Telephone Number System and Ring Back Tone

University of Stavanger

Name: Yizu Li Student NO.: 208016

2011-6-15

ABSTRACT

The development of Computer Network Technology has brought great changes to the modern society, especially e-commerce relying on network has leap-type development. So, how to use the network to improve users' experience, enabling users' operation to become faster, more convenient and intuitive, is now the focus of many enterprises. So that customers can stay at home for business, not only convenient for customers, but also saves the company's human resources.

I did this project with Altibox company, trying to establish a telephone number system, making individuals and business users can complete the business on the internet, and also implements a simple ring back tones system, the program is built with struts2, hibernate, oracle-database , tomcat, using HTML, JSP and struts2 tag for page display.

1. INTRODUCTION

Altibox has since 2002 provided triple play services (Internet Access, VoIP, IPTV) to residential customers in Norway and Denmark. Services from Altibox are delivered using Fiber to the Home. Altibox do not sell services directly to customers; a number of franchise partners sell the services and manage the customer relationship. In recent years additional services such as IP based residential alarms and mobile services have been added to the Altibox service portfolio.

The object model of the unified telephone number inventory system is quite simple. Number Series, consisting of a certain amount of telephone numbers, are allocated by the National Post and Telecommunications Authority (in Norway this is www.npt.no) to service providers who operate fixed-line or mobile services.

The unified telephone number inventory system shall act as a master resource repository for fixed-line and mobile telephone numbers, while a common inventory at Altibox will maintain the relationship between subscriber and number.

We are now working on a new telephone number system which can offer web service to the customers so that they can manage their telephone numbers on the internet. In additional, I am trying to build a simple RBT (Ring Back Tone) system, which allows users to choose ring back tone online, replacing the old boring original tone.

2. Background

Currently, if a user wants to apply for a telephone number, he has to go to a relevant office (such as stores, supermarket etc.), and the telephone number he obtained is randomly selected. If we can allow customers to choose a specific number that they want? Customers want to know that if their favorite number is already occupied, and if they can apply to get that number. The operators also hope to manage and classify telephone numbers, such as different phone numbers should belong to different areas, some contain special digit number should be distinguished from other ordinary numbers. For business users, they want to apply to a group of consecutive numbers. Therefore, now we need a new telephone number system to implement these new requirements.

2.1 System scope and context

r.001

The system shall keep track of Telephone Numbers. Any telephone number is part of a Telephone Number Series. A typical telephone number series in Norway contains between 1000 and 100.000 telephone numbers.

r.002

Telephone Number Series are allocated to one or more services. Typical services are Voice, Data, Fax, M2M.

r.003

Number series are categorized as either geographical or mobile. (although, the government plans to allow for use of numbers across these two categories)

r.004

Geographical number series are only allowed to operate in a certain geographic region. In Norway a geographic region is typically a County (such as Rogaland or Vest Agder).

r.005

Altibox operates in multiple countries, currently Norway and Denmark. The unified telephone number inventory system should be able to support operation in any country, at minimum the two mentioned countries

r.006

The system must support adding attributes to numbers, to support such features as “special number” (gold, silver, bronze)

r.007

Numbers must have a status. Possible statuses are “Available, Not Available, Reserved, Ported Out, Ported In, In Use”. The system must support additional statuses if needed.

r.008

Numbers assigned by regulatory authorities are imported into the system in number series. However, numbers may also be ported in by customers. This means that the number series which the specific number originates from is owned by another

operator. The same can happen to Altibox numbers, customers can choose to port out their telephone number to another operator.

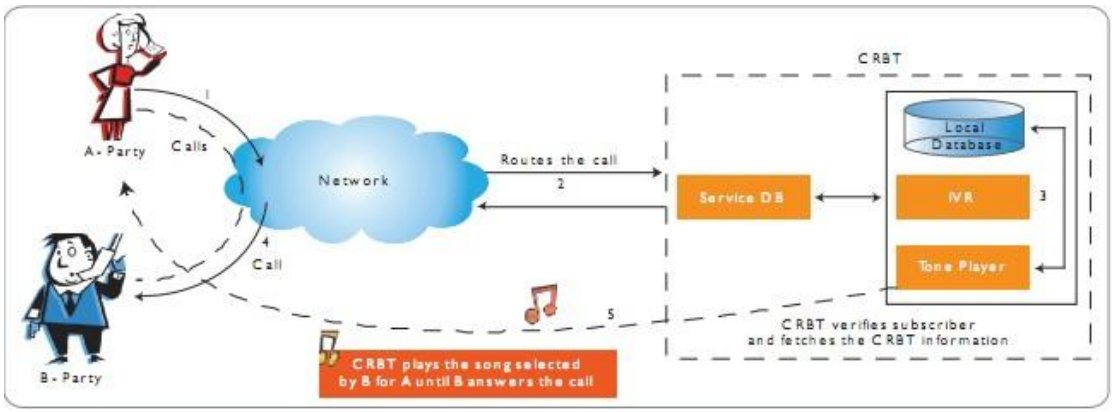
2.2 Ring Back Tone System

2.2.1 What is ring back tone?

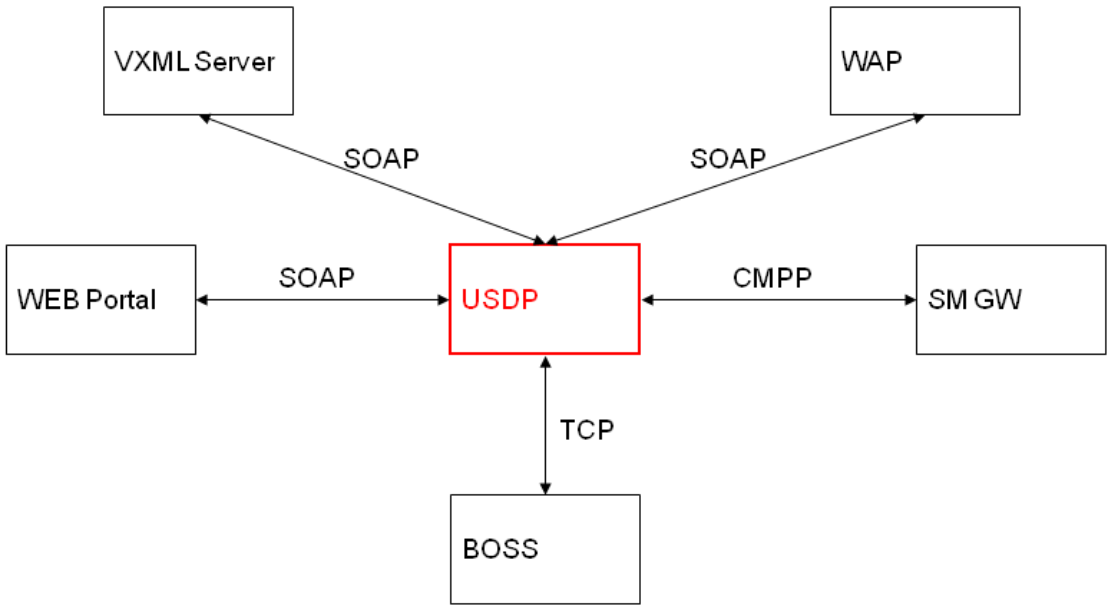
CRBT (Coloring Ring Back Tone) is a personalized ring back tone business. It is a business that the called user can customize personalized ring back tone to the calling user [1]. The original idea came from South Korean SK Corporation. Now it is very popular in East Asia (China, Japan, Korea etc.).

Traditional: The traditional voice communication services, when user A is calling user B, before user B answer the phone, user A will hear the monotonous “beep ~ beep ~” ring back tone.

CRBT: If user B has enabled the CRBT service and customized his ring back tone, then user A will hear the music which user B customized instead of the original ring back tone.



2.2.2 CRBT software side network diagram



Features of each module

1) WEB Portal: RBT Portal is the gateway to an external RBT service, the management system is divided into parts of the system administrator, SP part, some business users, customer service part, for system administrators, SP administrators, business users, provides access to customer service Interface. System administrator, SP administrator, RBT business users and customer service staff can be HTTP (Hyper Text Transport Protocol) access to Portal, through a simple intuitive customization for business, management and other functions.

2) VXML Server: Voice Extensible Markup Language VXML (Voice eXtensible Markup Language) workflow script, according to Interactive Voice Response IVR (Interactive Voice Response)'s request for the corresponding VXML process scripts.

3) USDP: General business development platform USDP (Universal Service Development Platform) is the core of a unified platform for value-added voice. Will be part of the interface with the core functional module separated by SOAP protocol to achieve the core database, core logic and business separation. USDP provide a unified external interface, and the message center interface directly with the intelligent network or the BOSS interface directly. Portal and VXML only simple logic.。

4) CTI / UI: call connection control and tones play channel control

5) DB: a database (including user data and all system-related data)

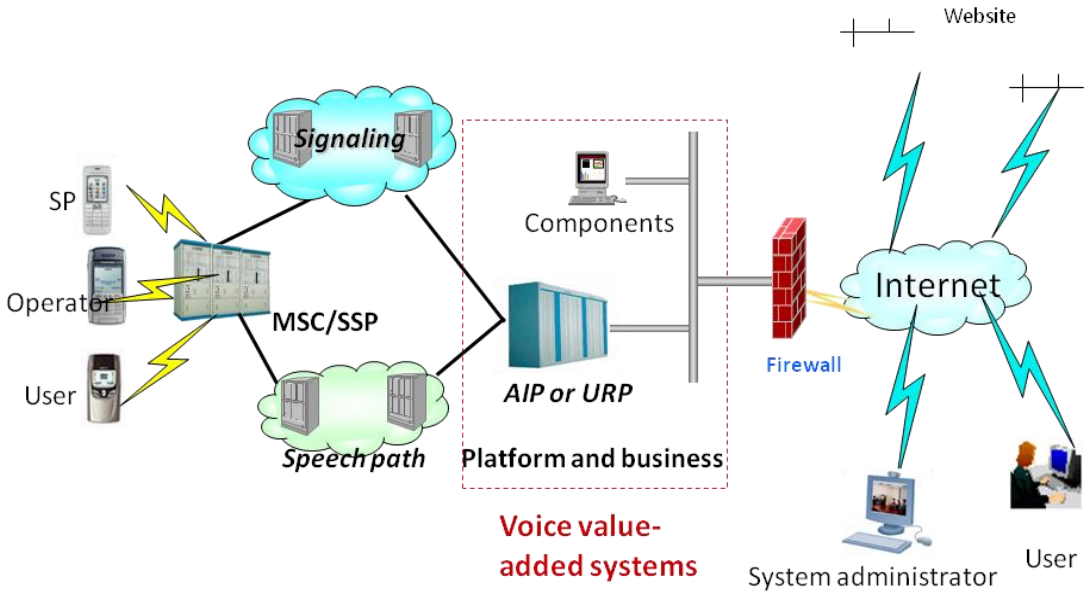
6) FS: ringtone file server (all the ring tones stored information file)

