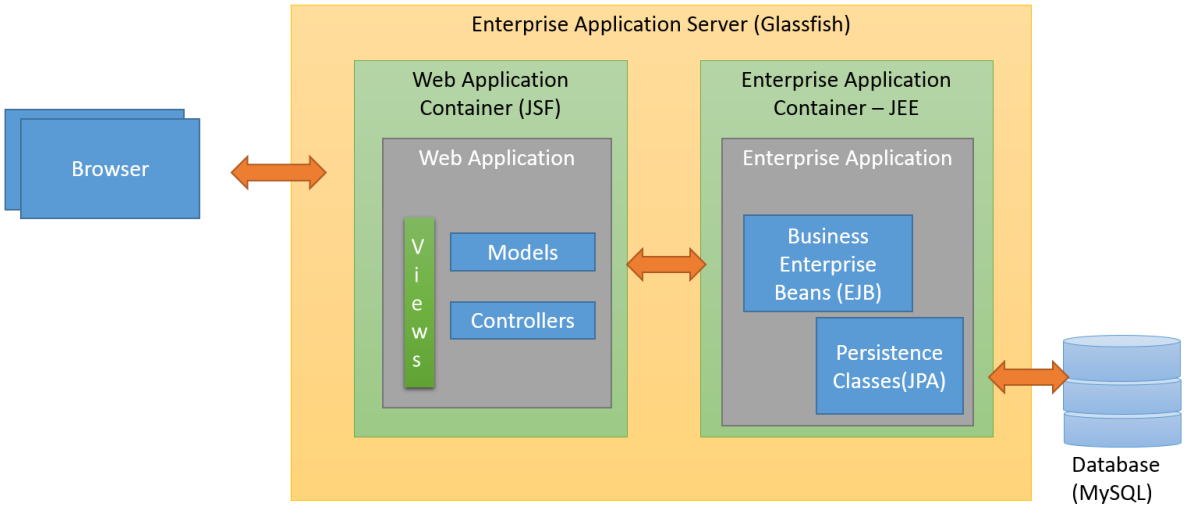
Secure Bank Pilot Project- Secure Software Engineering

By ..
Raghunath Nandyala(Raghu.nandyala@ttu.edu)
Nagendra Varma Totakura (nagendravarma.totakura@ttu.edu)
Siva Kiran Bhamidimarri(sivakiran.bhamidimarri@ttu.edu)

