Talieh Dastmalchi

Car Rental System. Internet technology project of third year in Information technology program at heriot-watt university - March 2011

Car Rental System

Tutor: Mr Adel Ramezani

Contents

[Introduction 3](#_Toc288293012)

[System use cases 4](#_Toc288293013)

[System prototype 5](#_Toc288293014)

[Registration page 5](#_Toc288293015)

[Renting page 5](#_Toc288293016)

[Cars List 6](#_Toc288293017)

[Login Page 7](#_Toc288293018)

[Administrative Control Panel 7](#_Toc288293019)

[Add a category 8](#_Toc288293020)

[Add a car 9](#_Toc288293021)

[Activation of users 9](#_Toc288293022)

[Renting a car 10](#_Toc288293023)

[Successful rent 11](#_Toc288293024)

[Unsuccessful rent 11](#_Toc288293025)

[Logout 12](#_Toc288293026)

[Technical info of system 12](#_Toc288293027)

[Data layer 12](#_Toc288293028)

[User 12](#_Toc288293029)

[Product 12](#_Toc288293030)

[renting 12](#_Toc288293031)

[category 13](#_Toc288293032)

[Business layer 13](#_Toc288293033)

[Common layer 27](#_Toc288293034)

[UserENT 27](#_Toc288293035)

[RentENT 30](#_Toc288293036)

[CategoryENT 32](#_Toc288293037)

[CarENT 32](#_Toc288293038)

[Web layer 34](#_Toc288293039)

[confirmUser 35](#_Toc288293040)

[InsertCar 35](#_Toc288293041)

[InsertCategory 36](#_Toc288293042)

[Login 36](#_Toc288293043)

[Logout 37](#_Toc288293044)

[Register 37](#_Toc288293045)

[rentACar 38](#_Toc288293046)

[searchForCar 39](#_Toc288293047)

[searchForCategory 40](#_Toc288293048)

[userList 40](#_Toc288293049)

[JSPS 40](#_Toc288293050)

[RMI layer 43](#_Toc288293051)

[SaleRMIServer 43](#_Toc288293052)

[SaleRMIInterface 47](#_Toc288293053)

[SaleRMIClinet 48](#_Toc288293054)

[NET Layer 49](#_Toc288293055)

[Net Server 49](#_Toc288293056)

[Net thread server 53](#_Toc288293057)

[Data Layer embedded codes 57](#_Toc288293058)

# Introduction

These days by development the internet technology people are transferring their information systems or other computer based application to the web based mode. Web based systems are available from everywhere only by accessing to the internet.

Car Rental System designed and implements to does such tasks and save all information for a car rental company. To show the application of system in the society we can define several scenarios, many of them listed below.

1. scenario #1:

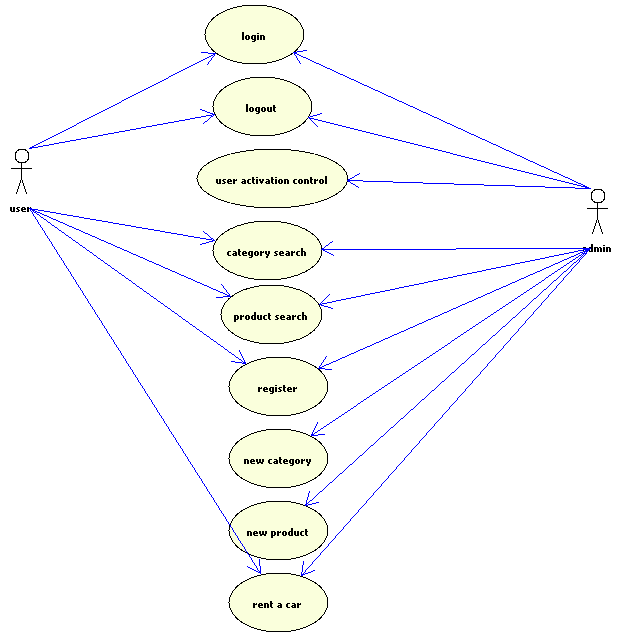
Joe is a tourist. He came to London yesterday from Tokyo. Instead of paying money to public vehicles for his journeys in the city, he wants to rent a Japanese car which is trustful in his mind. He can find us through a search engine for instance. After registering in the web site, now he can find the Toyota company for instance easily and rent his favorable car from the under the Toyota category cars.

1. Scenario #2:

Adams is the sale manager of a car rental company. People send their used car information to him daily by email or fax. Adams can add their car information to the exist list of cars which is available to rent in the system. For instance he can add a 2005, 3 series BMW under the BMW category. From now this car is ready to rent on the internet.

This document will have an overview over all technical information which implemented in development phase of the car rental system.

# System use cases



# System prototype

## Registration page

At first, before users start to rent a car in the system, they should register in it. The process started by completing and submitting the form which is shown in the below.



## Renting page

Through this page users can search and visit all available cars producer Company. They can either search between them. By clicking on each of rows the users redirected to the sub-category cars. For instance if they click on Toyota in the list, they will visit a list of all available cars which produced by it. In this step people cannot rent a car.



## Cars List

In this page which contains all available cars of a specific company, users can search through all available items to find the suitable choice.



## Login Page

To use the system capabilities all users should login first. They can do this by completing the exist form in this page. If a problem occurred during the login process a message will present to them else the welcome sentence presents.

Based on the users authority if admin of the system logged in, then it will redirected to the administrative control panel.



