Institut für Architektur von Anwendungssystemen

Universität Stuttgart

Universitätsstraße 38

D - 70569 Stuttgart

Studienarbeit Nr. 2266

System Activity Monitor for SWoM

Qi Qin

Studiengang:	Informatik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Dipl.-Phys. Dieter H. Roller
Begonnen am:	01. März 2010
Beendet am:	31. August 2010
CR-Klassifikation:	C.2.4, H.2.3, H.5.2, H.5.3

Abstract

Web-Service basiert auf die Service-orientierte Architektur,die als die Grundlage für moderne verteilte, heterogene Applikationen dienen. Sie sind sehr geeignet als die Funktionsschicht des Models, der „two level“ Struktur ist. Es ist charakteristisch für die Workflow-basierte Anwendungen. Wenn ein Fehler während der Ausführung eines Prozessesbeispiel auftritt, müssen die entsprechende Informationen in einer Tabelle in einem Datenbank geschrieben werden,um die korrigierende Aktionen durchzuführen. Das Administration Interface wird benutzt, um die gesammelten Informationen anzuzeigen. Die Funktionen von System Activity Monitor sind, die entsprechende Informationen, die über die Anforderungen aus anderer Komponenten von SWoM sind, zu sammeln.

Inhaltsverzeichnis

1. Einleitung

Das erste Kapital stellt die Struktur Studienarbeit dar und es wird ein Überblick über die Arbeit gegeben.

1.1 Motivation

Web Services spielen eine entscheidende Rolle in ihrer Funktion als „Motor“ für Service-orientierte Architekturen und dienstorientierte Anwendungen. Workflow-basierete Anwendungen bestehen aus zwei deutlich verschiedenen Teilen: Geschäftsprozesse (programming in the large) und Web Services implementieren die entsprechenden Algorithmen (programming in the small). In der Web Services Umgebung werden die ProzessModelle durch Web Services Business Process Excecution Language(WS-BPEL) beschieben und durch ein BPEL-konformes Workflow management(WFMS) ausgeführt.

Das Ziel von WFMS ist der Lebenszyklus der Business Prozesse zu verwalten , durch die assoziierte Prozess Modelle zu navigieren und die angemessene Web Services aufzurufen. Stuttgarter Workflowmaschine(SwoM) implentiert teilweise die entsprechenden WS-BPEL Standards.

Wenn ein Fehler während der Ausführung eines Prozessesbeispiel auftritt, müssen die entsprechende Informationen in einer Tabelle in einem Datenbank geschrieben werden,um die korrigierende Aktionen durchzuführen. Das Administration Interface wird benutzt, um die gesammelten Informationen anzuzeigen.

1.2 Ziel der Studienarbeit

Das Ziel meiner Studienarbeit ist es, ein System Activity Monitor zu beschreiben.Es kann helfen, die entsprechende Informationen über die

Anforderungen von anderen komponenten von SWoM zu sammeln.Das System Activity Monitor wurde als ein stateless EJB beschrieben,das die EJB 3.0 Anmerkung anwendet und in IBM WebSphere Application Server läuft.Die Informationen werden in IBM DB gespeichert. Die Komponenten,die die Spur von System Activity Monitor Anfragt und speichert, muss im System Management Bean der Administration Komponenten intergriert wird.

1.3 Aufbau des Dokuments

In dieser Studienarbeit gibt es folgende Kapitel:

Kapite 1 Einleitung

In diesem Kapitel gibt eine kurze Einleitung.

Kapitel 2 Technische Voraussetzungen

Dieses Kapitel beschreibt die technische Voraussetungen,was das ganze System brauchen müssen.In disem Kapitel werden IBM Rational Application Developer(RAD) und IBM DB2 Express vorgestellt und beschreibt die Installation der SwoM Runtime Umgebung.

Kapitel 3 Entwurf und Ausführung

Hier werden ganze Entwurf der StudienArbeit erklärt, inklusive erforderliche technische Kenntnisse ,notwendige Code und das Design des Interfaces.

Kapitel 4 Zusammenfassung

Dieses Kapitel enthält eine kurze Zusammenfassung.

Kapitel 5 Literaturverzeichnis

Eine Listeraturliste für die Studiernarbeit.

Kapitel 6 Abbildungsverzeichnis

Eine Abbildungsliste für die Studienarbeit.

2. Technische Voraussetzungen

In disem Kapitel werden notwendige technische Voraussetzungen für die Installation der SwoM beschrieben.