7) GW: CRBT access gateway (Provide SP, the central music platform and other services
and Ring management interface)

8) F5: Ethernet load balancers (Complete virtual network address mapping and traffic load balancing function)

9) AIP + VRS / URP: Call Admission and play ring tones

2.2.3 CRBT system external network



Due to Altibox company currently doesn't have their own switch machine, therefore, my project focus on the website part, to build a website that users can play and customize a ring back tone for their telephone number.

3. Software

The telephone number system is built with Struts2, Hibernate, Oracle-database, Java 6, Tomcat 5, etc. using HTML, JSP and struts2 tags for page display.

Struts2:

Apache Struts 2 is an elegant, extensible framework for creating enterprise-ready Java web applications [2]. The framework is designed to streamline the full development cycle, from building, to deploying, to maintaining applications over time.

Apache Struts 2 was originally known as WebWork 2. After working independently for several years, the WebWork and Struts communities joined forces to create Struts2. This new version of Struts is simpler to use and closer to how Struts was always meant to be. Apache Struts is an open-source web application framework for developing Java EE web applications [3]. It uses and extends the Java Servlet API to encourage developers to adopt a model-view-controller (MVC) architecture.

In this project, I use Struts2 to process the interaction of client web page and business logic, data validation and conversion.

Hibernate:

Hibernate is a high-performance Object/Relational persistence and query service [4]. The most flexible and powerful Object/Relational solution on the market, Hibernate takes care of the mapping from Java classes to database tables and from Java data types to SQL data types. It provides data query and retrieval facilities that significantly reduce development time. Hibernate's design goal is to relieve the developer from 95% of common data persistence-related programming tasks by eliminating the need for manual, hand-crafted data processing using SQL and JDBC. However, unlike many other persistence solutions, Hibernate does not hide the power of SQL from you and guarantees that your investment in relational technology and knowledge is as valid as always.

Hibernate responsible for data persistence and data communication with database in my project.

Oracle-database:

Oracle is the world's most complete, open, and integrated business software and hardware systems company. The Oracle Database (commonly referred to as Oracle

RDBMS or simply as Oracle) is an object-relational database management system produced and marketed by Oracle Corporation.

1. Globalization, cross-platform database.
2. Oracle to comply with data access language, operating system, user interface and industry standard network communication protocols.
3. Supports multiple users, high-performance transaction processing.
4. Strong security controls and integrity controls.
5. Support distributed databases and distributed processing.

In my project, I use the Oracle Database to store information and data of telephone numbers, users, ring back tones and transaction records.

Java 6:

Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible [5]. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.

I use Java application to manage and access the Oracle Database, achieving some function and the communication between client and server.

Tomcat 5:

Apache Tomcat (or Jakarta Tomcat or simply Tomcat) is an open source servlet container developed by the Apache Software Foundation (ASF) [6]. Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run.

4. Design

4.1 Overall Design:

In this section, the system architecture design will be explained. This section includes all design ideas was intended to be implemented, however, due to limited time, some of them was dropped, those not implemented elements will be pointed out in description.

4.2 Overall Architecture:

Multi-layered architecture

"A multilayered software architecture is using different layers for allocating the responsibilities of an application". A multi-layered architecture is easier to understand and maintain. Each layer has specific function or domain that obviously differ the others'. The most popular multilayer architecture is the "3-tier architecture",

Presentation ←→ Business Logic ←→ Data.

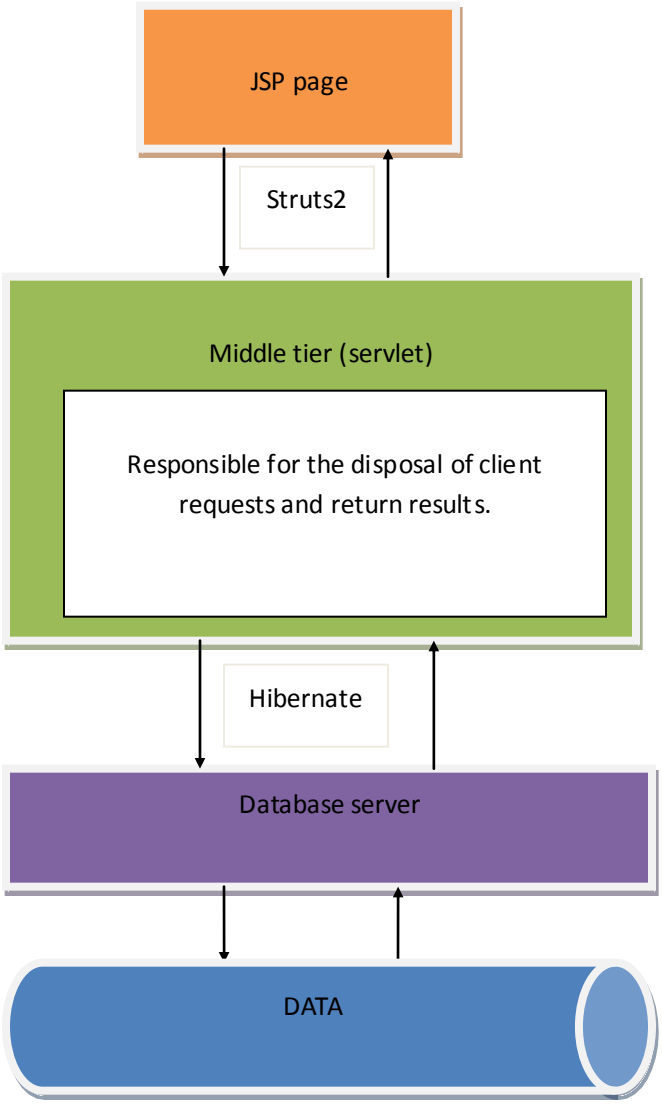
Each layer has its own responsibility in the system. Inside each layer, some sub-layers may exist. For example in the presentation layer involve Module-View-Controller pattern, then Module, View and Controller can be seen as the sub-layers.

I have used the model view controller design pattern in designing the system. In both client and server, I have used this design pattern to separate the business logic from the front end and database layer. The server side is handled by the controller which binds the methods of the server as a service and it contains implementation that separates the database implementation of the server.

There are three layer in my program:

1) Web client
2) Middle tier
3) Database

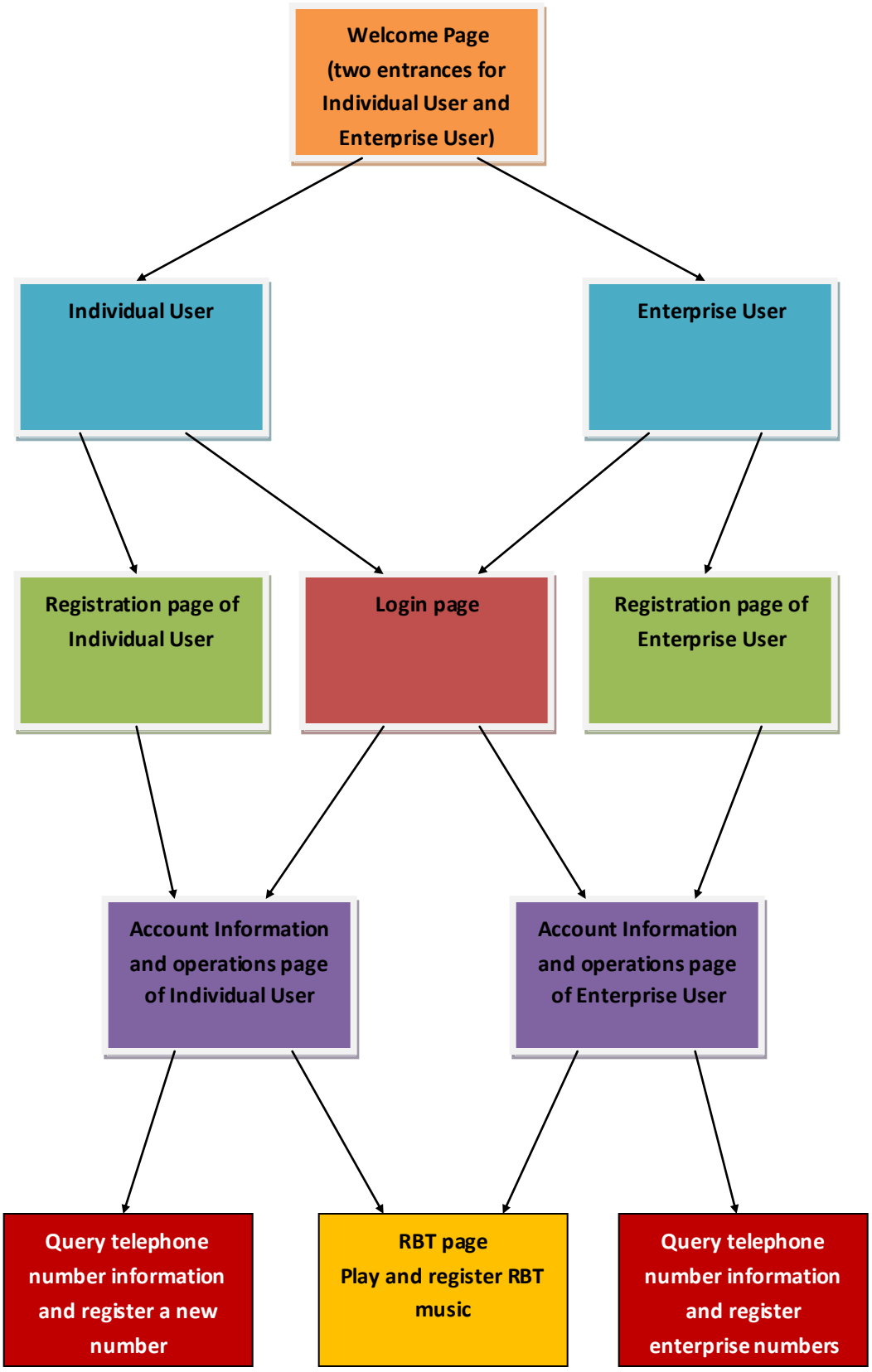
The following figure is the architecture:



4.3 Client Web Page Design

On the client web page side, I am supposed to provide the functions: login, which allows the client (individual user and enterprise user) to log into the server; new user registration, which allows the client to get a user account from server; telephone number information display, which allows the client to check his/her telephone number account, and manage his/her telephone numbers; new telephone number registration, to search and register a new available telephone number; Ring Back Tone registration, to play and bind a ring back tone to client's telephone numbers.

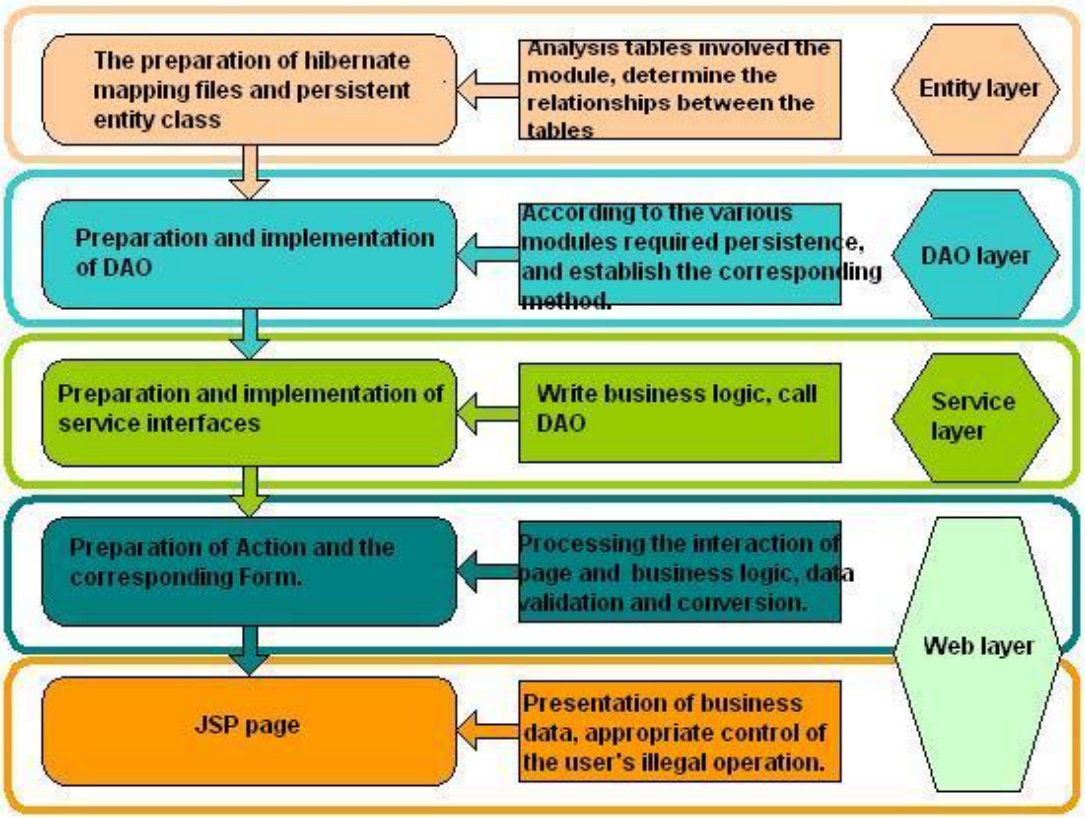
Figure shows the main web page design for client.


Welcome Page
(two entrances for Individual User and Enterprise User)
Individual User
Enterprise User
Registration page of Individual User
Login page
Registration page of Enterprise User
Account Information and operations page of Individual User
Account Information and operations page of Enterprise User
Query telephone number information and register a new number
RBT page
Play and register RBT music
Query telephone number information and register enterprise numbers

4.4 Middle Tier Design:

Basically, the middle tier is consist of Struts2 and Hibernate, Struts2 responsible for processing and validating the data, and Hibernate responsible for data persistence and data communication with database.

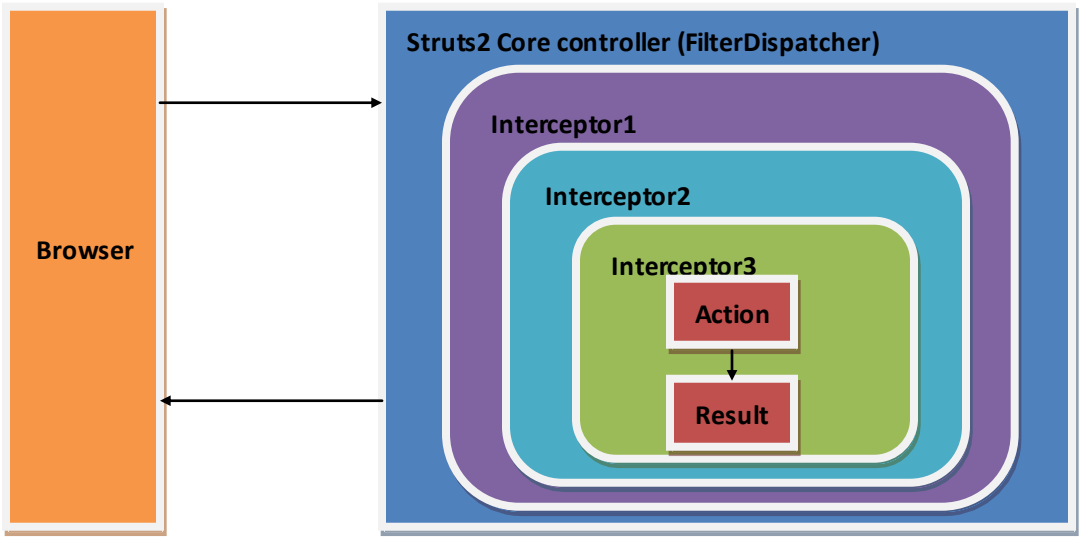
Following picture is the structure of middle tier.



4.4.1 Web layer:

In this layer, I used Struts2 to process the interaction of client web page and business logic, data validation and conversion.

Structure of Struts2:



Workflow:

1. Load the class (FilterDispatcher)
2. Read the configuration (Actions in Struts configuration files)
3. The distribution of the request (client sent a request)
4. Call Action (FilterDispatcher read the corresponding Action from Struts configuration file)
5. Enable the interceptor (WebWork interceptor chain automatically use common functions to the request, such as validation)
6. Operations processing (callback Action's execute() method)
7. Returns a response (using the execute() method returns information to the FilterDispatcher)
8. Search the response (FilterDispatcher resolve the information by searching the configuration, such as: SUCCESS, ERROER, and jump to the corresponding JSP page)
9. Response to user (JSP → client browser)

According to the operations in the web page, I defined several Actions corresponding with the forms in the web page by using Struts2-XML:

```
<struts>
    <package name="front" namespace="/" extends="struts-default" >
     <!-- <default-action-ref name="index"/>  --> <!-- bug!!! -->
```

```
    <action name="add" class="com.headoor.PND.action.UserAction"
method="addUser">
        <result>userLogin.jsp</result>
        </action>

        <action name="login" class="com.headoor.PND.action.UserAction"
method="loginUser">
        <result name="input">userLogin.jsp</result>
        <result>userLogin.jsp</result>
        </action>

        <action name="logout" class="com.headoor.PND.action.UserAction"
method="logoutUser">
        <result>userLogin.jsp</result>
        </action>

        <action name="load" class="com.headoor.PND.action.UserAction"
method="loadUser">
        <result>/success.jsp</result>
        </action>

        <action name="search"
class="com.headoor.PND.action.NumberAction" method="corpList">
        <result name="input">/corporateRegister.jsp</result>
        <result>/corporateRegister.jsp</result>
        </action>

        <action name="corpRegister"
class="com.headoor.PND.action.NumberAction" method="corpRegister">
        <result>/userLogin.jsp</result>
        <result name="input">/corporateRegister.jsp</result>
        </action>

        <action name="register"
class="com.headoor.PND.action.NumberAction" method="register">
        <result>/userLogin.jsp</result>
        <result name="input">/numberRegister.jsp</result>
        </action>

        <action name="check" class="com.headoor.PND.action.UserAction"
method="check">
        <result>/userInf.jsp</result>
        </action>
```

```
        <action name="customize"
class="com.headoor.PND.action.NumberAction" method="customize">
        <result name="input">/numberRegister.jsp</result>
        <result>/numberRegister.jsp</result>
        </action>

        <action name="RBT" class="com.headoor.PND.action.RBTAction"
method="listRBT">
        <result>/rbt.jsp</result>
        </action>

        <action name="registerRBT"
class="com.headoor.PND.action.RBTAction" method="registerRBT">
        <result type="chain">check</result>
        </action>

        <action name="goRegisterRBT"
class="com.headoor.PND.action.RBTAction" method="goRegisterRBT">
        <result type="chain">RBT</result>
        </action>

        <action name="enableNumber"
class="com.headoor.PND.action.NumberAction" method="enableNumber">
        <result type="chain">check</result>
        </action>

        <action name="suspendNumber"
class="com.headoor.PND.action.NumberAction" method="suspendNumber">
        <result type="chain">check</result>
        </action>
    </package>
</struts>
```

Each form or operation in the web page has a corresponding Action, in order to respond to client's request, send the results to the client.