## Administrative Control Panel

By logging the admin into system the below page will show. As is clear admin of the system can create new category as producer company, add new car to rent and finally can activate users. By activating them users will authorized to rent a car between two dates.



### Add a category



### Add a car



### Activation of users



## Renting a car

If users activated by the admin then they can visit a form in “rent a car” page under a category which able them to rent their favorable car between those specific dates. In next step they forwarded to a page which shows the rent status of them. If the car had rented between those two days, then a message aware them, and the rent process not executed else if the car was free, the rent process will executed and car will rented for that user



### Successful rent



### Unsuccessful rent



## Logout

The logout button will show if a user logged in. by clicking on it users all alive sessions will destroyed and redirect the user to login page

# Technical info of system

The system designed and implemented based on n-tired layer architecture. There are various layers which is considering as mentioned below.

## Data layer

In this layer database of the system placed. MYSQL manage this layer. The database contains many tables as shown below.

1. User:

This table contains all necessitate information about users

* 1. Varchar name
  2. Varchar familyName
  3. Text address
  4. Varchar email
  5. Varchar password
  6. Varchar phone
  7. Varchar username
  8. Binary confirmed

1. Product:

This table contains all information about cars as our system product

* 1. Integer id
  2. Integer name
  3. Integer categoryId
  4. Integer price

1. renting:

This table contains all information about rented car between specific dates

* 1. Varchar userName
  2. Integer productId
  3. Date fromDate
  4. Date toDate
  5. Integer rentId

1. category:

This table contains all information about categories, in our system they are car producer companies.

* 1. Varchar name
  2. Integer categoryId

## Business layer

This layer of the system bounded to connect the other layers with the data layer. Generally, programming in this layer defines in four steps. At first define a driver to connect to mysql, at next step we connect thorough this driver. In the third step a statement will defines which has the duty of select, update, delete and retrieve processes and finally we execute it. All connections and statements at the end of the methods should be closed.

This layer separate jobs by methods, it means that every task of the system allocated to a method for instance the register method, get a User entity and prepare all steps to save the user in database.

This table shows all available methods, their output and input in the system:

|  |  |  |  |
| --- | --- | --- | --- |
| Name of method | input | output | description |
| getCategory | Integer categoryId | CategoryENT | Get the integer of a categoryId and returns that category |
| getCategoryList |  | ArrayList<CategoryENT> | Returns a list of all categories |
| getCar | Integer carid | CarENT | Get the integer of a carId and returns that car |
| login | String username, String password | UserENT | Check the compatibility of username and password with each other, and then return all attributes of user. Return null if username and password wasn’t match each other. |
| rentCar | RentENT | boolean | Based on the status of a rent, it means that is car available between start and end dates, it returns true or false. True for the time that car is available and rent information saved and false for vice versa situation. |
| insertCar | CarENT | Void | Save the car entity |
| insertCategory | String categName | void | Save the category name |
| register | boolean | UserENT | If registration succeed, returns true else returns false |
| searchACar | CarENT | ArrayList<CarENT> | Get the information of a car and return a list of car which is similar to the input |
| searchACategory | String search | ArrayList<CategoryENT> | Get the search keyword and returns a list of similar categories with the word |
| activation | Boolean x, String uname | void | Change the uname status to x |

In the below codes of my car rental system has shown. It should be mentioned that all general four steps commented in codes.