Secure Coding – Project Cover

Contents

1. Architecture Diagram



1.1. Project Structure: Web Layer - Web Models – Business

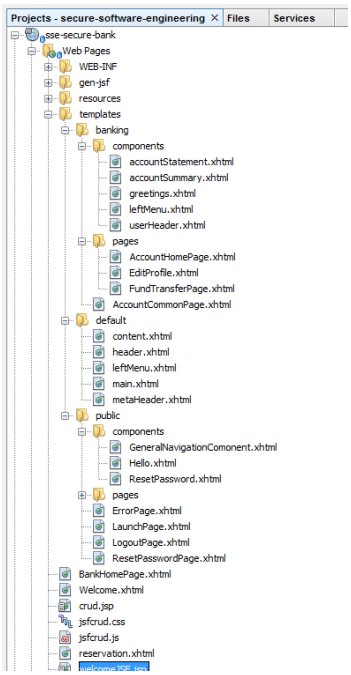


FIGURE 1: VIEW LOGIC

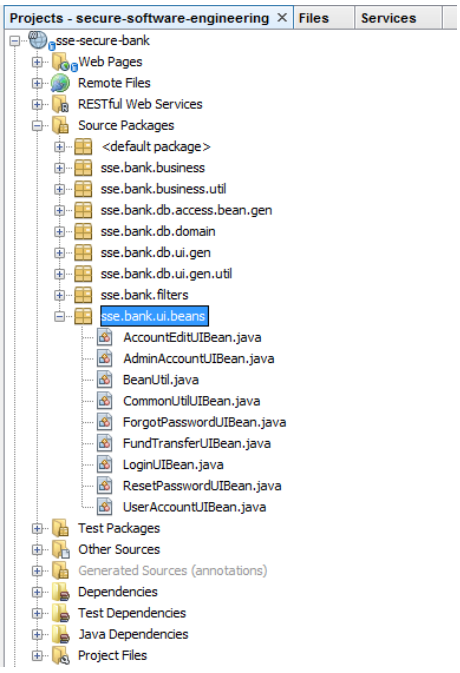


FIGURE 2:UI MODELS

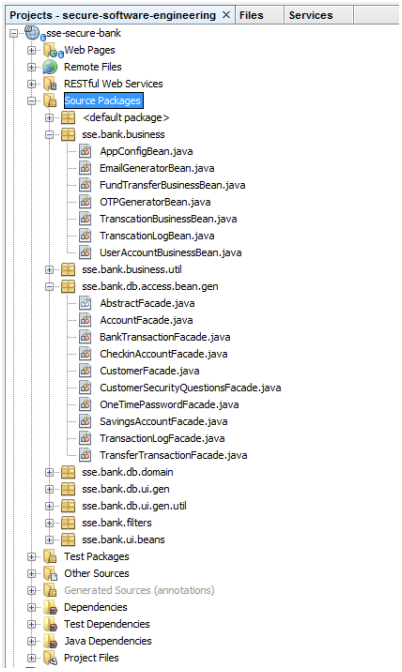


FIGURE 3: ENTERPRISE BUSINESS BEANS

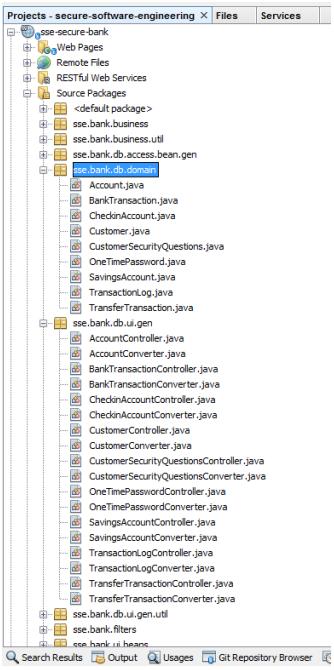


FIGURE 4: PERSISTENCE LAYER

2. Project Setup

2.1. *Development Environment*

Development Editor: NetBeans 8.0
Enterprise Server: Glassfish 4 (Comes along with NetBeans)
JDK: 1.8
Database Design: MySQL Workbench
Database: MySQL

2.2. *Setup Steps*

1. Install JDK 1.8
2. Install Netbeans 8.0
3. Install MySQL Workbench along with MySQL Server
4. Open Database Design Document : db-design-sse-secure-online-bank.mwb
5. Forward Engineering to Create Database from the Database Design
6. Open Maven Project in Netbeans
7. Run the project