2.1 IBM Rational Application Developer for WebSphere Software

Rational Application Developer (RAD) ist eine integrierte Entwicklungsumgebung der Firma IBM für die Entwicklung und Analyse von Web-, Webservices-, Java-, J2EE- und Portalanwendungen. IBM Rational Application Developer ist für IBM WebSphere-Software optimiert, unterstützt aber Laufzeitumgebungen verschiedener Anbieter und baut auf der quelloffenen Eclipse-Plattform auf, so dass Entwickler ihre Entwicklungsumgebung anpassen und erweitern können.

Es beinhaltet spezialisierten Assistenten,Editoren und Validatoren für verschiedene Technologien:

- Java Platform,Enterprise Edition(Java EE)
- Web Services
- Service Component Architecture(SCA)
- Extensible Markup Language(XML)
- Java EE Connector(J2C)
- Web Application

RAD beinhaltet Tools um die Qualität der Codes zu verbessern.

Um SWoM 3 zum Laufen,wurde IBM WebSphere Application Server V7.0 als Laufzeitumgebung installiert.

WebSphere Application Server(WAS) stellt die Verfügbarkeit und Sicherheit Ihres Unternehmens. Bei diesem Application Server handelt es sich um eine Laufzeitumgebung für JavaEE-Anwendungen (*Java Enterprise Edition*). Üblicherweise werden diese als EAR (Enterprise Application Archive) bzw. WAR (Web Application Archive) gepackt. EAR- und WAR-Dateien sind mit

dem Werkzeug „jar“ (Java Archive) gepackte Dateien. [1]

Das System Activity Monitor wird als ein stateless EJB beschrieben,das die EJB 3.0 Anmerkung anwendet und in IBM WebSphere Application Server läuft. Rational Application Developer V7.0 wird als Programmierplattform benutzt.

2.2 IBM DB2

IBM DB2 Enterprise Server Edition ist ein relationaler Model Database Server, von IBM entwicklt. In dieser Studienarbeit benutze ich die „no-charge“ Version von DB2—DB2 Express-C.

Die Audit-Datenbank,die Runtime-Datenbank und die Buildtime Datenbank müssen in DB2 dargestellt werden.In der drei Dankbanken speichert die SWoM die Informationen über Prozessmodell. [2]

- Die Audit-Dantenbank speichert die Informationen über das Leben von Prozess Instanzen,die von Prozessmodellen als Beipiel genommen wurden.Die Informationen fassen die Aktivität und Event-Daten der Prozessinstanz um.
- Die Runtime-Datenbank speichert alle Informationen über Prozessinstanz eines bestimmten Prozessmodells in der Laufzeit.
- Die Buildtime-Datenbank speichert alle Informationen über Prozessmodelle,die in die SWoM eingeführt werden, oder die Informationen über Prozessmodelle,die in die SWoM eingeführt und eingesetzt werden.

2.3 Stuttgart Workflow Maschine

Stuttgart Workflow Maschine(SWoM) wird in der „Studienprojekte SWoM 1-3“ entwickelt,implementiert und getestet,vom Institut für „Architektur von Anwendungssystemen(IAAS)“ der Universität Stuttgart organisiert. In dem SWoM Projekt wird ein Workflow Management System(WfMS) entworfen und

realisiert,welches wesentliche Vorzüge der unterschiedlichen Architekturen aufweist. Hierbei sind die wesentlichen Konzepte aus BPEL, als auch die BPEL-Spezifikation selber, die Grundlage für die Implementierung. Die SWoM soll eine Umgebung zur Verfügung stellen, die anhand aktueller Forschungsergebnisse eine BPEL Maschine umsetzt, die auf dem neuesten Stand der Technik ist. Neben aktuellen Erkenntnissen fließen auch langjährige Erfahrungen im Bereich Datenbanken, Workflow und Web Services in die Umsetzung der Maschine ein.［3］

Die SWoM besteht aus den folgenden Modulen:

- Prozess Exceution Modul
- Administration Modul
- Invocation Handler Modul
- Service Provider, ist ein allgemeiner Term für die Kollection der eingesetzten Prozessmodellen,die als Web Services exponiert werden.

Jeder wichtige Modul enthält unterschiedliche Komponenten. Die Module werden miteinander durch „Messaging Middleware“ verbunden. Die Architektur wird in einem UML Komponentendiagram in der Abbildung 2-1 geformt.