|  |
| --- |
| /\*  \* To change this template, choose Tools | Templates  \* and open the template in the editor.  \*/  package Business;  import Common.\*;  import java.sql.\*;  import java.util.ArrayList;  /\*\*  \*  \* @author Talieh Dastmalchi  \*/  public class bsManager {  public CategoryENT getCategory(int categoryid){  CategoryENT ent=null;  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection cn=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  Statement stmt = cn.createStatement();  //4 get result  ResultSet rs = stmt.executeQuery("select \* from category where categoryId="+categoryid);  if(rs.next()){  ent = new CategoryENT();  ent.setCategoryid(rs.getInt("categoryId"));  ent.setName(rs.getString("name"));  }  rs.close();  stmt.close();  cn.close();    }catch(Exception ex){  ex.printStackTrace();  }  return ent;  }    public ArrayList<CategoryENT> getCategoryList(){  ArrayList<CategoryENT> ents=null;  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection cn=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  Statement stmt = cn.createStatement();  //4 get result  ResultSet rs = stmt.executeQuery("select \* from category");  if(rs.next()){  CategoryENT ent = new CategoryENT();  ent.setCategoryid(rs.getInt("categoryId"));  ent.setName(rs.getString("name"));  ents.add(ent);  }  rs.close();  stmt.close();  cn.close();    }catch(Exception ex){  ex.printStackTrace();  }  return ents;  }    CarENT getCar(int carid){  CarENT ent=null;  try{    //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection cn = DriverManager.getConnection("jdbc:mysql://localhost:3306/sales","root","");  //3 get statement for execute  Statement stmt = cn.createStatement();  //4 get result  ResultSet rs = stmt.executeQuery("select \* from product where id="+carid);  if(rs.next()){  ent = new CarENT();  ent.setProductid(rs.getInt("id"));  ent.setName(rs.getString("name"));  ent.setPrice(rs.getInt("price"));  ent.setCategoryid(rs.getInt("categoryId"));      CategoryENT category = getCategory(ent.getCategoryid());  ent.setCategory(category);    }  rs.close();  stmt.close();  cn.close();    }catch(Exception ex){  ex.printStackTrace();  }  return ent;  }  public ArrayList<UserENT> usersList() {  ArrayList<UserENT> al = new ArrayList<UserENT>();  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  String sql="Select \* from user";  Statement stmt=con.createStatement();  //4 get result  ResultSet rs = stmt.executeQuery(sql);    while(rs.next())  {  UserENT user = new UserENT();  user.setActive(rs.getBoolean("confirmed"));  user.setUserName(rs.getString("username"));  user.setName(rs.getString("name"));  user.setFName(rs.getString("familyName"));  al.add(user);  }    rs.close();  stmt.close();  con.close();  }  catch (Exception EX){  EX.printStackTrace();  }  return al;  }  public UserENT login(String u, String p) {  UserENT user = new UserENT();  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  String sql="Select \* from user where username = ? AND password = ? ";  PreparedStatement ps=con.prepareStatement(sql);  ps.setString(1,u);  ps.setString(2,p);  //4 get result  ResultSet rs = ps.executeQuery();    if(rs.next()){  user.setActive(rs.getBoolean("confirmed"));  user.setUserName(rs.getString("username"));  user.setName(rs.getString("name"));  user.setFName(rs.getString("familyName"));  }else{  user=null;  }    rs.close();  ps.close();  con.close();  }  catch (Exception EX){  user=null;  EX.printStackTrace();  }    return user;  }  public boolean rentCar(RentENT ent) {  boolean res = false;  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  String sql="Select \* from renting where productId = ? and toDate > ? and fromDate < ? ";  PreparedStatement ps=con.prepareStatement(sql);  ps.setInt(1, ent.getProductId());  ps.setDate(2, ent.getsDate());  ps.setDate(3, ent.geteDate());  //here we check the available cars between the days which user wants to rent  ResultSet rs = ps.executeQuery();  if(rs.next())  {  res = false;  }else{  res = true;  sql="insert into renting (userName, productId, fromDate, toDate) values (?,?,?,?)";  ps=con.prepareStatement(sql);  ps.setString(1, ent.getUserName());  ps.setInt(2, ent.getProductId());  ps.setDate(3, ent.getsDate());  ps.setDate(4, ent.geteDate());  ps.execute();  }  rs.close();  ps.close();  con.close();    }  catch (Exception EX){  EX.printStackTrace();    }  return res;  }  public void insertCar(CarENT ent) {  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  String sql="Insert into product(name, categoryId, price) values (?,?,?)";  PreparedStatement ps = con.prepareStatement(sql);  ps.setString(1,ent.getName());  ps.setInt(2,ent.getCategoryid());  ps.setInt(3,ent.getPrice());  ps.execute();  ps.close();  con.close();    }  catch (Exception EX){  EX.printStackTrace();  }  }  public void insertCategory(String categName) {  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  String sql="Insert into category(name) values (?)";  PreparedStatement ps = con.prepareStatement(sql);  ps.setString(1,categName);  ps.execute();  ps.close();  con.close();    }  catch (Exception EX){  EX.printStackTrace();  }  }  public boolean register(UserENT UserENT) {  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  String sql="Insert into user (name, familyName, address, email, password, phone, username, confirmed) values (?,?,?,?,?,?,?,?)";  PreparedStatement ps = con.prepareStatement(sql);  ps.setString(1,UserENT.getName());  ps.setString(2,UserENT.getFName());  ps.setString(3,UserENT.getAddress());  ps.setString(4,UserENT.getEmail());  ps.setString(5,UserENT.getPassword());  ps.setString(6,UserENT.getPhoneNo());  ps.setString(7,UserENT.getUserName());  ps.setBoolean(8,UserENT.isActive());  ps.execute();  ps.close();  con.close();  return true;  }  catch (Exception EX){  EX.printStackTrace();  return false;  }  }  public ArrayList<CarENT> searchACar(CarENT carENT) {  ArrayList<CarENT> carENTs = new ArrayList<CarENT>();  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  String sql="Select \* from product where name like ? and categoryId= ?";  PreparedStatement ps=con.prepareStatement(sql);  ps.setString(1, "%"+carENT.getName()+"%");  ps.setInt(2, carENT.getCategoryid());  //4 get result  ResultSet rs = ps.executeQuery();  while(rs.next())  {  CarENT ent = new CarENT();  ent.setCategoryid(rs.getInt("categoryId"));  ent.setName(rs.getString("name"));  ent.setPrice(rs.getInt("price"));  ent.setProductid(rs.getInt("id"));  ent.setCategory(getCategory(ent.getCategoryid()));  carENTs.add(ent);  }  rs.close();  ps.close();  }  catch (Exception EX){  EX.printStackTrace();  }  return carENTs;  }    public ArrayList<CategoryENT> searchACategory(String name) {  ArrayList<CategoryENT> categoryENTs = new ArrayList<CategoryENT>();  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  String sql="Select \* from category where name like ?";  PreparedStatement ps=con.prepareStatement(sql);  ps.setString(1, "%"+name+"%");  ResultSet rs = ps.executeQuery();  //4 get result  while(rs.next())  {  CategoryENT ent = new CategoryENT();  ent.setCategoryid(rs.getInt("categoryId"));  ent.setName(rs.getString("name"));  categoryENTs.add(ent);  }  rs.close();  ps.close();  }  catch (Exception EX){  EX.printStackTrace();  }  return categoryENTs;  }  public void activation(boolean x, String uname) {  try{  //1 load driver  Class.forName("com.mysql.jdbc.Driver");  //2 connect  Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/carrental", "root", "root");  //3 get statement for execute  String sql="UPDATE user SET confirmed = ? WHERE username = ? ";  PreparedStatement ps=con.prepareStatement(sql);  ps.setBoolean(1,x);  ps.setString(2,uname);  ps.execute();  ps.close();    con.close();  }  catch (Exception EX){  EX.printStackTrace();  }  }    } |

## Common layer

Based on the principle of object oriented programming, after designing the system and extracting the class diagram from the user needs in the analyze phase of a system development life cycle, we start to develop this layer. All classes and their attributes define here. As we can consider below a class file contains 3 steps. At first the definition of the attribute for instance name of a user, and for two final steps we define a getter and a setter method to get or set specific attribute in the suitable situation.