Struts2 responsible for processing and validating the data, and deliver it to the next layer.

When a user has logged in server, Struts2 will create a session from ActionContext, and put user in the session until user logout or session time expired, so that server can keep tracking user's operations:

```
public String loginUser() {
        User loginUser = userService.loadUser(user);
```

```
        if (null == loginUser || loginUser.equals(null)) {
            this.addFieldError("Loginerror", "User is not exist!");
            return "input";
        }
        if(!loginUser.getPassword().equals(user.getPassword())){
            this.addFieldError("Loginerror", "Password Wrong!");
            return "input";
        }

        Map<String, Object> attibutes =
ActionContext.getContext().getSession();
        attibutes.put("user", loginUser);
        return SUCCESS;
    }
```

Server checks user's validation for every request that user sends, if user has logged out, not exist or time expired, then return error information to the client:

```
HttpServletRequest req = ServletActionContext.getRequest();
    User loginUser = (User) req.getSession().getAttribute("user");//to
get login user
    if (null == loginUser || loginUser.equals(null)) {
        this.addFieldError("registererror", "Wrong!");
        return "input";//if user logout or time expired then returns error
    }
```

Struts2 also responsible for data validity check before delivering it to the next layer, for instance, in user registration, server must check if the username is already exist, password is valid, user has input all the information we need, etc.

Following is a segment of RegisterAction-Validation:

```
<validators>
<field name="user.userName">
   <!—String cannot be empty -->
   <field-validator type="requiredstring">
    <!—Space trim -->
    <param name="trim">true</param>
    <!—Error message -->
    <message>username cannot be empty</message>
   </field-validator>

   <!—Check the length of the string -->
   <field-validator type="stringlength">
    <param name="minLength">2</param>
```

```
    <param name="maxLength">20</param>
    <message> Username length should be between 2-18 characters </message>
   </field-validator>
</field>

<field name="user.password">
   <field-validator type="requiredstring">
    <param name="trim">true</param>
    <message>Password cannot be empty</message>
   </field-validator>

   <field-validator type="stringlength">
    <param name="minLength">6</param>
    <param name="maxLength">18</param>
    <message> Password length should be between 6-18 characters </message>
   </field-validator>
</field>

<field name="user.age">
   <field-validator type="int">
    <param name="min">1</param>
    <param name="max">150</param>
    <message> Age should be between 1 to 150 </message>
   </field-validator>
</field>
<!—Date type validation -->
<field name="user.birthday">
   <field-validator type="date">
    <param name="min">1900-01-01</param>
    <param name="max">2008-10-16</param>
    <message>Birthday should be from 1900-01-01 to 2011-5-31</message>
   </field-validator>
</field>
</validators>
```

OGNL

OGNL is the Object-Graph Navigation Language, it is a powerful expression language (Expression Language, referred to as EL), through its simple and consistent expression syntax, you can access any property of the object, call the object's method , traversing the object structure, to achieve field type conversion functions [7]. It uses the same expression to access the object.

OGNL allows us to use very simple expressions to access object layer, for example, the root object in the current environment as user1, by using expression person.address [0]. province can access the user1's person attributes' province property of the first address.

OGNL (Object-Graph Navigation Language) can easily operate the open source object property expression language, make the page more concise.

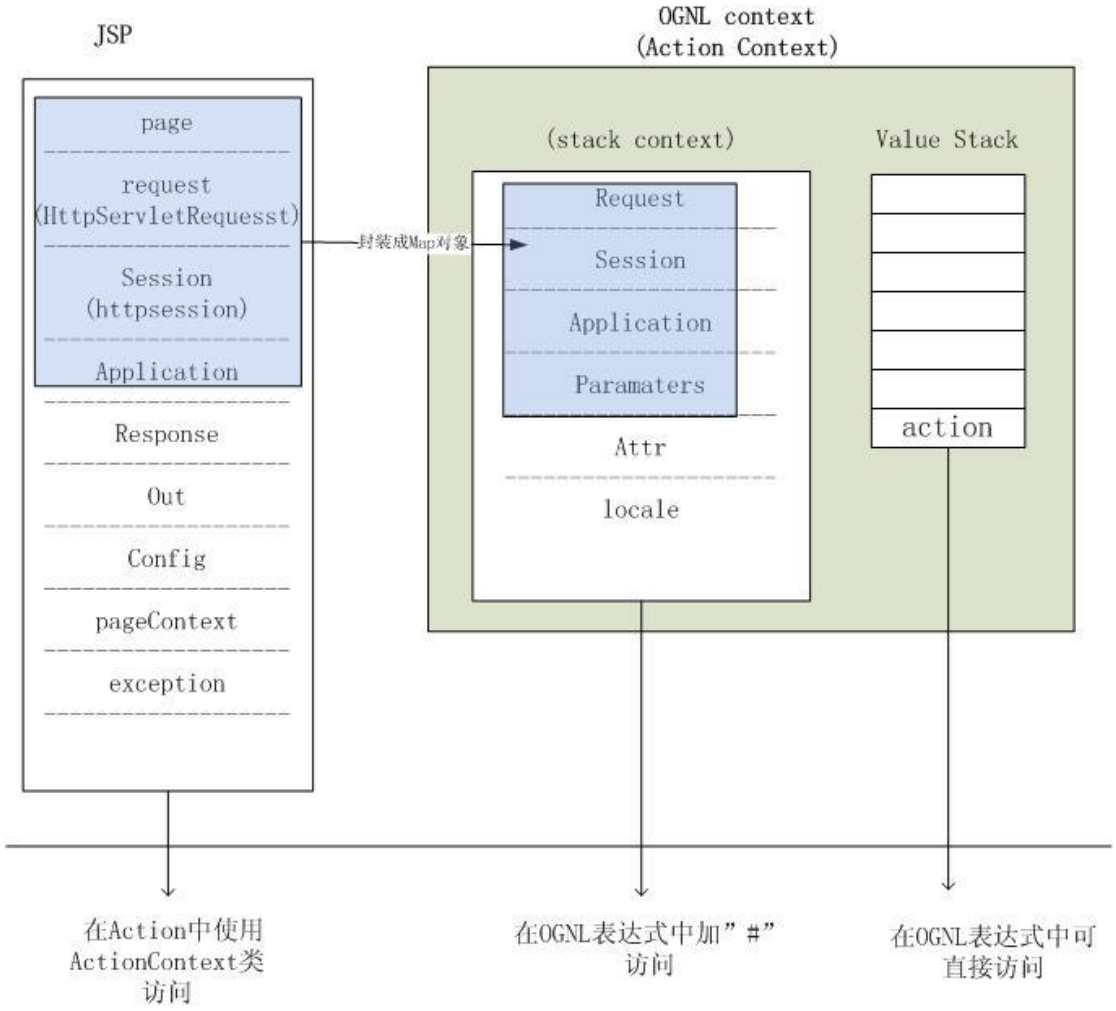
Struts 2 default expression language is OGNL, it supports object method calls, class static method calls and the value of the access, assignment and expression in series, access OGNL context (OGNL context) and ActionContext, operations for collection object.

Following is a code snippet written with OGNL and Struts2 tags:

```
<s:set name="user" value="#session.user"></s:set>
<s:if test="#user != null">
<div>
Welcome back!
<s:property value="#session.user.username" /> !<br/>
Last logged in: <s:date name="#session.user.last_login_date" />
<ul>
<li>
<a href="check">Check my telephone numbers</a>
</li>
<li>
<a href="logout">Logout</a>
</li>
</ul>
</div>
</s:if>

<s:else>
<div>
Welcome User! <br>
<s:fielderror />
```

Following figures is the relationship chart of ValueStack , StackContext and ActionContext, and sample view in my system:


JSP
OGNL context
(Action Context)
page
request
(HttpServletRequesst)
Session
(httpsession)
Application
Response
Out
Config
pageContext
exception
封装成Map对象
(stack context)
Request
Session
Application
Paramaters
Attr
locale
Value Stack
action
在Action中使用
ActionContext类
访问
在OGNL表达式中加”#”
访问
在OGNL表达式中可
直接访问

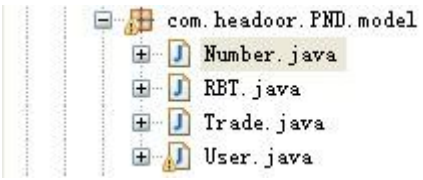
Value Stack Contents

Object	Property Name	Property Value
com.headoor.PND.action.UserAction	texts	null
	actionErrors	[]
	errors	{}
	fieldErrors	{}
	errorMessages	[]
	number_list	[com.headoor.PND.model.Number@916ab8]
	userService	com.headoor.PND.util.UserServiceHibernateImpl@15ea69f
	locale	zh_CN
	actionMessages	[]
	user	null
com.headoor.PND.action.RBTAction	texts	null
	actionErrors	[]
	errors	{}
	fieldErrors	{}
	errorMessages	[]
	locale	zh_CN
	RBTService	com.headoor.PND.util.RBTServiceHibernateImpl@80bc28
	actionMessages	[]
	rbt	com.headoor.PND.model.RBT@1c10834
	number	com.headoor.PND.model.Number@19707
	user	null
	rbt_list	null
com.opensymphony.xwork2.DefaultTextProvider	texts	null

com.opensymphony.xwork2.ActionContext.container	com.opensymphony.xwork2.inject.ContainerImpl@5c2445
com.opensymphony.xwork2.ActionContext.session	{user=com.headoor.PND.model.User@1c06a6d}
com.opensymphony.xwork2.ActionContext.actionInvocation	com.opensymphony.xwork2.DefaultActionInvocation@57f389
com.opensymphony.xwork2.util.ValueStack.ValueStack	com.opensymphony.xwork2.ognl.OgnlValueStack@145959c
session	{user=com.headoor.PND.model.User@1c06a6d}
report.conversion.errors	false
xwork.MethodAccessor.denyMethodExecution	false
struts.actionMapping	org.apache.struts2.dispatcher.mapper.ActionMapping@4cc81c
user	com.headoor.PND.model.User@1c06a6d
com.opensymphony.xwork2.ActionContext.parameters	{rbt.id=[Ljava.lang.String;@1c8e80d}

4.4.2 Entity layer:

According to the tables in the database, there are four entity classes, namely, “User”, “Number”, “RBT” and “Trade”, see figure.