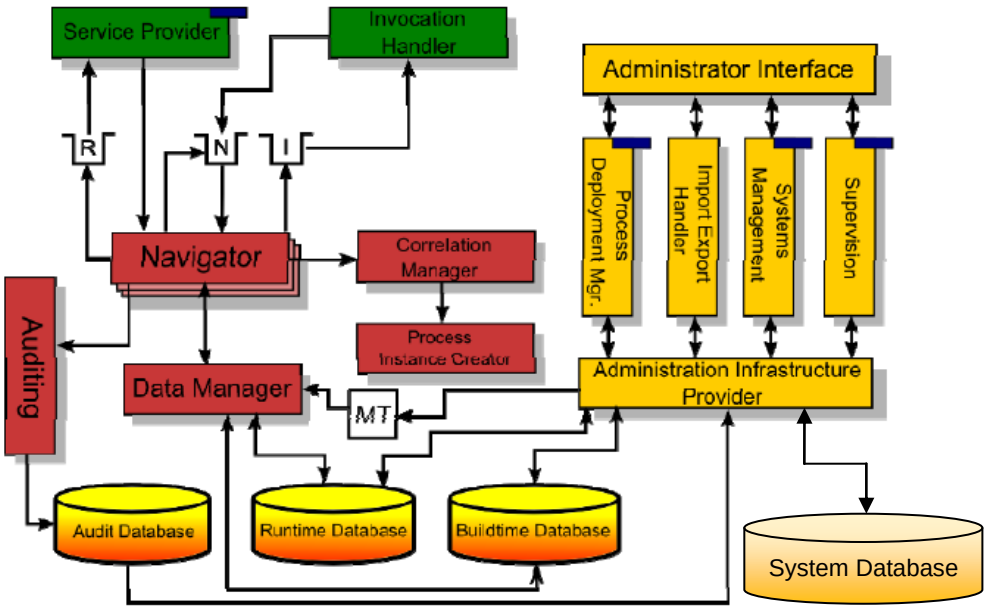

Abbildung 2- 1: SWoM Architektur Überblick［4］

Die grüne Boxe beschreiben das Anruf-Anwender und Service-Provider. Die gelbe Boxe beschreiben die Komponenten des Adminstration-Moduls und die rote Boxe sind die Komponenten des Process-Execution-Moduls. Die kleine blaue Boxe zeigt die Web Service Interfaces.Die kleine blaue Box von Import/Export Anwender ist kleiner als andere. „I“ bedeutet die Anruf-Reihen, „N“ bedeutet die Navigationsreihen und „R“ bedeutet Wiederholungsreihen. „MT“ zeichnet der Manager-Inhalt. Die Pfeile zeigen die Kommunikationsabhängigkeiten.Die Zylinder bedeuten die Datenbanken. Der an ganz rechte Seite Zylinder ist System Dantenbank, da stehen die System Activity Events. ［4］

3. Entwurf und Ausführung

Hier werden ganze Entwurf der StudienArbeit erklärt, inklusive erforderliche technische Kenntnisse ,notwendige Code und das Design des Interfaces.

3.1 Struktur der Anwendungssystem

Die ganze Anwendung besteht aus zwei Teile: „Web Service“ und „Server“.
Die Struktur wird in der Abbildung 3-1 gezeigt.
„Web Service“ läuft im J2EE-Container. In der Service-Seite verwendet das Programm eine Stateless Session Bean(SystemActivityMonitorBean), um die Daten von System Activity Event aus der Datenbank zu erhalten, am End wird dieses EJB als „Web Service“ veröffentlicht; In der Kunde-Seite, in der dynamischen Web-Projekt wird die Schnittstelle von Web service durch RAD importiert. Und wird die Datentabelle von System Activity Event durch die „JavaSever Faces“ Technologie für Endnutzer angezeigt. Hier wurde Ajax benutzt, um die lokalen Seite wiederaufzufrischen.