Car rental system contains four classes which shown below with the embedded code of them

### UserENT

|  |
| --- |
| /\*  \* To change this template, choose Tools | Templates  \* and open the template in the editor.  \*/  package Common;  /\*\*  \*  \* @author Talieh Dastmalchi  \*/  public class UserENT {  private String name;  private String fName;  private String phoneNo;  private String email;  private boolean active=false;  private String address;  private String password;  private String userName;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getFName() {  return fName;  }  public void setFName(String fName) {  this.fName = fName;  }  public String getPhoneNo() {  return phoneNo;  }  public void setPhoneNo(String phoneNo) {  this.phoneNo = phoneNo;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public boolean isActive() {  return active;  }  public void setActive(boolean active) {  this.active = active;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getUserName() {  return userName;  }  public void setUserName(String userName) {  this.userName = userName;  }  } |

### RentENT

|  |
| --- |
| /\*  \* To change this template, choose Tools | Templates  \* and open the template in the editor.  \*/  package Common;  import java.sql.Date;  /\*\*  \*  \* @author Talieh Dastmalchi  \*/  public class RentENT {  private Date sDate;  private Date eDate;  private int productId;  private String userName;  CarENT carENT = new CarENT();  public CarENT getCarENT() {  return carENT;  }  public void setCarENT(CarENT carENT) {  this.carENT = carENT;  }      public Date getsDate() {  return sDate;  }  public void setsDate(Date sDate) {  this.sDate = sDate;  }  public Date geteDate() {  return eDate;  }  public void seteDate(Date eDate) {  this.eDate = eDate;  }  public int getProductId() {  return productId;  }  public void setProductId(int productId) {  this.productId = productId;  }  public String getUserName() {  return userName;  }  public void setUserName(String userName) {  this.userName = userName;  }    } |

### CategoryENT

|  |
| --- |
| package Common;  /\*\*  \*  \* @author Talieh Dastmalchi  \*/  public class CategoryENT {  private int categoryid;  private String name;  public int getCategoryid() {  return categoryid;  }  public String getName() {  return name;  }  public void setCategoryid(int categoryid) {  this.categoryid = categoryid;  }  public void setName(String name) {  this.name = name;  }  } |

### CarENT

|  |
| --- |
| /\*  \* To change this template, choose Tools | Templates  \* and open the template in the editor.  \*/  package Common;  /\*\*  \*  \* @author Talieh Dastmalchi  \*/  public class CarENT {  private int productid;  private String name;  private int price;  private int categoryid;  private CategoryENT category=new CategoryENT();  public int getProductid() {  return productid;  }  public void setProductid(int productid) {  this.productid = productid;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public int getPrice() {  return price;  }  public void setPrice(int price) {  this.price = price;  }  public int getCategoryid() {  return categoryid;  }  public void setCategoryid(int categoryid) {  this.categoryid = categoryid;  }  public CategoryENT getCategory() {  return category;  }  public void setCategory(CategoryENT category) {  this.category = category;  }  } |

## Web layer

This layer contains the servlet files and JSPs. JSPs and servlets transfer information between each other by requests. Servlets are interlayer between JSPs and Business layer actually. Sometimes some information between JSPs and servlets transfer through sessions. Servlets implemented in 4 main phases. At first we get the parameters from request which came from JSPs. In the next phase we call the method and get the result if was necessary. In the third step we prepare parameters which we get from methods in the sessions or in the request. And finally in the last step we redirect a user to a servlet or a JSP file.

### confirmUser

|  |
| --- |
| response.setContentType("text/html;charset=UTF-8");  PrintWriter out = response.getWriter();  try {  //1: setting parameters  boolean p = Boolean.*parseBoolean*(request.getParameter("activate"));  String uname = request.getParameter("activateId");  //2: calling method  bsManager bs = new bsManager();  bs.activation(p,uname);  //3:redirecting user  response.sendRedirect("userList");    } finally {  out.close();  } |

### InsertCar

|  |
| --- |
| try {    //1: setting parameters  CarENT ent = new CarENT();  int categId = Integer.*parseInt*(request.getParameter("categId"));  int price = Integer.*parseInt*(request.getParameter("price"));  String name = request.getParameter("name");  ent.setPrice(price);  ent.setName(name);  ent.setCategoryid(categId);  //2: calling method  bsManager bs = new bsManager();  bs.insertCar(ent);  //3: redirecting to pages  response.sendRedirect("insertCar.jsp?message=Car added");      } finally {  out.close();  } |

### InsertCategory

|  |
| --- |
| try {      //1: setting parameters  String categName = request.getParameter("categoryName");  //2: calling method  bsManager bs = new bsManager();  bs.insertCategory(categName);  //3: redirecting to pages  response.sendRedirect("insertCategory.jsp?message=Category added");  } finally {  out.close();  } |