For using Hibernate, we need to map the entities to the database, there are two ways, XML and Annotations. In my project, I chose to use Annotations, because it is simple, convenient and intuitive. It is a new trend.

For example, to map entity User, the following is a simplified version of code:

```java
@Entity              //to demonstrate an entity
@Table(name="T_User")//to define a specific name to the entity in database
public class User {
    private int id;
    private String name;
    private String username;
    private String password;
    private String rePassword;
    private Date register_date;
    private Date last_login_date;
    private Set<Number> numbers = new HashSet<Number>();

    @Id              //to specify the primary key
    @GeneratedValue  //auto-increase
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    @Column(unique=true)        //the data in this column is unique
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }

    @Transient //to inform Hibernate that not to create a column for it
    public String getRePassword() {
        return rePassword;
```

```
    }
    public void setRePassword(String rePassword) {
        this.rePassword = rePassword;
    }

    @OneToMany(mappedBy="user")     //the relationship between tables
    public Set<Number> getNumbers() {
        return numbers;
    }
    public void setNumbers(Set<Number> numbers) {
        this.numbers = numbers;
    }

    @Temporal(TemporalType.TIMESTAMP)
    public Date getLast_login_date() {
        return last_login_date;
    }
    public void setLast_login_date(Date last_login_date) {
        this.last_login_date = last_login_date;
    }
```

So, there are four persistent classes, hibernate responsible for the management and operation of these classes. In the Java web application based on MVC design pattern, Hibernate can be used as the model layer/data access layer. It is through the configuration file (hibernate.properties or hibernate.cfg.xml) and the mapping file (***.hbm.xml) to Java objects or PO (Persistent Object) are mapped to data in the database, and then by operating PO, to add, delete, update, search and other operations to the data in the tables.

For example, to register a new user, user needs to fill a form on the client web page and submit the form to server; after server receiving the request from client, Struts2 responsible for processing and validating the data, and deliver it to Hibernate to carry out database operations.

Following is the code snippet of adding new user:

```
public static SessionFactory sessionFactory;
    static {
        try {
            sessionFactory = new
AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }

public void addUser(User user) {
        Session session = sessionFactory.getCurrentSession();
        Transaction tx = null;
        try{
        tx = session.beginTransaction();
        user.setRegister_date(new Date());
        user.setLast_login_date(new Date());
        session.save(user);
        tx.commit();
        } catch (Exception e) {
            if (tx != null) {
                tx.rollback();
            }
        }
    }
```

Applications first call Configuration class to read configuration files and Hibernate mapping file information, and use this information to generate a SessionFactory object.

Read Configuration of Database:

```
<property
name="connection.driver_class">oracle.jdbc.driver.OracleDriver</prope
rty>
<property
name="connection.url">jdbc:oracle:thin:@localhost:1521:PND</property>
<property name="connection.username">scott</property>
<property name="connection.password">tiger</property>
<property
name="dialect">org.hibernate.dialect.OracleDialect</property>
```

Read Configuration of PO (Persistent Object):

```
<mapping class="com.headoor.PND.model.User"/>
<mapping class="com.headoor.PND.model.Number"/>
<mapping class="com.headoor.PND.model.RBT"/>
<mapping class="com.headoor.PND.model.RBT"/>
```

Then Hibernate generates a Session object from SessionFactory object, and generates Transaction object with the Session object; through the Session object's get(), load(), save(), update(), delete() and saveOrUpdate() and other methods to load

on the PO, to save, update, delete or other operations; in the case of queries, create a Query object through the Session object, and then use the Query object to execute queries; if no exception, Transaction object will submit the result of these operations to the database.

Following is the query which Hibernate created when a use is registering:

```
Hibernate:
    insert
    into
        T_User
        (address, birthdate, last_login_date, name, other_information,
password, region, register_date, username, usertype)
    values
        (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
```

And this is the Query of checking telephone number information:

```
Hibernate:
    select
        user0_.id as id0_,
        user0_.address as address0_,
        user0_.birthdate as birthdate0_,
        user0_.last_login_date as last4_0_,
        user0_.name as name0_,
        user0_.other_information as other6_0_,
        user0_.password as password0_,
        user0_.region as region0_,
        user0_.register_date as register9_0_,
        user0_.username as username0_,
        user0_.usertype as usertype0_
    from
        T_User user0_
    where
        user0_.username=?
lyz9
Hibernate:
    select
        numbers0_.user_id as user16_2_,
        numbers0_.id as id2_,
        numbers0_.id as id1_1_,
        numbers0_.c_p as c2_1_1_,
        numbers0_.geographical as geograph3_1_1_,
        numbers0_.last_operator as last4_1_1_,
        numbers0_.last_user_id as last5_1_1_,
```

```
    numbers0_.number_type as number6_1_1_,
    numbers0_.operator as operator1_1_,
    numbers0_.other_information as other8_1_1_,
    numbers0_.partner as partner1_1_,
    numbers0_.rbt_id as rbt15_1_1_,
    numbers0_.region as region1_1_,
    numbers0_.register_date as register11_1_1_,
    numbers0_.status as status1_1_,
    numbers0_.telephone_number as telephone13_1_1_,
    numbers0_.types as types1_1_,
    numbers0_.user_id as user16_1_1_,
    rbt1_.id as id2_0_,
    rbt1_.rbt_content as rbt2_2_0_,
    rbt1_.rbt_name as rbt3_2_0_,
    rbt1_.rbt_prise as rbt4_2_0_
from
    T_Number numbers0_
left outer join
    RBT rbt1_
        on numbers0_.rbt_id=rbt1_.id
where
    numbers0_.user_id=?
```

4.5 Database:

There are four parts in phone number database:

1) Telephone Numbers
2) Users
3) RBT
4) Operation Record

So there are four tables in database: T_Users, T_Numbers, T_RBT and T_Trade.

According to the request of system scope and context, each table has several segments:

Telephone numbers:

1. Id (int, primary key)
2. telephone_number (varchar2, unique)

3. user_id (int, foreign key)
4. last_user_id (int, foreign key)
5. RBT_id (int, foreign key)
6. operator (varchar2)
7. last_operator (varchar2)
8. status (Available, Not Available..etc)(varchar2)
9. register_date (Date)
10. types (Voice, Data, Fax, M2M..etc)(varchar2)
11. number_type (Gold, Silver, Normal, etc)(varchar2)
12. geographical (varchar2)
13. region (varchar2)
14. cooperate or private (varchar2)
15. partner (varchar2)
16. other_information (varchar2)

Users:

1. id (int, primary)
2. name (first middle and last name)(varchar2)
3. birthday (date)
4. telephone_number (varchar2)
5. username (varchar2)
6. password (varchar2)
7. address (varchar2)
8. region (or nationality)(varchar2)
9. other_information (varchar2)
10. usertype (varchar2)
11. register_date (date)
12. last_login_date (date)

RBT

1. id (int, primary key)
2. rbt_name (varchar2)
3. rbt_prise (varchar2)
4. rbt (varchar2)

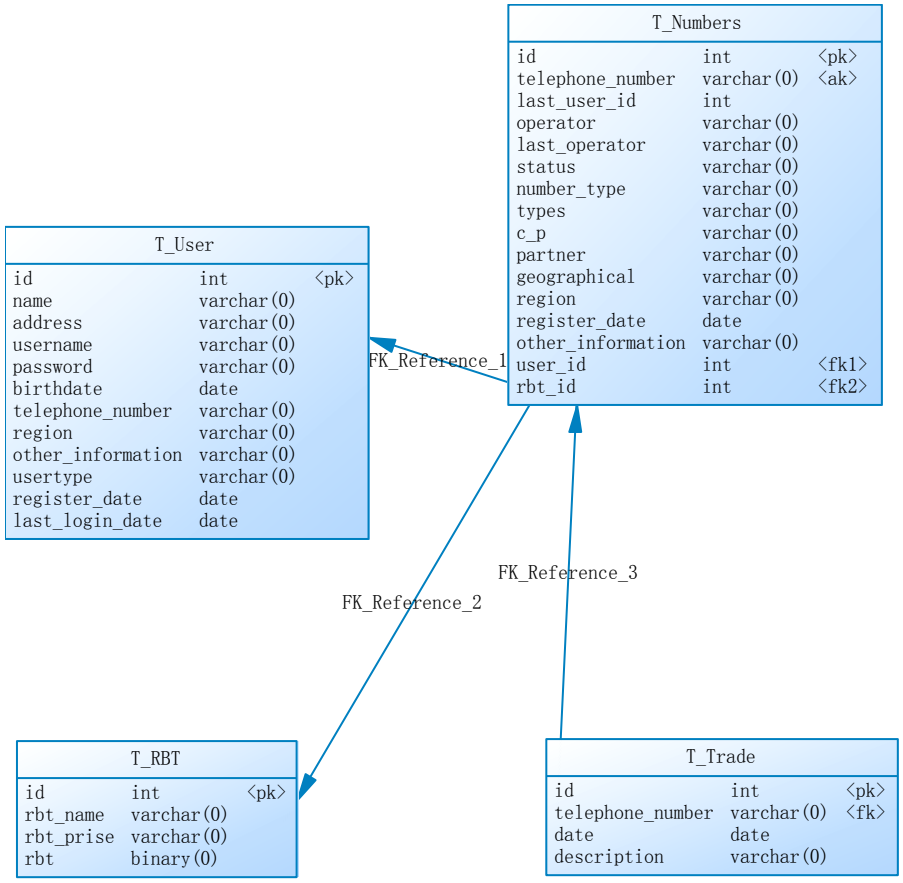
Operation Record

1. id (int, primary key)
2. telephone_number (varchar2, foreign key)
3. date (date)
4. description (varchar2)

Varchar2 is a data type proper to Oracle Database, it is Variable-length character data, and its maximum length is 4000B. The biggest difference with char data type is the varchar2 automatically filter out spaces.