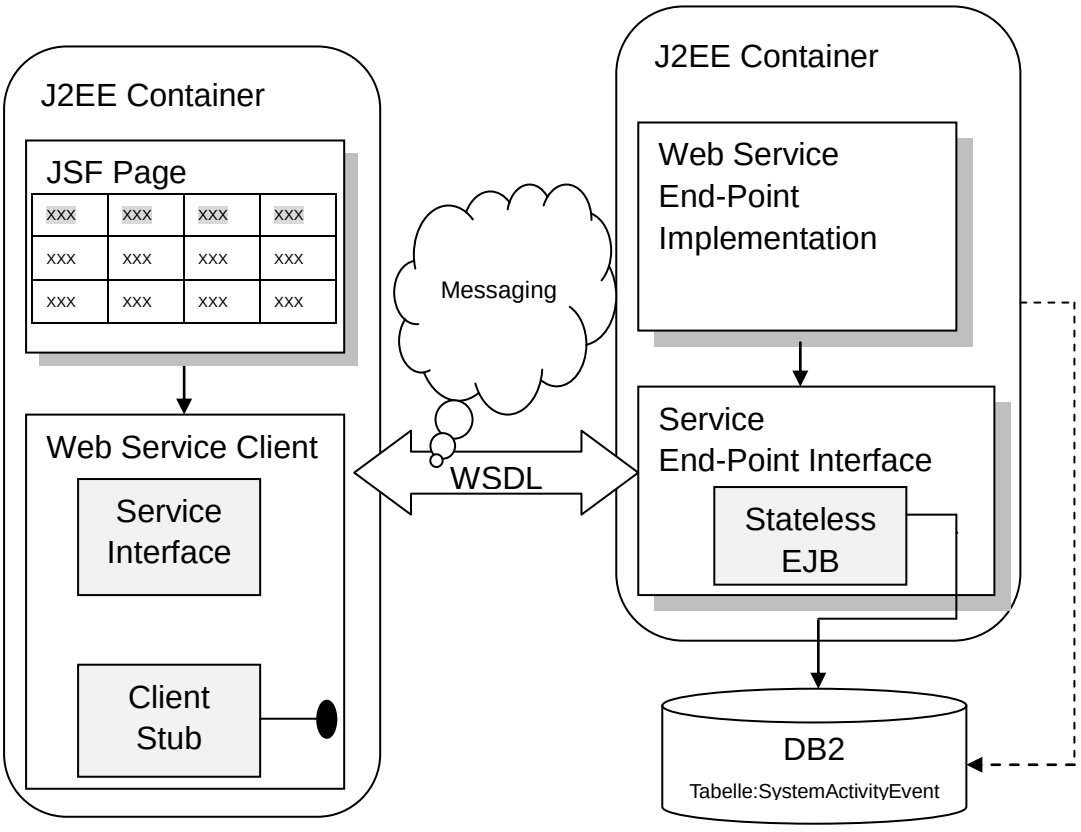

Abbildung 3-1 Struktur der Anwendungssystem

3.2 System Activity Monitor

3.2.1 System Activity Event

In Class „System Activity Event" wurden die alle wichtigen Attribute der „SystemActivityID" definiert, die „time", „area", „severity", „message" enthalten. Die Funktion von System Activity Monitor ist es,solche Informationen zu sammeln.

```
    public SystemActivityEvent(final short area, final short severity,
final String message) {
        this.time = new Timestamp(new Date().getTime());
        this.area = area;
        this.severity = severity;
        this.message = message;
        this.parameters = new LinkedList <String> () ;

    }
```

Abbildung 3-2 Code von „SystemActivityEvent"

In Class „Area" wurden verschiedende Werte von „Area" definiert. Jeder Bereich hat eigene Nummer zu identifizieren.

```
Package iaas.swom.sharedejbs.systemActivityMonitor;

public class Area {

    public static final short PROCESS_EXECUTION = 1;

    public static final short ADMINISTRATION = 2;

    public static final short DATABASE = 3;

    public static final short MESSAGING = 4 ;
}
```

Abbildung 3-3 Code von „Area"

In Class „Severity“ werden verschiedene Werte von „Severity“ definiert. „Severity“ schliesst „Information“, „Warning“, „Error“, „Severe-Error“ und „Termination“.

```
package iaas.swom.sharedejbs.systemActivityMonitor;

public class Severity {

    public static final short INFORMATION = 0;

    public static final short WARNING = 4;

    public static final short ERROR = 8;

    public static final short SEVERE_ERROR = 12 ;

    public static final short TERMINATION = 16 ;

}
```

Abbildung 3-4 Code von „Severity“

3.2.2 addEvent

In „SystemActivityMonitorBean“ wird „addEvent“ Methode definiert.