### Login

|  |
| --- |
| try {  //1: setting parameters  String u = request.getParameter("userN");  String p = request.getParameter("pass");  UserENT ent = new UserENT();  //2: calling method  bsManager bs = new bsManager();  ent=bs.login(u,p);  //3: setting sessions and redirecting to pages  if(ent!=null){  request.getSession().setAttribute("user", ent);  if(ent.getUserName().equalsIgnoreCase("admin")){  response.sendRedirect("admin.jsp");  }else{  response.sendRedirect("index.jsp?message=welcome Mr-MRS " + ent.getName()+" " + ent.getFName());  }  }else{  response.sendRedirect("index.jsp?message=Invalid username or password");  }  } finally {  out.close();  } |

### Logout

|  |
| --- |
| try {  //1: invalidate and close all sessions  request.getSession(true).invalidate();  response.sendRedirect("index.jsp");  } finally {  out.close();  } |

### Register

|  |
| --- |
| response.setContentType("text/html;charset=UTF-8");  PrintWriter out = response.getWriter();  try {  //1: setting parameters  String name = request.getParameter("name");  String fname = request.getParameter("fname");  String address = request.getParameter("address");  String email = request.getParameter("email");  String password = request.getParameter("password");  String phone = request.getParameter("phone");  String usern = request.getParameter("usern");  UserENT ent = new UserENT();  ent.setName(name);  ent.setFName(fname);  ent.setAddress(address);  ent.setEmail(email);  ent.setPassword(password);  ent.setPhoneNo(phone);  ent.setUserName(usern);    //2: calling method  bsManager bs = new bsManager();  boolean registered = bs.register(ent);    //3: setting sessions and redirecting to pages  if(registered){  response.sendRedirect("register.jsp?message=Thanks for your registration");  }else{  response.sendRedirect("register.jsp?message=Registration faild");  }  } finally {  out.close();  } |

### rentACar

|  |
| --- |
| response.setContentType("text/html;charset=UTF-8");  PrintWriter out = response.getWriter();  try {  //1: setting parameters  RentENT rent = new RentENT();  rent.setsDate(Date.*valueOf*(request.getParameter("startDate")));  rent.seteDate(Date.*valueOf*(request.getParameter("endDate")));  rent.setUserName(request.getParameter("userName"));  rent.setProductId(Integer.*parseInt*(request.getParameter("carId")));  //2: calling method  bsManager bs = new bsManager();  boolean res = bs.rentCar(rent);  //3: redirecting to pages  if(res)  response.sendRedirect("rentSucceed.jsp?message=successful renting");  else  response.sendRedirect("rentSucceed.jsp?message=the car rented before");  } finally {  out.close();  } |

### searchForCar

|  |
| --- |
| response.setContentType("text/html;charset=UTF-8");  PrintWriter out = response.getWriter();  try {  //1: setting parameters  CarENT ent = new CarENT();  ArrayList<CarENT> ents = new ArrayList<CarENT>();  CategoryENT categoryENT = new CategoryENT();  int categId = Integer.*parseInt*(request.getParameter("categId"));  String name = request.getParameter("search");  if("null".equalsIgnoreCase(name)||name==null)  name="";  ent.setCategoryid(categId);  ent.setName(name);  //2: calling method  bsManager bs = new bsManager();  ents = bs.searchACar(ent);  categoryENT = bs.getCategory(categId);  //3: setting sessions and redirecting to pages  request.getSession().setAttribute("category", categoryENT);  request.getSession().setAttribute("cars", ents);  response.sendRedirect("carList.jsp");    } finally {  out.close();  } |

### searchForCategory

|  |
| --- |
| response.setContentType("text/html;charset=UTF-8");  PrintWriter out = response.getWriter();  try {  //1: setting parameters  String name = request.getParameter("search");  if("null".equalsIgnoreCase(name)||name==null)  name="";  //2: calling method  bsManager bs = new bsManager();  ArrayList<CategoryENT> ents = new ArrayList<CategoryENT>();  ents = bs.searchACategory(name);  //3: setting sessions and redirecting to pages  request.getSession().setAttribute("categoryList", ents);  response.sendRedirect("categoryList.jsp");  } finally {  out.close();  } |

### userList

|  |
| --- |
| response.setContentType("text/html;charset=UTF-8");  PrintWriter out = response.getWriter();  try {  bsManager bs = new bsManager();  ArrayList<UserENT> al = bs.usersList();  request.getSession().setAttribute("userListSession", al);  response.sendRedirect("userList.jsp");  } finally {  out.close();  } |

### JSPS

Information between servlet and JSPs transfer in two types which is deferent based on the departure and destination. If departure was a JSP they usually send information by forms for instance register form of car rental system is shown below. The form sends all information to the register as action of the form.

|  |
| --- |
| <form action=*"register"* method=*"post"* name=*"form"*>  <table width=*"324"* align=*"center"*>  <tr>  <td colspan=*"2"* align=*"center"*><span >  Register</span></td>  </tr>  <tr>  <td width=*"130"* >Name: </td>  <td width=*"178"*>  <input name=*"name"* type=*"text"*></td>  </tr>  <tr>  <td >Family Name: </td>  <td><input name=*"fname"* type=*"text"*>  </td>  </tr>  <tr>  <td >Telephone:</td>  <td><input name=*"phone"* type=*"text"*></td>  </tr>  <tr>  <td >Email:</td>  <td><input name=*"email"* type=*"text"*></td>  </tr>  <tr>  <td >Password:</td>  <td><input name=*"password"* type=*"password"*></td>  </tr>  <tr>  <td >Username:</td>  <td><input name=*"usern"* type=*"text"*></td>  </tr>  <tr>  <td valign=*"top"*>Address: </td>  <td><textarea name=*"address"*></textarea></td>  </tr>  <tr>  <td colspan=*"2"* align=*"center"*>  <input name=*""* type=*"submit"* value=*"Register"*>  </td>  </tr>  </table>  </form> |

