

Saya Mini Project Report

Saya Web Interface

Final Report

Hadas Shveky 40316531

David Stein 31410145

Guidance:

Professor Shlomi Dolev

Mr. Michel Orlov

Ben-Gurion University of the Negev,

Computer Science Department

December 2007

Table Of Contents

Introduction.	1
Architecture and platform overview.	2
The Database.	3
Overview.	3.1
:The Database Definition.	3.2
:The Java Interface.	4
Java Database Interface.	4.1
:Database Update Script.	4.2
:Building and running the java programs.	4.3
Web Interface.	5
Overview.	5.1
Web authentication.	5.2
:Login and registration.	5.3
.Main page.	5.4
Summary and future development.	6
Reference.	7

1. Introduction

Saya is a robot receptionist of the Ben-Gurion University Computer Science department. In this mini project we designed and implemented a Database of CS users and visitors with support for messages vacation entries. a web interface for authentication, registering and login, and a java interface for updating the DB and retrieving information.

2. Architecture and platform overview

The Database name/scheme name is *sayaweb1*, located on the mysql server on *the webdev* host (database user is also *sayaweb1*).

The web site was implemented using html and php 4, running on the CS department apache web server. It is installed and accessible at: <http://www.cs.bgu.ac.il/~steind/saya/login.php>

The java interface was implemented using the mysql JDBC driver which is required in the java classpath at runtime.

3. The Database

3.1.Overview

In order to manage Saya's information, we implemented a Database scheme using MySQL database, and manage the data stored in it with java JDBC interface.

3.2.The Database Definition:

The database definition is provided in the DBdefinition.sql file.
The scheme of the database is defined as follows:

```
CREATE SCHEMA sayaweb1;
```

The database contains 7 tables, which are defined as follows:

Table *person*:

- Contains information about a person who is a CS member.

- Contains 10 fields of data.
- Primary key is *username*.
- Definition:

```
CREATE TABLE sayaweb1.person(
  username VARCHAR(20) NOT NULL,
  firstName VARCHAR(15) NOT NULL,
  lastName VARCHAR(15) NOT NULL
  email VARCHAR(45) DEFAULT NULL,
  homepage VARCHAR(45) DEFAULT NULL,
  office VARCHAR(5) DEFAULT NULL,
  building VARCHAR(5) DEFAULT NULL,
  phone VARCHAR(15) DEFAULT NULL,
  fax VARCHAR(15) DEFAULT NULL,
  box VARCHAR(5) DEFAULT NULL,
  PRIMARY KEY (username)
)
ENGINE = InnoDB;
```

Table *message*:

- Contains messages who are sent to a certain person/visitor
- Contains 5 fields of data.
- Primary key is *msgID,username*.
- Definition:

```
CREATE TABLE saya.message (
  msgID MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
  username VARCHAR(20) NOT NULL,
  sender VARCHAR(20) ,
  date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  msgBody TEXT,
  PRIMARY KEY (msgID, username)
)
ENGINE = InnoDB;
```

Table *reception_hours*:

- Contains information about office reception hours of a certain person.
- Contains 4 fields of data.
- Primary key is *username*.

- Foreign key constraint to *username* field in *person* table.
- Definition:

```
CREATE TABLE saya.reception_hours (
  username VARCHAR(20) NOT NULL,
  officeDay TINYINT(3) UNSIGNED NOT NULL,
  startTime FLOAT UNSIGNED NOT NULL,
  endTime FLOAT UNSIGNED NOT NULL,
  PRIMARY KEY (username),
  CONSTRAINT FK_reception_hours FOREIGN KEY FK_reception_hours
  (username)REFERENCES person (username)
  ON DELETE CASCADE
  ON UPDATE CASCADE
)
ENGINE = InnoDB;
```

Table *titles*:

- Contains all the titles of a certain person.
- Contains 2 fields of data.
- Primary key is *username,title*.
- Foreign key constraint to *username* field in *person* table.
- Definition:

```
CREATE TABLE saya.titles (
  username VARCHAR(20) NOT NULL,
  title VARCHAR(45) NOT NULL,
  PRIMARY KEY (username, title),
  CONSTRAINT FK_titles FOREIGN KEY FK_titles (username)
  REFERENCES person (username)
  ON DELETE CASCADE
  ON UPDATE CASCADE
)
ENGINE = InnoDB;
```

Table *vacation*:

- Contains information about vacations of a certain person.
- Contains 3 fields of data.
- Primary key is *username,startDate,endDate*.
- Foreign key constraint to *username* field in *person* table.