Considering that a user can own more than one telephone numbers, but a phone number belongs to only one user, so the relationship between users and telephone numbers is (1:n), and the relationship between telephone numbers and ring back tone is (1:1).

E-R model of phone number database:



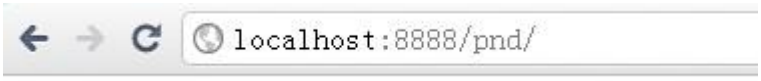
5. Results

Sample Run:

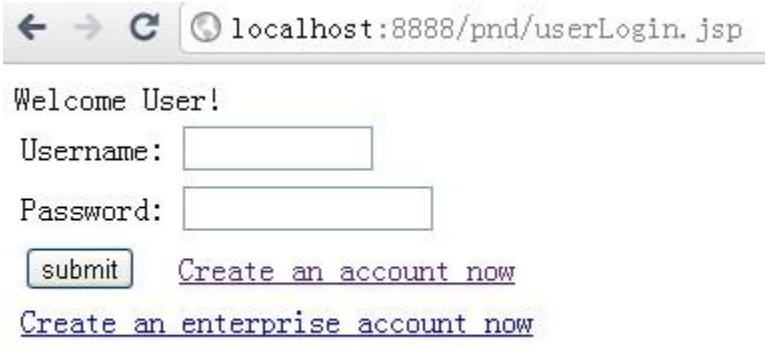
1) Setup and configure Tomcat server and Oracle database, run the system:

```
2011-6-14 15:48:54 org.apache.coyote.http11.Http11AprProtocol start
信息: Starting Coyote HTTP/1.1 on http-8888
2011-6-14 15:48:54 org.apache.coyote.ajp.AjpAprProtocol start
信息: Starting Coyote AJP/1.3 on ajp-8009
2011-6-14 15:48:54 org.apache.catalina.storeconfig.StoreLoader load
信息: Find registry server-registry.xml at classpath resource
2011-6-14 15:48:54 org.apache.catalina.startup.Catalina start
信息: Server startup in 91016 ms
```

2) Index page:



3) Login page:

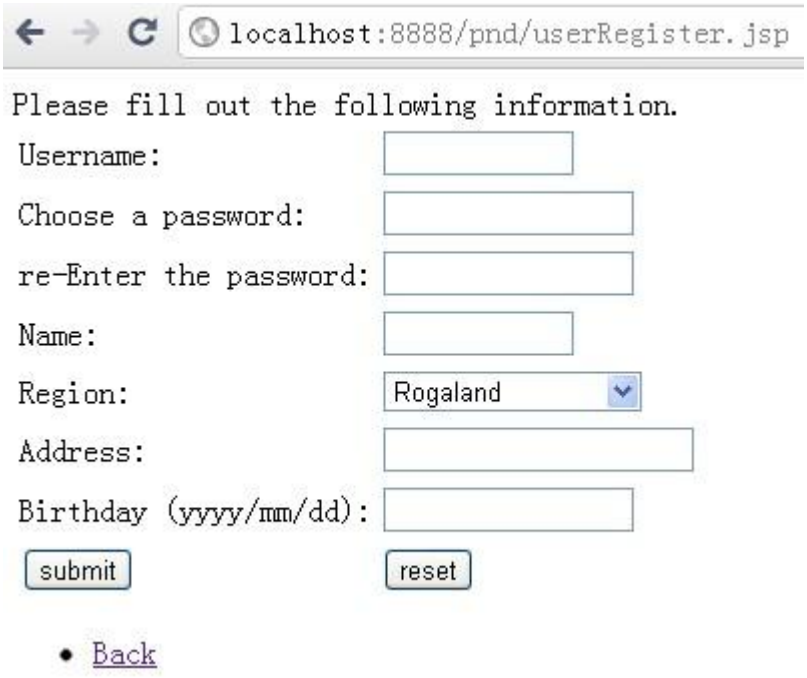


Above is the login page for user, if username that user input is not exist or password is wrong, a error message will be shown as following:

figure 1 username not exist

figure 2 wrong password

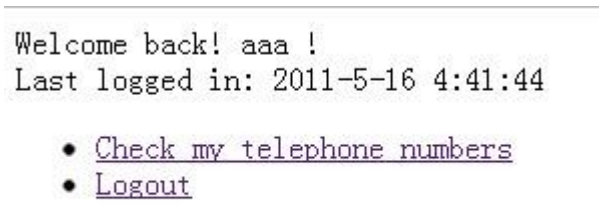
4) Registration page:



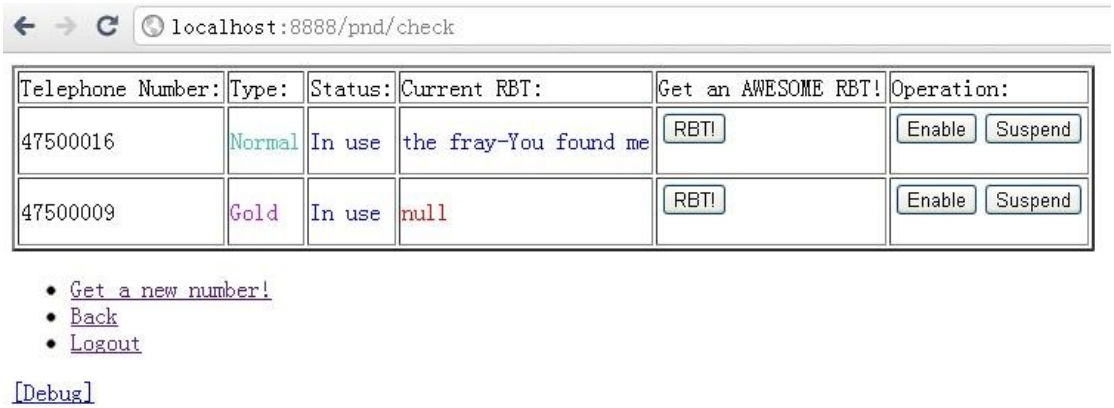
Here user will fill some personal information, and submit it to the server, after the validity check, user will get a new account, and user's information will be registered into system database:

name	other_information	password	region	register_date	username	usertype	last_login_date
altibox	(NULL)	123	Rogaland	2011-05-16 10:19:32	xxx	Enterprise	2011-05-16 10:19:32
lyz	(NULL)	123	Rogaland	2011-05-16 12:18:26	polar9	Individual	2011-05-16 12:18:26
lyz	(NULL)	123	Rogaland	2011-05-16 00:39:38	polar2	Individual	2011-05-16 04:52:42

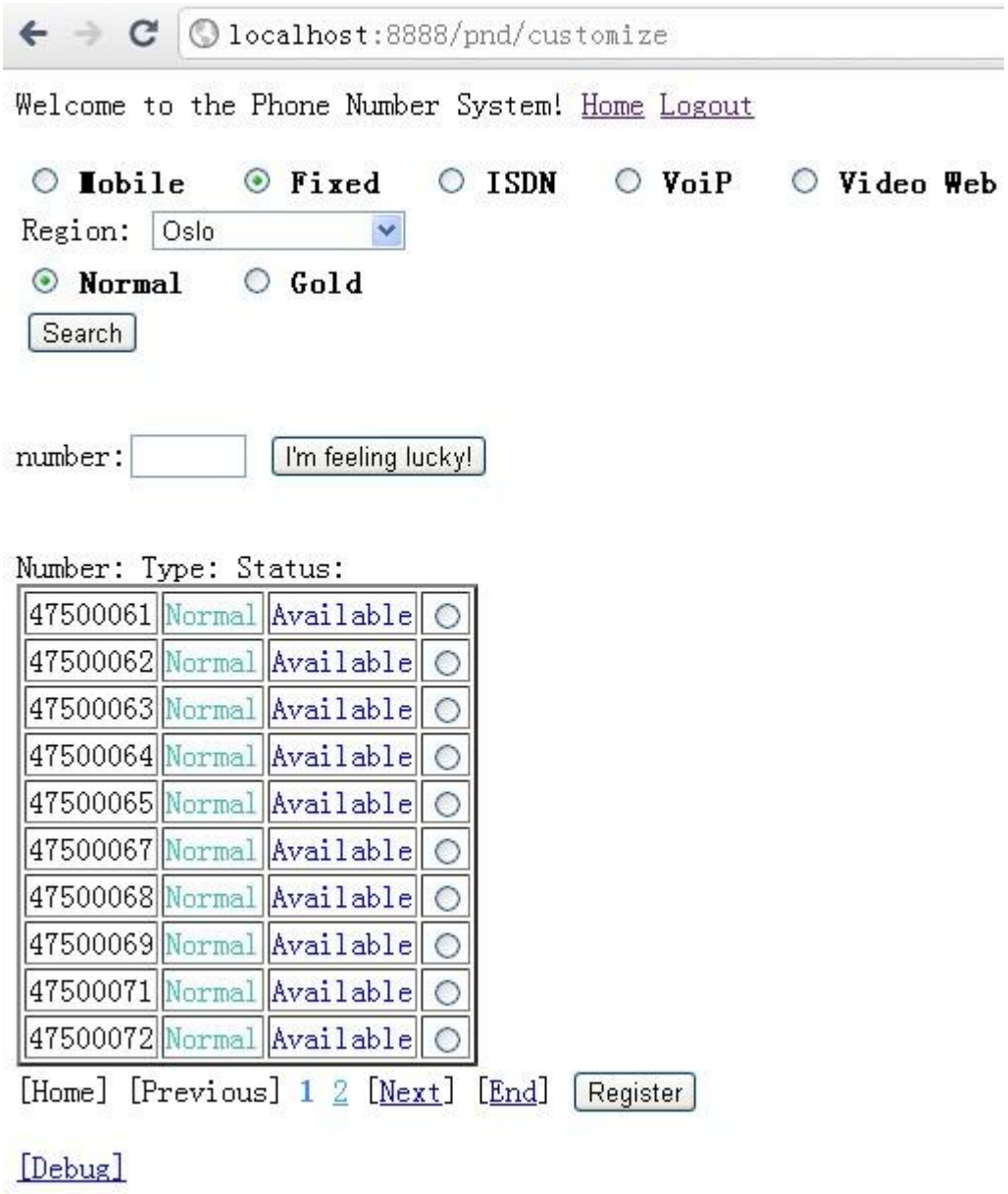
5) User information page:



By clicking link "Check my telephone numbers", users can check their telephone number information in their account.