Or maybe they call pages through servlets for instance in the example below, we generate pages through categoryId. It means when a user click on one of the anchors in the list they call a servlet by sending categoryId to it, then servlet will show the specified page of that categoryId

|  |
| --- |
| <td class=*"listItem"*>  <a href=*"searchForCar?categId=*<%=ents.get(i).getCategoryid() %>*"*> <%= ents.get(i).getName() %></a>  </td> |

Now, if a servlet was departure and JSP was destination. In this case like clicking on “rent a car” button in the menu we call searchForCategory servlet.

|  |
| --- |
| <a href=*"searchForCategory"*>rent a car</a> |

The servlet create a session which contains a list of all categories and redirect the user to “categoryList.jsp”. In this page we should extract information from session and show them as an HTML format. The “for” loop in the session and lists all of them.

|  |
| --- |
| <% ArrayList<CategoryENT> ents = (ArrayList<CategoryENT>)session.getAttribute("categoryList");  //create a list of categories from the categoryList session  for(int i = 0 ; i < ents.size() ; i ++ ){  //list all the categories attributes in the table  %>  <tr>  <td class=*"listItem"*>  <%= i+1 %>  </td>  <td class=*"listItem"*>  <a href=*"searchForCar?categId=* <%=ents.get(i).getCategoryid() %>*"*>  <%= ents.get(i).getName() %></a>  </td>  </tr>  <%}%> |
|  |

The other pages such as carList, UserList, combo box of the NewCar form and many other pages follow this structure.

## RMI layer

In distributed systems which use similar data layer architecture, we can use our system methods in business layer remotely. This layer contains three parts. At the first there exist a remote server; it contains the methods definition and what they should to do as similar as we done in the business layer. In the second step there is an interface which included all exist methods in the server side RMI, the methods should throws a Remote Exception. And finally in the client side when a method wants to call the client should call them through the interface. The codes of RMI layer of the car rental system shown below.

## SaleRMIServer

|  |
| --- |
| /\*  \* To change this template, choose Tools | Templates  \* and open the template in the editor.  \*/  package RMI;  import java.net.MalformedURLException;  import java.rmi.Naming;  import java.rmi.RemoteException;  import java.rmi.server.UnicastRemoteObject;  import java.util.ArrayList;  import Business.bsManager;  import Common.CarENT;  import Common.CategoryENT;  import Common.RentENT;  import Common.UserENT;  /\*\*  \*  \* @author Talieh Dastmalchi  \*/  public class SaleRMIServer extends UnicastRemoteObject implements  SaleRMIInterface {  public static void main(String[] args) {  try {  SaleRMIServer obj = new SaleRMIServer();  } catch (RemoteException ex) {  ex.printStackTrace();  }  }  public SaleRMIServer() throws RemoteException {  try {  Naming.rebind("SaleRMIServer", this);  } catch (MalformedURLException ex) {  // Logger.getLogger(SaleRMIServer.class.getName()).log(Level.SEVERE,  // null, ex);  // server - mine  ex.printStackTrace();  // client -  throw new RemoteException(ex.getMessage());  }  }  public String helloRMI(String name) throws RemoteException {  // throw new UnsupportedOperationException("Not support yet.");  return name + " Welcome to RMI";  }  public void activation(boolean p, String uname) throws RemoteException {  bsManager bsManager = new bsManager();  bsManager.activation(p, uname);  }  public UserENT login(String u, String p) throws RemoteException {  bsManager bsManager = new bsManager();  return bsManager.login(u, p);  }  public boolean register(UserENT UserENT) throws RemoteException {  bsManager bsManager = new bsManager();  return bsManager.register(UserENT);  }  public ArrayList<UserENT> usersList() throws RemoteException {  bsManager bsManager = new bsManager();  return bsManager.usersList();  }  public CategoryENT getCategory(int categoryid) throws RemoteException {  bsManager bsManager = new bsManager();  return bsManager.getCategory(categoryid);  }  public ArrayList<CategoryENT> getCategoryList() throws RemoteException {  bsManager bsManager = new bsManager();  return bsManager.getCategoryList();  }  public void insertCar(CarENT ent) throws RemoteException {  bsManager bsManager = new bsManager();  bsManager.insertCar(ent);  }  public void insertCategory(String categName) throws RemoteException {  bsManager bsManager = new bsManager();  bsManager.insertCategory(categName);  }  public boolean rentCar(RentENT ent) throws RemoteException {  bsManager bsManager = new bsManager();  return bsManager.rentCar(ent);  }  public ArrayList<CarENT> searchACar(CarENT carENT) throws RemoteException {  bsManager bsManager = new bsManager();  return bsManager.searchACar(carENT);  }  public ArrayList<CategoryENT> searchACategory(String name)  throws RemoteException {  bsManager bsManager = new bsManager();  return bsManager.searchACategory(name);  }  } |

