement

- Schnittstelle zwischen Servlet und Programm
- Beinhaltet wichtigste Verwaltungsmethoden
- Erhält von MasterServlet Methodenaufrufe mit Parametern
- Speichert und liest aus DAO-Klassen

Rate Booking


```
else if(rcv.equals("RateBooking")){
    String type = (String)session.getAttribute("person");
    Person person = PersonDAO.getPersonbyUsername(type);
    if(person instanceof Customer){
    String xbookingnumber = request.getParameter("bookingnumber");
    String rating = request.getParameter("ratearea");
    int bookingnumber = Integer.parseInt(xbookingnumber);
    check = x.RateBooking(bookingnumber, rating);
    }
    else{
        check = "index.jsp";
    }
}
```

```
    public String RateBooking(int bookingnumber , String rating){
        System.out.println(bookingnumber + rating);
//      if(session instanceof Customer){
            int x = bookingDAO.getRoombyBookingnumber(bookingnumber);
            if(x != 0){
            System.out.println(x);
            Room newRoom = roomDAO.getRoombyRoomnumber(x);
            newRoom.setRating(rating);
            roomDAO.updateRoom(newRoom);
            return "CustomerInterfaceSucess.jsp";
            }
            else{
                return "CustomerInterfaceFail.jsp";
            }
        }
```

Season Statistic

```
else if(rcv.equals("SeasonStatistic")){
    String type = (String)session.getAttribute("person");
    Person person = PersonDAO.getPersonbyUsername(type);
    if(person instanceof Analyst){
    String xyear = request.getParameter("year");
    int year = Integer.parseInt(xyear);
    sstat = x.SeasonStatistic(year);
    check = "AnalystSeason.jsp";
    }
    else{
        check = "index.jsp";
    }
}
```

```
public int [][]SeasonStatistic(int refyear){
    //if(session instanceof Analyst){
            bookinglist = bookingDAO.getBookinglist();

            int [][] season= new int [2][12];
            System.out.println("test");

            for(Booking booking : bookinglist){
                Date start = booking.getBookingstart();
                Calendar timeref = Calendar.getInstance();
                timeref.setTime(start);
                int year = timeref.get(Calendar.YEAR);
                int month = timeref.get(Calendar.MONTH) +1;

                System.out.println(month);
                if(refyear == year){
                    if(month == 1){
                        season[0][0] += 1;
                        season[1][0] += booking.getBnop();
                    }
                    else if(month == 2){
                        season[0][1] += 1;
                        season[1][1] += booking.getBnop();
                    }
                    else if(month == 3){
                        season[0][2] += 1;
                        season[1][2] += booking.getBnop();
                    }
                    else if(month == 4){
                        season[0][3] += 1;
                        season[1][3] += booking.getBnop();
                    }
                    else if(month == 5){
                        season[0][4] += 1;
                        season[1][4] += booking.getBnop();
```

New Room

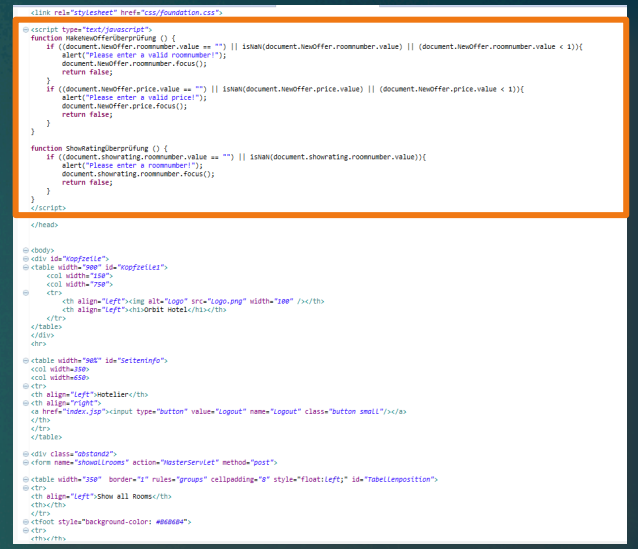
```
else if(rcv.equals("Create")){
    String type = (String)session.getAttribute("person");
    Person person = PersonDAO.getPersonbyUsername(type);
    if(person instanceof Hotelier){
    String xroomnumber = request.getParameter("roomnumber");
    String xnop = request.getParameter("nop");
    String equipment = request.getParameter("equipment"); // muss nicht konvertiert werden
    String xprice = request.getParameter("price");
    int roomnumber = Integer.parseInt(xroomnumber);
    int nop = Integer.parseInt(xnop);
    double price = Double.parseDouble(xprice);
    String rating ="No rating available";
    check = x.NewOffer(roomnumber, nop, equipment, price, rating);
    }
    else{
        check = "index.jsp";
    }
}
```

```
    public String NewOffer(int roomnumber, int nop, String equipment, double price, String rating){
//      if(session instanceof Hotelier){
            Room newRoom =  new Room(roomnumber, nop, equipment, price, rating);
            String check = roomDAO.saveRoom(newRoom);

            if(check.equals("success"))
                return "HotelierInterfaceNewOffer.jsp";
            else
                return "HotelierInterfaceFail.jsp";
        }
```