- **Definition:**

```
CREATE TABLE saya.vacation (
    username VARCHAR(20) NOT NULL,
    startDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    endDate TIMESTAMP DEFAULT '0000-00-00 00:00:00',
    PRIMARY KEY (username, startDate, endDate),
    CONSTRAINT FK_vacations FOREIGN KEY FK_vacations(username)
        REFERENCES person (username)
        ON DELETE CASCADE
        ON UPDATE CASCADE
)
ENGINE = InnoDB;
```

Table *code_mapping*:

- Contains information authentication code mapping to a bluetooth ID String.
- Contains 3 fields of data.
- Primary key is *AuthentCode,BlueToothID*.
- **Definition:**

```
CREATE TABLE saya.code_mapping (
    authentCode INTEGER UNSIGNED NOT NULL,
    blueToothID VARCHAR(50) NOT NULL,
    expirationDate TIMESTAMP,
    PRIMARY KEY (authentCode, blueToothID)
)
ENGINE = InnoDB;
```

Table *visitor*:

- Contains information of a person who is a visitor of a CS member.
- Contains 3 fields of data.
- Primary key is *username, password*.
- **Definition:**

```
CREATE TABLE saya.visitor (
    username VARCHAR(20) NOT NULL,
    password VARCHAR(45) NOT NULL,
    visitorHost VARCHAR(20) NOT NULL,
    PRIMARY KEY (username, password)
)
ENGINE = InnoDB;
```

4. The Java Interface:

4.1. Java Database Interface

The sayaweb.dinterface package contains 4 classes:

`DataBaseInterface.java` - This class exposes an interface for managing data in the database. This object should first connect to the database, and eventually disconnect from it. The API includes:

```
* This method recieves a username and gets his office number. Returns a
public String getPersonOfficeAndBuilding(String username) {}

* This method receives a username and gets his office hours. Returns a
public Vector<String> getPersonReceptionHours(String username) {}

* This method recieves a username and checks what is the return date from
public Date backToWorkDate(String username) {}

* This method recieves a username and checks if he's out of office (i.e. if
public boolean isOutOfOffice(String username) {}
```

```

    * This method receives a username and gets his titles. Returns a vector.
public Vector<String> getPersonTitles(String username) {}

    * This method receives a username and gets his email address. Returns his.
public String getPersonEmail(String username) {}

    * This method receives a username and gets his phone number. Returns his.
public String getPersonPhone(String username) {}

    * This method receives a username and gets his fax number. Returns his fax.
public String getPersonFax(String username) {}

    * This method receives a username and gets his box number. Returns his box.
public String getPersonBox(String username) {}

    * This method receives a message for username from sender containing text.
public void addMessage(String username, String sender, String text) {}

    * This method receives a username and gets his messages. Returns a vector.
public Vector<Message> getMessages(String username) {}

    * This method receives a username and a msgID from his mailbox. Deletes.
public int deleteMessage(String username, int id) {}

    * This method receives a username and deletes all messages from his.
public int deleteAllMessages(String username) {}

```

```

    * This method receives an authenticationCode, BlueToothID, expirationDate.
    public void addCodeMapping(int authentCode, String BlueToothID,

    * This method receives an authentication code for the BT system and returns
    public String getBlueToothID(int authentCode) {

    * This method checks if a certain username appear in Database (i.e. appear
    public boolean isInDataBase(String username) {

    * This method updates the first name for a certain username in the
    public void updateFirstName(String username, String fname) {

    * This method updates the last name for a certain username in the Database.
    public void updateLastName(String username, String lname) {

    * This method updates the email for a certain username in the Database.
    public void updateEmail(String username, String email) {

    * This method updates the homepage for a certain username in the Database.
    public void updateHomepage(String username, String homepage) {

    * This method updates the office for a certain username in the Database.
    public void updateOffice(String username, String office) {

    * This method updates the building for a certain username in the Database.
    public void updateBuilding(String username, String building) {

    * This method updates the phone for a certain username in the Database.
    public void updatePhone(String username, String phone) {

    * This method updates the fax for a certain username in the Database.
    public void updateFax(String username, String fax) {

    * This method updates the box for a certain username in the Database.
    public void updateBox(String username, String box) {

    * This method updates the info of reception hours for a certain username.
    public void updateReceptionHours(String username, OfficeHour[] officeHours) {

    * This method deletes all rows of a certain username from tableName in the
    public void deleteRows(String username, String tableName) {

    * This method adds reception hour info (day,startTime,endTime) for a
    public void addReceptionHour(String username, String officeDay,

```

`Person.java` - This class holds information of a person. This Class corresponds to table *person* in the database.