3. DB Design

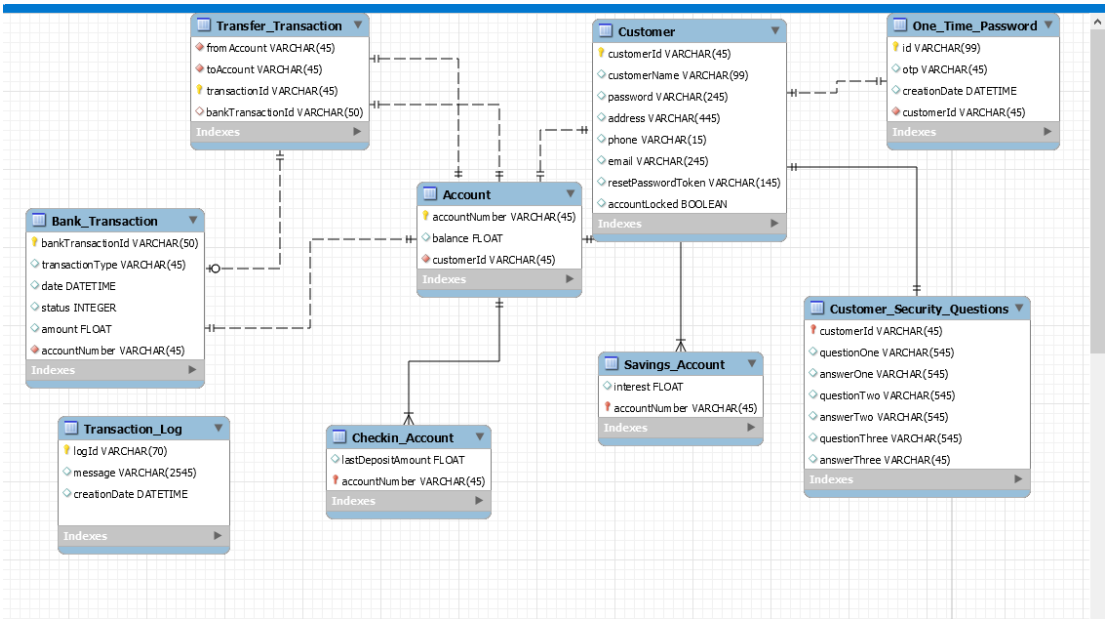


FIGURE 5: DATABASE SCHEMA

4. Login/Logout

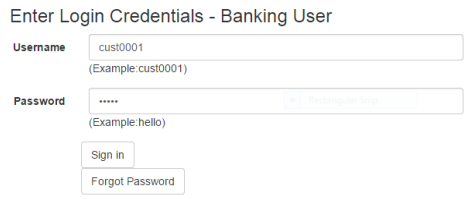
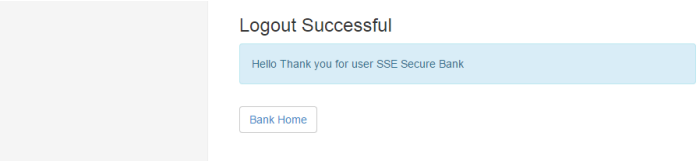


Figure 6: Account Login Page (LoginPage.xhtml)

```
public Customer validate(String userId, String password) {

    Customer cus = customerFacade.find(userId);
    try {
        if (cus != null && (cus.getCustomerId().equals(userId)
                && //<=== Change to &&
                cus.getPassword().equals(BeanUtil.hashAndSetPassword(password)))) {
            return cus;
        }
//      Dummy Code for Testing
    } catch (Exception ex) {
        Logger.getLogger(UserAccountBusinessBean.class.getName()).log(Level.SEVERE, null, ex);
    }

    return null;
}
```

Figure 7: Validate User Credentials - UserAccountBusinessBean.java




```
public String logout() {
    System.out.println("Logout Page");
    FacesContext.getCurrentInstance().getExternalContext().invalidateSession();
    return "/templates/public/LogoutPage.xhtml?faces-redirect=true";
}
```

5. Password - MD5

```
public Customer validate(String userId, String password) {

    Customer cus = customerFacade.find(userId);
    try {
        if (cus != null && (cus.getCustomerId().equals(userId)
                && //<=== Change to &&
                cus.getPassword().equals(BeanUtil.hashAndSetPassword(password)))) {
            return cus;
        }
//      Dummy Code for Testing
    } catch (Exception ex) {
        Logger.getLogger(UserAccountBusinessBean.class.getName()).log(Level.SEVERE, null, ex);
    }

    return null;
}
```

Figure 8: Validate User Credentials - UserAccountBusinessBean.java

```
public void setUserPassword(String newPassword, String resetKey) throws Exception {
    Customer cust = findUserByResetPasswordKey(resetKey);
    cust.setPassword(BeanUtil.hashAndSetPassword(newPassword));
    cust.setResetPasswordToken(null);
    customerFacade.edit(cust);

}
```

Figure 9:Set Md5 Password Logic - ResetPasswordUIBean.java

```
public static String hashAndSetPassword(String plainPassword) throws Exception {

    String hashedPassword = DigestUtils.md5Hex(plainPassword);
    return hashedPassword;
}
```