6) Telephone number registration page:



Also users can input their favorite numbers to query.

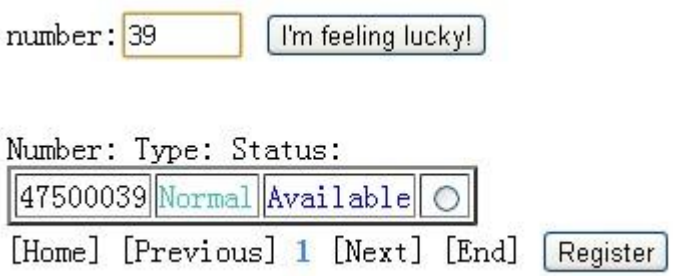


figure 3 telephone number available

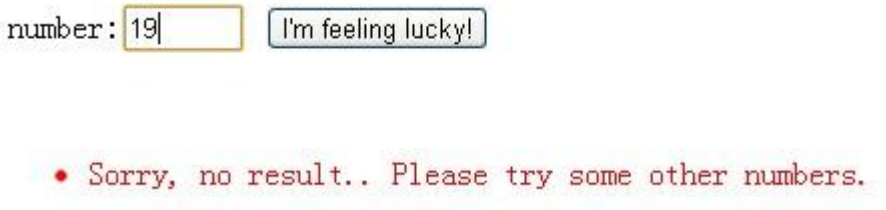
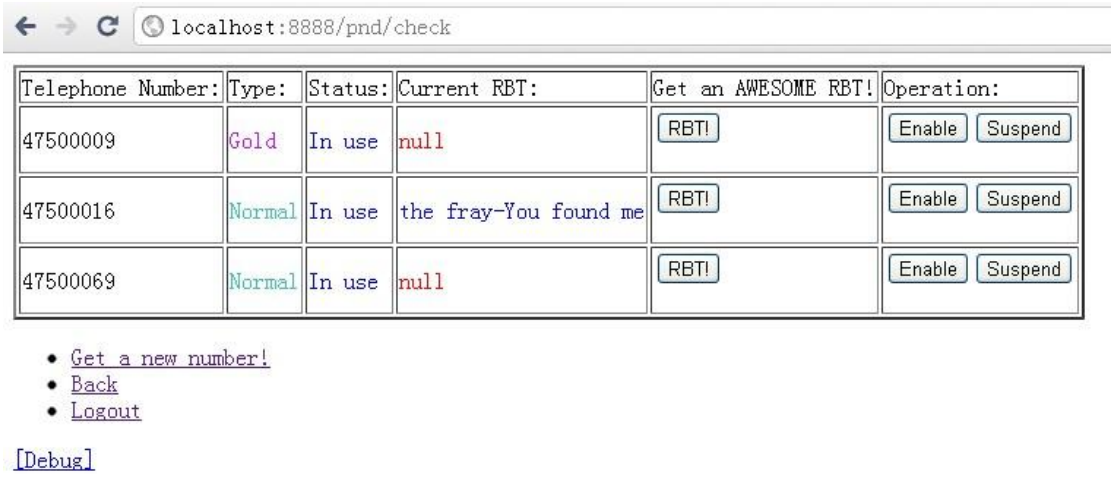
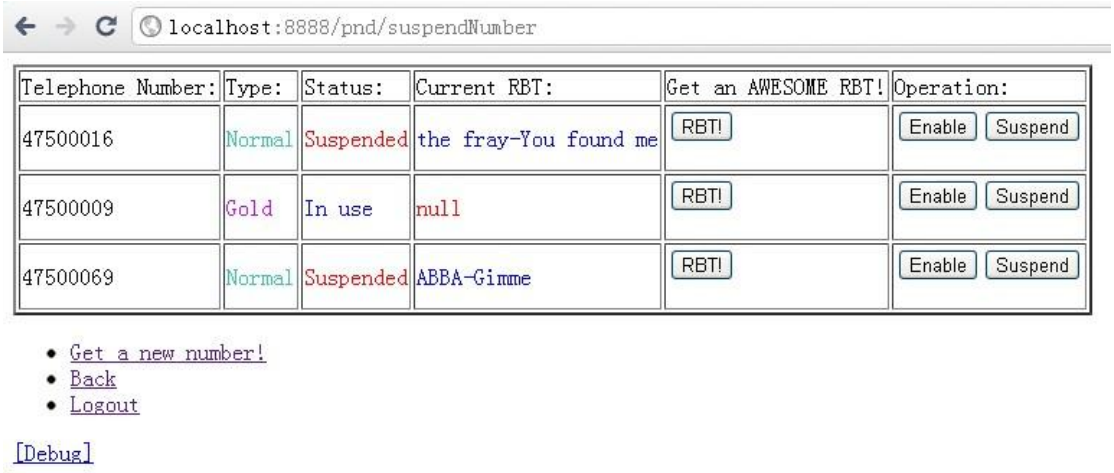
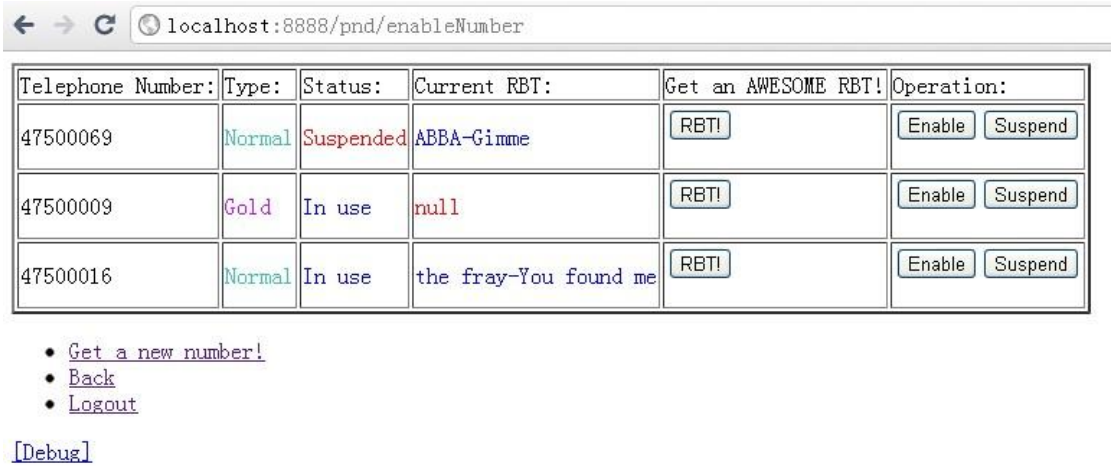


figure 4 no number available

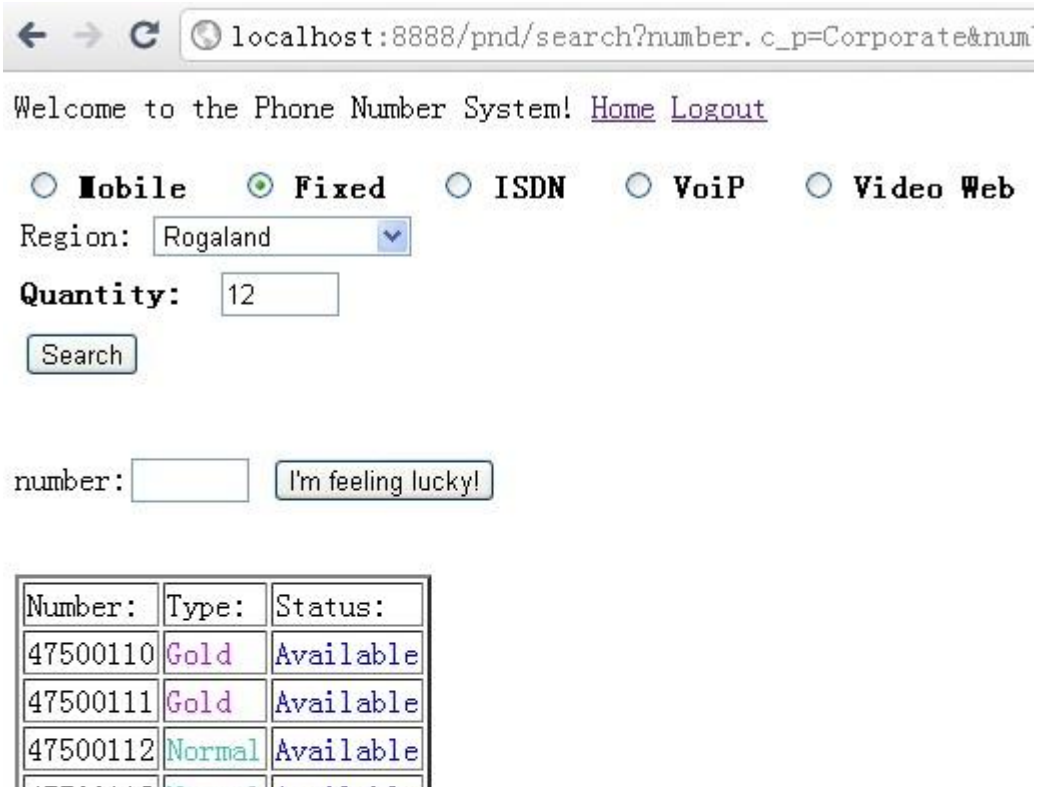
If successfully registered, telephone number will be bound with user's account, they can Enable or Suspend their telephone numbers, and operation will be recorded to inform administer.