`Message.java` - This class is a structure using to hold information of a message. This Class corresponds to table *message* in the database.

`OfficeHour.java` - This class is a structure using to hold information of a person's office reception hours. This Class corresponds to table *reception_hours* in the database.

4.2.Database Update Script:

It is located under the sayaweb1 package with the name UpdateScript.

This script is written in order to update daily the information of CS members in the database. The script is written in Java, and does the following steps:

1. Get update file - Via URL address, gets a file with updated information of the CS members. This file is updated daily from the department database. The file is in xml format, and is copied locally in the same format.
2. Parse file – The locally stored xml file is parsed in order to get the updated information from it.
3. Update database – The database is updated using the information extracted from file.
4. Create txt file – A txt file named "res.txt" is created locally, containing a list of usernames of CS members and their room number. The file is to be used by the Saya dictionary application (developed by a different project).

❖ Notes:

1. The update script is valid only for a CS member who already exists in the database, i.e. who has a line representing him/her in the *person* table. New cs members must be added to the *person* in the first time, to be updated routinely by the Update script. A new CS user can add himself to the Person table using the web interface as described below.

4.3. Building and running the java programs:

The sources are under the src directory. The project can be built using Apache Ant program. An Ant configuration file, build.xml, is provided. It saves the compile java classes under the *classes* folder and packages them into sayaweb.jar file under the lib folder.

To run the java update script or use the DataBaseInterface class be sure to add the mysql JDBC driver to your java classpath (provided here as mysql.jar).

5. Web Interface

5.1. Overview

The web interface was implemented in html and php4 code. It is located under the *htdocs* folder

It has the following files:

- login.php - login page.
- auth.php - authentication handling
- error.php - error page
- main_menu.php - the CS user main page.
- main_visitor.php - the visitor main page.
- new_visitor.php - a form for registering a new visitor who entered a valid authentication code.
- logout.php - logout from session.
- scripts/config.php - Database connection details.
- script/opendb.php - Database connection.
- script/closedb.php - Database disconnection.

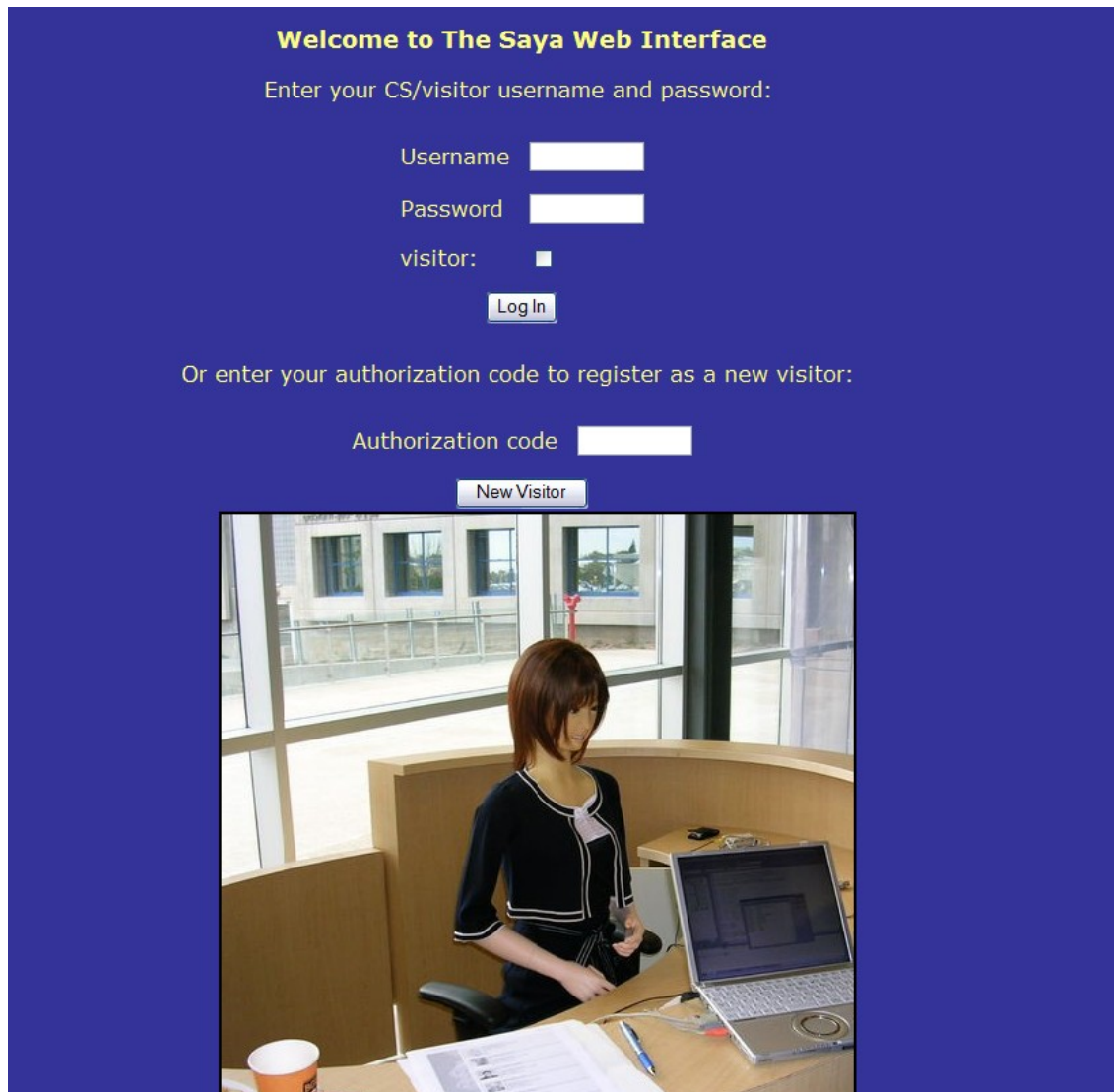
mysql_dialog.php and mysqlldata.php – these sources for handling mysql queries and displaying them are taken from the AdminPro class and MySQL List Data projects respectively, and modified to our needs. Both projects have GPL License. See Reference below.