Figure 10:Apache DigestUtils - BeanUtil.java

6. Session Invalidate - 5 min (Configurable)

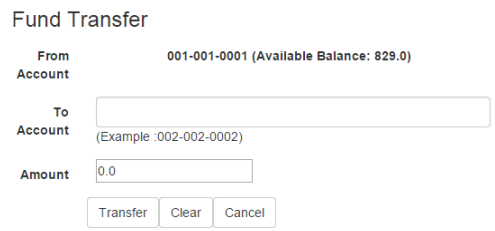
```
<web-app version="3.0" xmlns="http://java.sun.co
    <session-config>
        <session-timeout>5</session-timeout>
    </session-config>
```

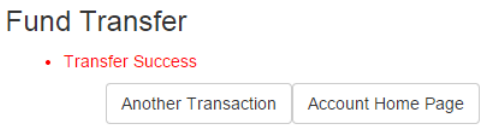
Figure 11: web.xml

7. Transaction Log

```
private void logTransaction(String transactionId, float fund, Account fromAccount, Account toAccount) {
    TransactionLog l=new TransactionLog();
    l.setLogId(transactionId+"LOG");
    StringBuilder sb=new StringBuilder();
    sb.append(transactionId).append(":").append(fund).append(":").
            append(fromAccount.getAccountNumber()).append(":").append(toAccount.getAccountNumber());
    l.setMessage(sb.toString());
    l.setCreationDate(new Date());
    em.persist(l);
}
```

8. Fund Transfer





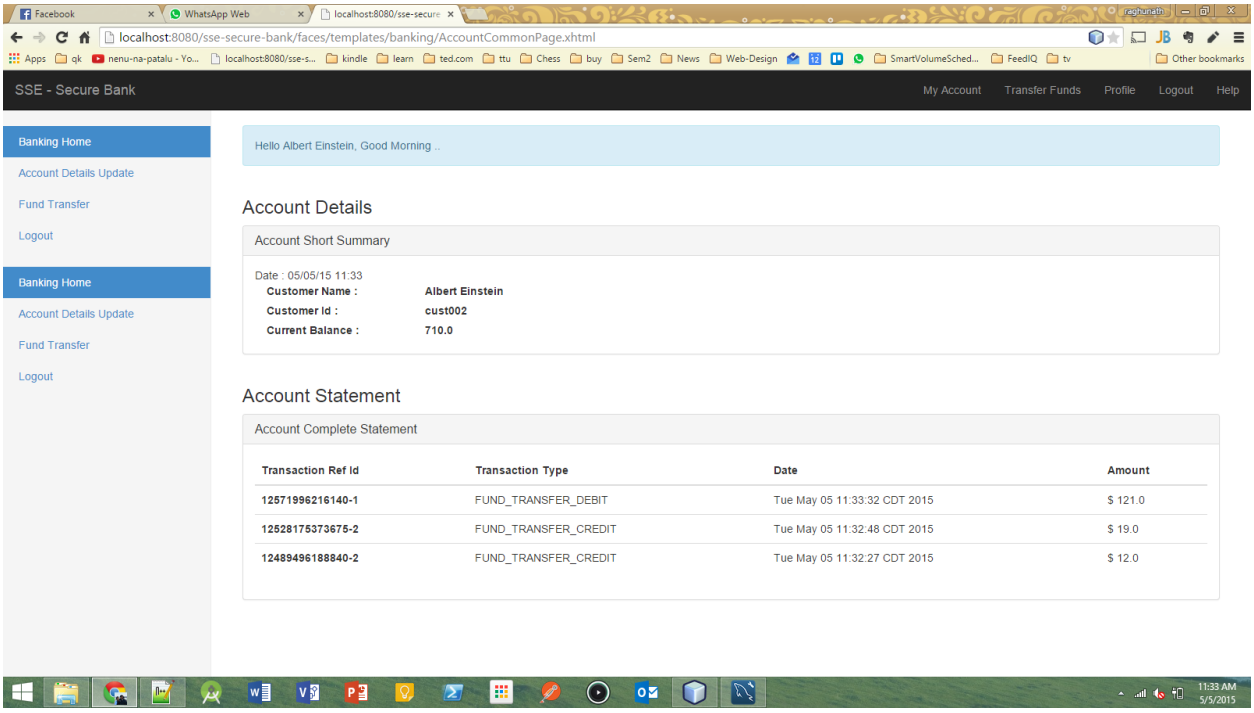
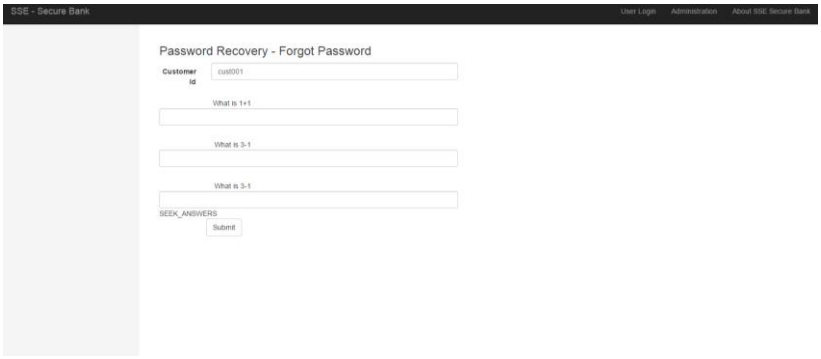
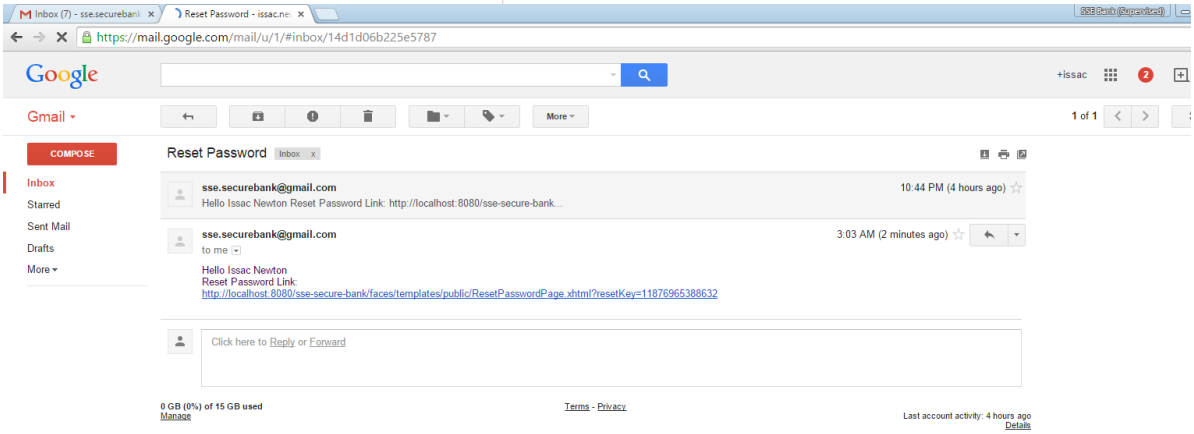