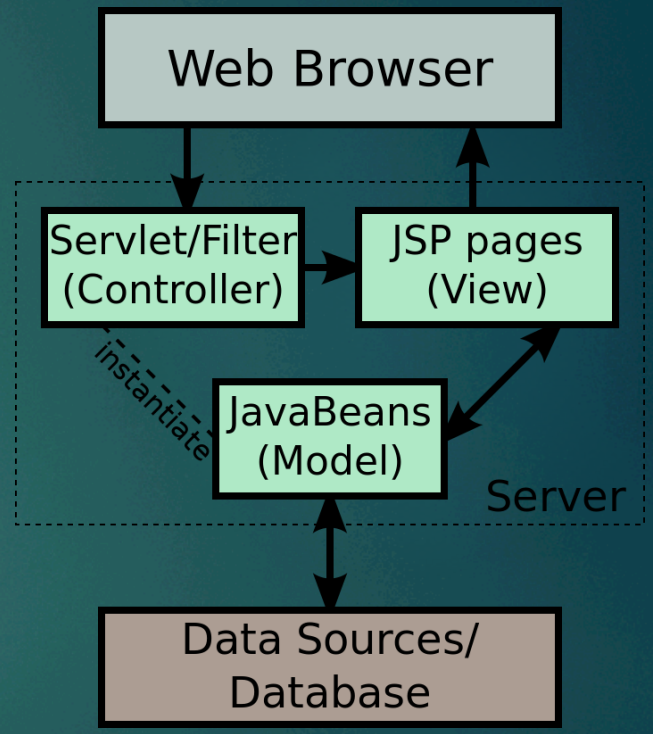
Java Script Implementierung

- Zu Beginn der JSP Seiten
- Formulareingaben werden validiert, bevor sie an das Servlet gesendet werden
- Erstellung einer Funktion (z.B. RegisterÜberprüfung())
  - Überprüfung mittels if() Statements
  - Bei „return false“ erfolgt keine Weiterleitung → Alert-Fenster
  - Bei „return true“ Weiterleitung erfolgt
- Ziel: Entlastung des Servlets durch zu viele unsinnige Eingaben



JSP JavaServer Pages

- Verwendung von HTML- und Java-Code zur Webserver-Ausgabe
- „Verlinkung“ zum Servlet durch form action=„ServletName“
- „Verlinkung“ zu einer anderen JSP durch form action=„Name.jsp“
- form method=„post“: Formular wird mit der Post-Methode übertragen
- CSS Cascading Style Sheets



Bildquelle: http://upload.wikimedia.org/wikipedia/commons/7/72/JSP_Model_2.svg

Komplexität

- Lines of Code
  - Java Resources (inkl. Javadoc) ~2500 LoC
  - JSP Resources (inkl. CSS & JavaScript) ~3650 LoC
- Klassenanzahl 12
- Package Struktur
  1. Package: hotel2013.hm
  2. Package: hotel2013.hm.dao
  3. Package: hotel2013.hm.data
  4. Package: hotel2013.hm.users