5.2. Web authentication

The site and all its pages are protected with a session check. Sessions are authenticated in the following way: CS users are authenticated with their CS user name and password. It is checked with the ypmatch utility with the password encryption.

Visitors are authenticated with the username and password, unencrypted, in the visitor table. All mysql queries are escaped with the php mysql_real_escape_string function to minimize the risk of an sql injection.

5.3.Login and registration:



The login page allows a CS member to login using his username/password credentials. If a CS user is authenticated with his username/password but he is not in the person DB table,

he gets the following page, allowing him to add himself to the Person table:



No other data is required by the user, as it will be updated the next time the Java UpdateScript is run.

A visitor must check the visitors checkbox in order to login. He is authenticated with the visitor table username/password. A new visitor can register himself entering a valid authentication code in the login page and clicking on "New Visitor" button. The authentication code is checked in the code_mapping table for correctness and date

expiration, and the visitor gets a form to enter his username and password:



Saya Web Interface.
◆ Welcome

Please enter username and password to register new user:

user name:

Password:

>

The username is checked for uniqueness against the Person and Visitor tables.

5.4.Main page.

A registered user main page allows him to view and delete messages and to view, delete and add vacation entries:

[logout](#)

Saya Web Interface.
 ♦ Welcome back David Stein.

Your messages:

Messages					
msg ID	username	Sender	Date	Message text	Action
7	steind	Saya	2007-12-29	Welcome to Saya web interface	Delete

Your vacations:

Vacations			
User	Vacation Start	Vacation End	Action
steind	2008-02-05	2008-02-23	Delete

Enter new vacation here:

Start date (yyyy-mm-dd):

End date (yyyy-mm-dd):

>

The visitors main page allows him to view and delete messages:

[logout](#)

Saya Web Interface.
 ♦ Welcome back david.

Your messages:

Messages					
msg ID	username	Sender	Date	Message text	Action
5	david	Saya	2007-12-29	Welcome to Saya web interface	Delete

6. Summary and future development

We developed a Database and java interfaces for accessing and modifying the users and visitors tables, We also implemented a web site for managing authentication and login issues, new user/visitor registration and main user/visitor page for viewing and deleting messages and vacation entries.

Future development should include integrating with other saya projects, and extending the functionality of this project:

- The text file res.txt created by the UpdateScript should be used by saya to respond to questions about faculty members room numbers.
- Saya should use the isOutOffice(username) method to answer about vacation status of CS users, as well as other information retrievable from the database.
- An authentication code, i.e using Bluetooth protocol should invite new visitors, updating the code_mapping table accordingly.
- The messages table could be changed to support files of voice messages, and the website should allow downloading/playing them. i.e a Person could record a message by asking the saya robot to leave a message to a user.
- More useful information could be shown and/or inserted in the users/visitors main page.

7. Reference

Some php code was used and modified from the following GPL Licensed projects:

mysql_dialog.php was taken and modified from AdminPro class:

<http://www.phpclasses.org/browse/package/1830.html>

amysqlldata.php was taken and modified from MySQL List Data

<http://www.phpclasses.org/browse/package/3020.html>