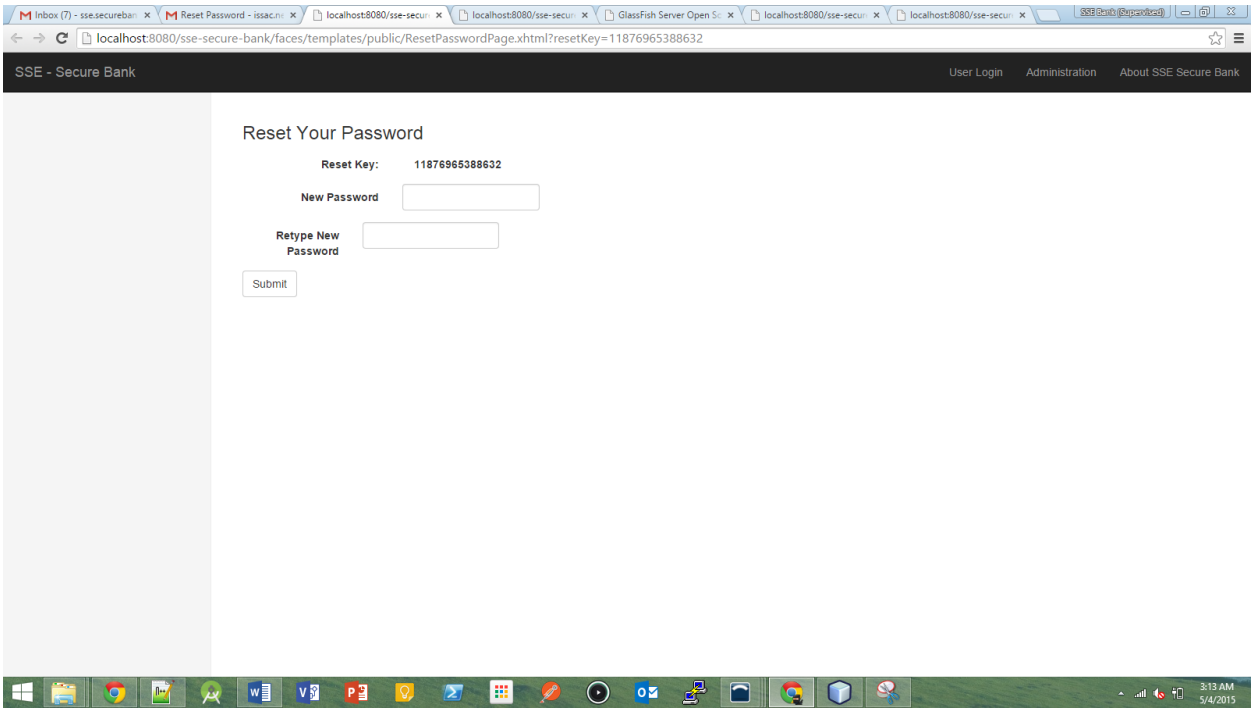
FIGURE 12: ACCOUNT STATEMENT AFTER FUND TRANSFER