## SaleRMIInterface

|  |
| --- |
| /\*  \* To change this template, choose Tools | Templates  \* and open the template in the editor.  \*/  package RMI;  import java.rmi.Remote;  import java.rmi.RemoteException;  import java.util.ArrayList;  import Common.CarENT;  import Common.CategoryENT;  import Common.RentENT;  import Common.UserENT;  /\*\*  \*  \* @author Talieh Dastmalchi  \*/  public interface SaleRMIInterface extends Remote {  public String helloRMI (String name) throws RemoteException;  public ArrayList<UserENT> usersList() throws RemoteException;  public UserENT login(String u, String p) throws RemoteException;  public boolean register(UserENT UserENT) throws RemoteException;  public void activation(boolean p, String uname) throws RemoteException;  public ArrayList<CategoryENT> searchACategory(String name) throws RemoteException;  public ArrayList<CarENT> searchACar(CarENT carENT) throws RemoteException;  public void insertCategory(String categName) throws RemoteException;  public void insertCar(CarENT ent) throws RemoteException;  public boolean rentCar(RentENT ent) throws RemoteException;  public ArrayList<CategoryENT> getCategoryList() throws RemoteException;  public CategoryENT getCategory(int categoryid) throws RemoteException;  } |

## SaleRMIClinet

A main method created to test the system RMI client side.

|  |
| --- |
| public static void main (String[] args){  try{  System.*setSecurityManager*( new RMISecurityManager());  SaleRMIInterface bs = (SaleRMIInterface) Naming.*lookup*("rmi://localhost:1000/SaleRMIServer");  String st = bs.helloRMI("ali");  System.*out*.println(st);  }catch (Exception ex) {  ex.printStackTrace();    }    } |

# NET Layer

Sometimes, some people want to access some ability of systems through a command mode instead of a web browser. They can call domain name of the system in that mode, then they see a welcome screen and a list of commands which some of them such as getting category list get a search parameter for instance and some others doesn’t.

Codes embedded in the table below. As an example of use system through this layer, after viewing the commands line user enter the “CATEGORYLIST b” then the system will print a list of all categories which name of them contains ‘b’ for example **B**MW, Mercedes **B**enz and etc.

### Net Server

|  |
| --- |
| /\*  \* To change this template, choose Tools | Templates  \* and open the template in the editor.  \*/  package NET;  import java.io.BufferedReader;  import java.io.InputStreamReader;  import java.io.PrintWriter;  import java.net.ServerSocket;  import java.net.Socket;  import java.util.ArrayList;  import Business.bsManager;  import Common.CarENT;  import Common.CategoryENT;  import Common.UserENT;  /\*\*  \*  \* @author Talieh Dastmalchi  \*/  public class SaleNetServer {  public void runserver1(){    try{  //1  ServerSocket server = new ServerSocket(1000);  //2  Socket s = server.accept();  //3  s.getOutputStream().write("Welcome to sale".getBytes());  s.getOutputStream().flush();  //4-1  PrintWriter out = new PrintWriter(s.getOutputStream());    //4-2  InputStreamReader isr = new InputStreamReader(s.getInputStream());  BufferedReader in = new BufferedReader(isr);  //7  while(true){    out.println("\r\nenter command:\r\nUSERLIST \r\nLOGIN \r\nCATEGORYLIST \r\nPRODUCTLIST \r\n...");  out.flush();  String command = in.readLine();  //5  if("USERLIST".equals(command.toUpperCase())){  bsManager bs = new bsManager();  ArrayList<UserENT> ar = new ArrayList<UserENT>();// bs.productlist(); yani az badaz ar= be jaye new... bayad bs.pr... ra bayad benevisim vali ye joor dige neveshtimo commentash kardim chon database saleBS ra nasakhtim ghablan out.print("\r\n productname \t price \t category");  ar=bs.usersList();  out.print("name\t family name \t username \t");  for (int i=0;i<ar.size();i++){  out.print("\r\n");  out.print(ar.get(i).getName());  out.print("\t");  out.print(ar.get(i).getFName());  out.print("\t");  out.print(ar.get(i).getUserName());  out.print("\t");  }  out.flush();  //6  }else if("LOGIN".equals(command.toUpperCase())){  bsManager bs = new bsManager();  String[] st = command.split(" ");  UserENT ent = bs.login(st[1], st[2]);  if(ent==null)  out.print("invalid username or password");  else  out.print("welcome Mr/Mrs "+ent.getName()+ ent.getFName());  }else if("CATEGORYLIST".equals(command.toUpperCase())){  bsManager bs = new bsManager();  ArrayList<CategoryENT> ents = bs.getCategoryList();  out.print("category name");  for (int i = 0 ; i < ents.size() ; i ++){  out.print("\r\n");  out.print(ents.get(i).getName());  }  }else if("EXIT".equals(command.toUpperCase())){  out.print("\r\n Good Bye .");  break;  }    out.println("\r\nyou entered :"+command);  out.flush();    }//While inja baste shod    s.close();  server.close();  }catch(Exception ex){  ex.printStackTrace();      }  }  public void runserver(){  try{  //1  ServerSocket server = new ServerSocket(1000);  //8  boolean b = true;  while(b){  //2  Socket s = server.accept();  //8  SaleNetThread t = new SaleNetThread();  t.setCustomer(s);  t.start();    }  server.close();  }catch(Exception ex){  ex.printStackTrace();  }  }  } |

### Net thread server

In the NET layer we define a Net thread server. The duty of threads is to handle requests which came to server in the same time. For instance suppose that at just one moment 10000 people send the request of getting USERLIST from the server, then the threads starts to parallel processing in CPU and responding to requests at that time.