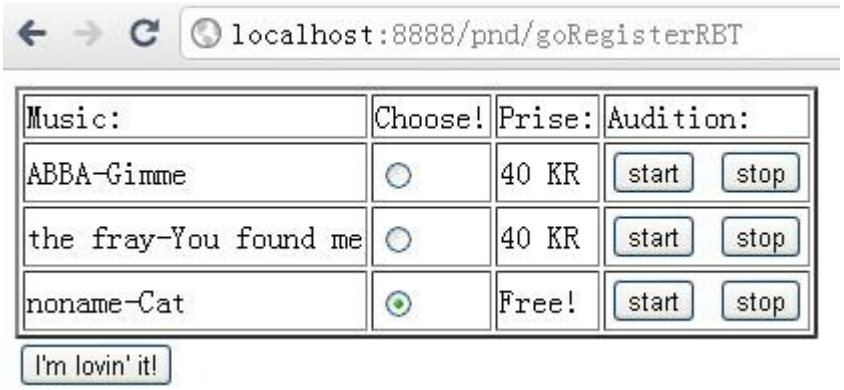




For enterprise users, they can search a certain number of consecutive telephone numbers from the system, and register them at one time.



7) Ring back tone page:



In this page, users can play and choose a ring back tone for their telephone numbers, also they can find their favorite song by searching the same of music.

6. Discussion

6.1 Originality of this work

I created a survey web site (see Appendix-A) while I was doing this project, and sent it to hundreds of people, from the analysis of the results, I found that people on the communication service requirements are raising, individual demand is increasing, we need better systems to meet this trend, this is what this project trying to do, providing a new telephone number system that

1) Users can apply telephone numbers and manage their telephone numbers by staying indoors.

2) Classify telephone numbers, give the users more choice.

3) Operator can manage telephone number series systematically, efficiently and automatically.

4) Users can chose colorful ring back tone for their telephone numbers to make their numbers more personality.

6.2 Relevant works

Each operator has its own system, but most of them cannot meet user's requirement perfectly, we need to continually improve our system, try our best to satisfy the customer. For example, CRBT system is already exists and popular for a long time in East Asia, SK and HUAWEI provide this system, but currently there is no operators provide this service in Norway.

6.3 Learning experience

I learned a lot from designing such a system. For me, J2EE is almost brand new knowledge. Although I learned some knowledge about Java and database, practice and build a whole project is totally another thing. I learned how to design web page (HTML, JavaScript, CSS, JSP, Struts2 tags), how to build server (Struts2, Hibernate, some Spring and JUnit), how to use tomcat, Oracle database and MySQL database.

Designing an extendable frame work is another challenge I met. The problems are, what features I want to extend in future, and how much extendable is enough? Since no design is perfect, I cannot make a fully extendable design that meet every future needs, it is like perperual motion, is impossible. The only design choice is to design it with respect to current requirements and the most possible new requirements. So I just listed all I need to support right now, then design the system model, without too much thinking or future.

6.4 Limitations

There is much to be desired in this project, due to time and knowledge is limited, a lot of functions and ideas have not been achieved. The system needs to be stronger to handle any possible request coming from users.

No performance test. Due to time limitation, I only performed some basic tests to make sure the system runs correctly. To make it really usable, more tests, especially performance test should be performed in order to reveal more problems in design and implementation.

Owing to Altibox company currently does not have its own switch machine and some other technical limitations, I only finished the web part and database part of CRBT system.

6.5 Further work

1. Security enhancement.
2. Performance test.
3. More operations for users and more aesthetically pleasing interface.
4. Telephone number port-in and port-out (need API support).
5. Improve the RBT system.

7. Reference

[1] http://www.dialogic.com/solutions/mobile-vas/mobile-crbt.htm

[2] http://struts.apache.org/2.2.1/index.html

[3] http://en.wikipedia.org/wiki/Struts2

[4] http://www.hibernate.org/about

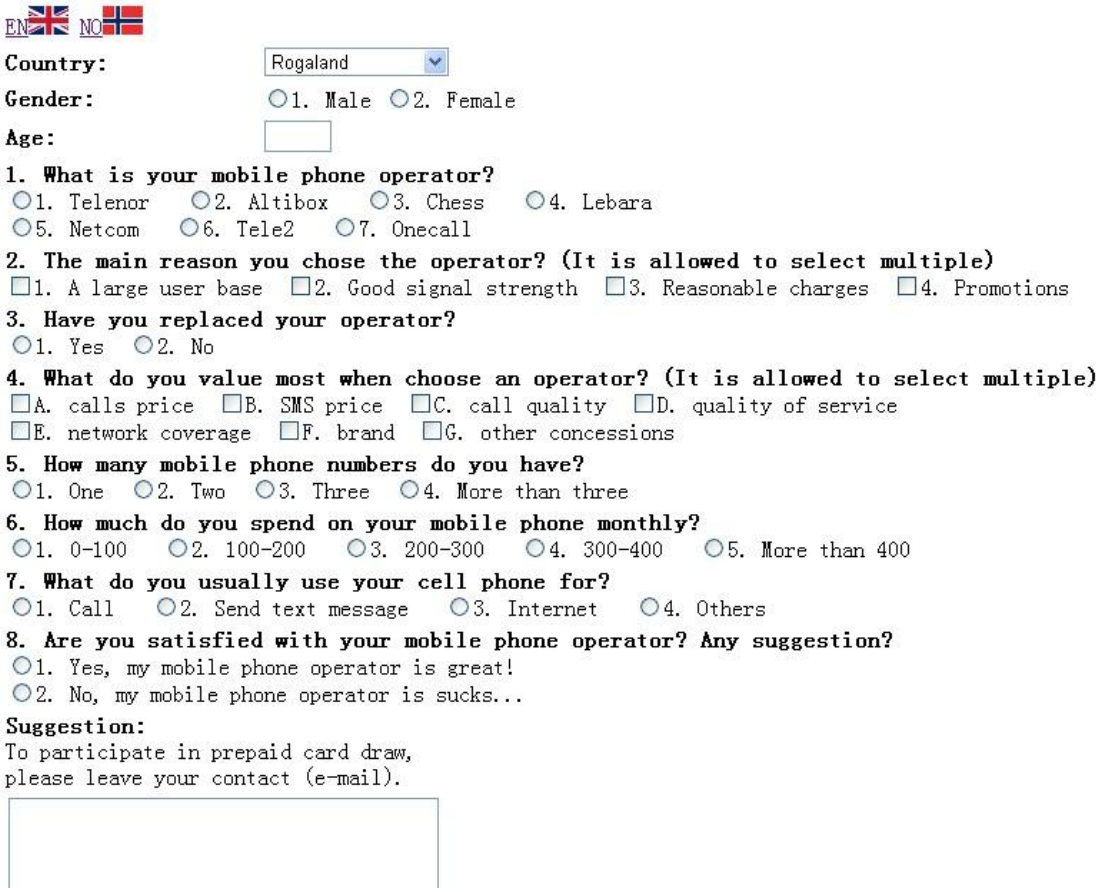
[5] http://en.wikipedia.org/wiki/Java_(programming_language)

[6] http://en.wikipedia.org/wiki/Apache_Tomcat

[7] http://www.hudong.com/wiki/OGNL

8. Appendix-A

Survey and Result:



EN NO

Country: Rogaland

Gender: ○1. Male ○2. Female

Age:

1. What is your mobile phone operator?
○1. Telenor ○2. Altibox ○3. Chess ○4. Lebara
○5. Netcom ○6. Tele2 ○7. Onecall

2. The main reason you chose the operator? (It is allowed to select multiple)
☐1. A large user base ☐2. Good signal strength ☐3. Reasonable charges ☐4. Promotions

3. Have you replaced your operator?
○1. Yes ○2. No

4. What do you value most when choose an operator? (It is allowed to select multiple)
☐A. calls price ☐B. SMS price ☐C. call quality ☐D. quality of service
☐E. network coverage ☐F. brand ☐G. other concessions

5. How many mobile phone numbers do you have?
○1. One ○2. Two ○3. Three ○4. More than three

6. How much do you spend on your mobile phone monthly?
○1. 0-100 ○2. 100-200 ○3. 200-300 ○4. 300-400 ○5. More than 400

7. What do you usually use your cell phone for?
○1. Call ○2. Send text message ○3. Internet ○4. Others

8. Are you satisfied with your mobile phone operator? Any suggestion?
○1. Yes, my mobile phone operator is great!
○2. No, my mobile phone operator is sucks...

Suggestion:
To participate in prepaid card draw,
please leave your contact (e-mail).

Question 1: 1. 40% 2. <1% 3. 32% 4. 8% 5. 16% 6. <1% 7. <1%

Question 2: A. 16% B. 24% C. 44% D. 24%

Question 3: 1. 48% 2. 52%

Question 4: A. 72% B. 48% C. 4% D. 36% E. 8% F. 4% G. 8%

Question 5: 1. 68% 2. 24% 3. <1% 4. 4%

Question 6: 1. 28% 2. 32% 3. 24% 4. 4% 5. 12%

Question 7: 1. 44% 2. 48% 3. <1% 4. 4%

Question 8: 1. 84% 2. 16%