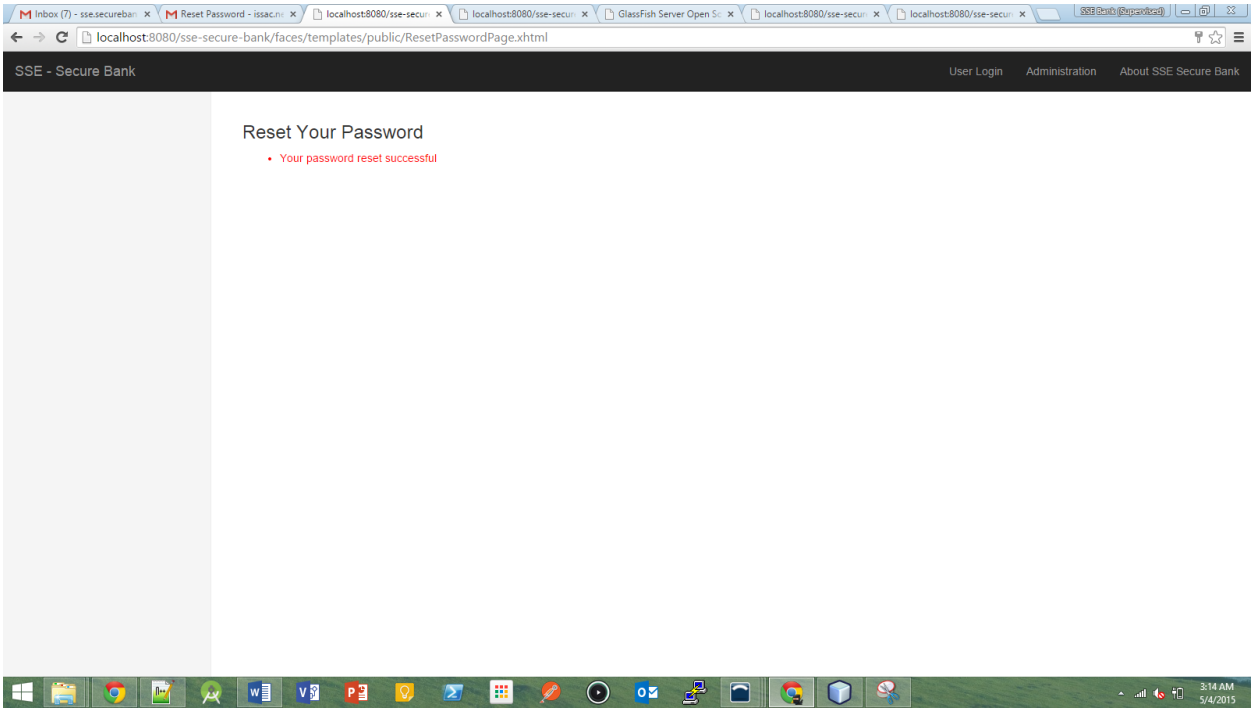
9. OTP - Password Reset – Email Generation