|  |
| --- |
| /\*  \* To change thicustomer template, choocustomere Tools | Templates  \* and open the template in the editor.  \*/  package NET;  import java.io.BufferedReader;  import java.io.InputStreamReader;  import java.io.PrintWriter;  import java.net.Socket;  import java.util.ArrayList;  import java.util.Date;  import java.util.Hashtable;  import javax.jms.Session;  import Business.bsManager;  import Common.CategoryENT;  import Common.UserENT;  /\*\*  \*  \* @author Talieh Dastmalchi  \*/  public class SaleNetThread extends Thread{  private Socket customer = null;  Hashtable session=new Hashtable();    public Socket getCustomer() {  return customer;  }  public void setCustomer(Socket customer) {  this.customer = customer;  }  public void run(){  try{  //3  customer.getOutputStream().write("Welcome to sale".getBytes());  customer.getOutputStream().flush();  //4-1  PrintWriter out = new PrintWriter(customer.getOutputStream());    //4-2  InputStreamReader isr = new InputStreamReader(customer.getInputStream());  BufferedReader in = new BufferedReader(isr);  //7  while(true){    out.println("\r\nenter command:\r\nSEARCH \r\nPRODUCTLIST \r\nLOGIN \r\n...");  out.flush();  String command = in.readLine();  //5  if("USERLIST".equals(command.toUpperCase())){  bsManager bs = new bsManager();  ArrayList<UserENT> ar = new ArrayList<UserENT>();// bs.productlist(); yani az badaz ar= be jaye new... bayad bs.pr... ra bayad benevisim vali ye joor dige neveshtimo commentash kardim chon database saleBS ra nasakhtim ghablan out.print("\r\n productname \t price \t category");  ar=bs.usersList();  for (int i=0;i<ar.size();i++){  out.print("\r\n");  out.print(ar.get(i).getName());  out.print("\t");  out.print(ar.get(i).getFName());  out.print("\t");  out.print(ar.get(i).getUserName());  out.print("\t");  }  out.flush();  //6  }else if("LOGIN".equals(command.toUpperCase())){  bsManager bs = new bsManager();  String[] st = command.split(" ");  UserENT ent = bs.login(st[1], st[2]);  if(ent==null){  out.print("invalid username or password");  session.put("user",ent);  }else  out.print("welcome Mr/Mrs "+ent.getName()+ ent.getFName());  }else if("CATEGORYLIST".equals(command.toUpperCase())){  bsManager bs = new bsManager();  ArrayList<CategoryENT> ents = bs.getCategoryList();  out.print("category name");  for (int i = 0 ; i < ents.size() ; i ++){  out.print("\r\n");  out.print(ents.get(i).getName());  }  }  else if("EXIT".equals(command.toUpperCase())){  out.print("\r\n Good Bye .");  break;  }          out.flush();    }//While inja baste shod  customer.close();  }catch(Exception ex){  ex.printStackTrace();  }  }//run  }//class |

# Data Layer embedded codes

|  |
| --- |
| /\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;  /\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;  /\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;  /\*!40101 SET NAMES latin1 \*/;  SET FOREIGN\_KEY\_CHECKS=0;  CREATE DATABASE `carrental`  CHARACTER SET 'latin1'  COLLATE 'latin1\_swedish\_ci';  USE `carrental`;  CREATE TABLE `category` (  `categoryId` int(11) NOT NULL AUTO\_INCREMENT,  `name` varchar(20) DEFAULT NULL,  PRIMARY KEY (`categoryId`)  ) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=latin1;  CREATE TABLE `product` (  `id` int(11) NOT NULL AUTO\_INCREMENT,  `name` varchar(20) COLLATE latin1\_general\_ci NOT NULL DEFAULT '0',  `categoryId` int(11) NOT NULL DEFAULT '0',  `price` int(11) NOT NULL DEFAULT '0',  PRIMARY KEY (`id`)  ) ENGINE=MyISAM AUTO\_INCREMENT=9 DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;  CREATE TABLE `renting` (  `userName` varchar(20) COLLATE latin1\_general\_ci NOT NULL DEFAULT '',  `productId` int(11) NOT NULL DEFAULT '0',  `fromDate` date NOT NULL DEFAULT '0000-00-00',  `toDate` date NOT NULL DEFAULT '0000-00-00',  `rentId` int(11) NOT NULL AUTO\_INCREMENT,  PRIMARY KEY (`rentId`),  UNIQUE KEY `rentId` (`rentId`)  ) ENGINE=MyISAM AUTO\_INCREMENT=3 DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;  CREATE TABLE `user` (  `name` varchar(20) COLLATE latin1\_general\_ci NOT NULL DEFAULT '',  `familyName` varchar(20) COLLATE latin1\_general\_ci NOT NULL DEFAULT '',  `address` text COLLATE latin1\_general\_ci NOT NULL,  `email` varchar(50) COLLATE latin1\_general\_ci NOT NULL DEFAULT '',  `password` varchar(20) COLLATE latin1\_general\_ci NOT NULL DEFAULT '',  `phone` varchar(20) COLLATE latin1\_general\_ci NOT NULL DEFAULT '',  `username` varchar(20) COLLATE latin1\_general\_ci NOT NULL DEFAULT '',  `confirmed` binary(1) NOT NULL DEFAULT '\0',  PRIMARY KEY (`username`)  ) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1\_general\_ci;  /\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;  /\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;  /\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/; |