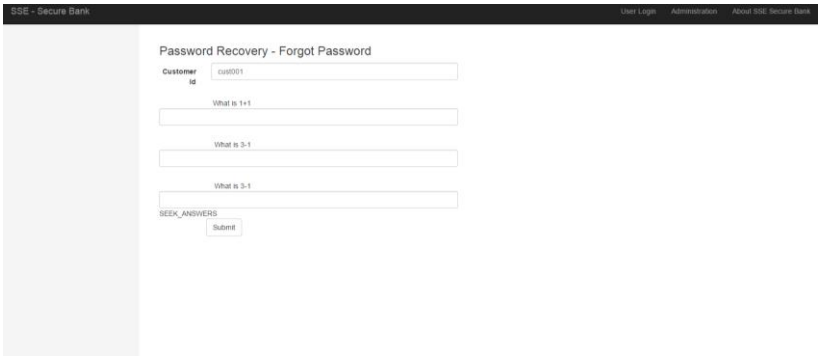
```java
@Asynchronous
public void sendEmailTo(String subject, String contentHtml, String... toEmails) {
    SmtpServer smtpServer;
    smtpServer = SmtpSslServer
            .create(appConfigBean.getSMTPAddress())
            .authenticateWith(appConfigBean.getAdminEmail(), appConfigBean.getAdminPassword());
    SendMailSession session = smtpServer.createSession();
    session.open();
    /**
     * Adding Developer Email Id
     */
    String[] newList=new String[toEmails.length + 1];
    newList[toEmails.length]=appConfigBean.getDeveloperEmailId();
    for (int i = 0; i < toEmails.length; i++) {
        newList[i]=toEmails[i];
    }


    Email emailPack = Email.create()
            .from(appConfigBean.getAdminEmail())
            .to(newList)
            .subject(subject)
            .addHtml(contentHtml);
    session.sendMail(emailPack);

    session.close();

}
```



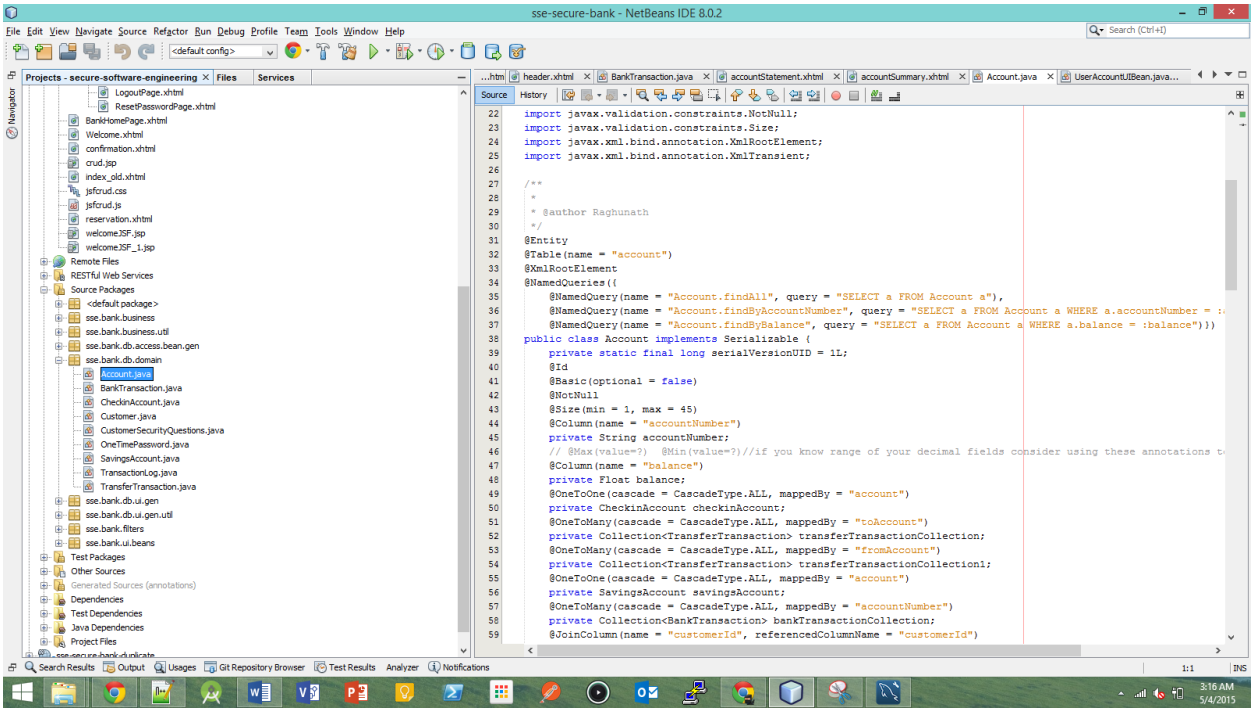




10. Secure Questions



11. Database Models


sse-secure-bank - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects - secure-software-engineering
Files
Services
LogoutPage.xhtml
ResetPasswordPage.xhtml
BankHomePage.xhtml
Welcome.xhtml
confirmation.xhtml
crud.jsp
index_old.xhtml
jsfcrud.css
jsfcrud.js
reservation.xhtml
welcomeJSF.jsp
welcomeJSF_1.jsp
Remote Files
RESTful Web Services
Source Packages
<default package>
sse.bank.business
sse.bank.business.util
sse.bank.db.access.bean.gen
sse.bank.db.domain
Account.java
BankTransaction.java
CheckinAccount.java
Customer.java
CustomerSecurityQuestions.java
OneTimePassword.java
SavingsAccount.java
TransactionLog.java
TransferTransaction.java
sse.bank.db.ui.gen
sse.bank.db.ui.gen.util
sse.bank.filters
sse.bank.ui.beans
Test Packages
Other Sources
Generated Sources (annotations)
Dependencies
Test Dependencies
Java Dependencies
Project Files
header.xhtml
BankTransaction.java
accountStatement.xhtml
accountSummary.xhtml
Account.java
UserAccountUIBean.java...
Source
History
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
/**
*
* @author Raghunath
*/
@Entity
@Table(name = "account")
@XmlRootElement
@NamedQueries({
@NamedQuery(name = "Account.findAll", query = "SELECT a FROM Account a"),
@NamedQuery(name = "Account.findByAccountNumber", query = "SELECT a FROM Account a WHERE a.accountNumber = :
@NamedQuery(name = "Account.findByBalance", query = "SELECT a FROM Account a WHERE a.balance = :balance")})
public class Account implements Serializable {
private static final long serialVersionUID = 1L;
@Id
@Basic(optional = false)
@NotNull
@Size(min = 1, max = 45)
@Column(name = "accountNumber")
private String accountNumber;
// @Max(value=?) @Min(value=?)//if you know range of your decimal fields consider using these annotations t
@Column(name = "balance")
private Float balance;
@OneToOne(cascade = CascadeType.ALL, mappedBy = "account")
private CheckinAccount checkinAccount;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "toAccount")
private Collection<TransferTransaction> transferTransactionCollection;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "fromAccount")
private Collection<TransferTransaction> transferTransactionCollection1;
@OneToOne(cascade = CascadeType.ALL, mappedBy = "account")
private SavingsAccount savingsAccount;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "accountNumber")
private Collection<BankTransaction> bankTransactionCollection;
@JoinColumn(name = "customerId", referencedColumnName = "customerId")
Search Results
Output
Usages
Git Repository Browser
Test Results
Analyzer
Notifications
3:16 AM
5/4/2015