Zuerst wurde die Verbindung mit der Datenbank dargestellt. Durch „sqlStatement“ wurde die Werten von „Time“, „Severity“, „Area“, „Message_ID“, „Message“ aus Datenbank bekommen. Und dann wurde UUID erzeugt und System Activity ID gefüllt. „Severity“ und „Area“ wurden aus System Activity Event festgelegt. Danach wurden die Nachrichten für die späte Abfragen bearbeitet. Am End muss das Kontakt mit der Datenbank abgetrennt werden.

```
public void addEvent(SystemActivityEvent systemActivityEvent)
        throws InternalErrorException {

    final String methodName = "addEvent";
    logger.entering(className, methodName);

    getSystemDBConnection () ;

    try {

        String sqlStatement = "INSERT INTO SWOM.SYSTEM_ACTIVITY
(SYSTEM_ACTIVITY_ID, "
                + "TIME, SEVERITY, AREA, MESSAGE_ID, MESSAGE) "
                + "VALUES (?, ?, ?, ?, ?, ?)";

        PreparedStatement stmt =
StatementFactory.getStatement(this.logger,
                systemDBConnection, sqlStatement);

        /*
         * Generate UUID and fill system activity id
         */
        String key = this.uuidgen.getUUID();
        stmt.setString(1, key);

        /*
         * Generate time stamp
         */

        Date current = new Date () ;
        Timestamp currentTimestamp = new Timestamp
(current.getTime()) ;

        stmt.setTimestamp(2, currentTimestamp);

        /*
         * Set severity and area from system activity event
         */
        stmt.setInt(3, systemActivityEvent.getSeverity());
        stmt.setInt(4, systemActivityEvent.getArea());


        /*
         * Process message and strip off message ID for later queries
```

```
            */

            String message = systemActivityEvent.getMessage() ;
            String messageID = message.substring(0,8) ;
            stmt.setString(5, messageID);
            int messageLength = message.length();
            String messageText = message.substring(9,messageLength) ;
            stmt.setString(6,messageText) ;

            if (stmt.executeUpdate() != 1) {
                logger.logp(Level.SEVERE, className, methodName,
                        "SQL Problem when inserting systema activity
event");
            }
        } catch (SQLException SQLe1) {
            logger.logp(Level.SEVERE, className, methodName, "SQL
Exception :",
                    SQLe1);

            String errorMessage;

            try {
                systemDBConnection.close();
                errorMessage = Messages.getString("SHASA100");
            } catch (Exception SQLe2) {
                logger.logp(Level.SEVERE, className, methodName,
                        "SQLException :", SQLe2);
                errorMessage = Messages.getString("SHASA101");
            }
            InternalErrorException internalError = new
InternalErrorException(
                    errorMessage);
            throw internalError;
        }

        /*
         * Close connection
         */

        closeSystemDBConnection () ;
    }
```

Abbildung 3-5 Code von „addEvent“

3.2.3 deleteEvent und delete Set of Event

In „SystemActivityMonitorBean“ wird „deleteEvent“ und „deleteSetOfEvent“ Methode definiert.

- deleteEvent

  Durch diese Methode kann man ein Event aus der System ActivityTabelle löschen.

Zuerst wurde die Verbindung mit der Datenbank dargestellt. Durch „sqlStatement“ wurde das System Activity ID bekommen, das der Kunde löschen möchte. Wenn ein Fehler passiertet, wurde Errormessage angezeigt. Am End musste das Kontakt mit der Datenbank abgetrennt werden.

```
/**
     * Delete an event from the system activity table
     */
    public void deleteEvent(String systemActivityEventIdentifier)
            throws UserException, InternalErrorException {

        final String methodName = "deleteEvent";
        logger.entering(className, methodName);
                getSystemDBConnection () ;

        try {
            String sqlStatement = "DELETE FROM SWOM.SYSTEM_ACTIVITY WHERE

SYSTEM_ACTIVITY_ID = ?";

            PreparedStatement pstmt = StatementFactory.getStatement(
                    this.logger, systemDBConnection, sqlStatement);
            pstmt.setString(1, systemActivityEventIdentifier);
            boolean result = (pstmt.executeUpdate() == 1);
            pstmt.close();
        } catch (SQLException e) {
            String errorMessage;
            try {
                systemDBConnection.close();
                errorMessage = Messages.getString("SHASA102");
```

```
            } catch (Exception SQLe2) {
                logger.logp(Level.SEVERE, className, methodName,
                        "SQLException :", SQLe2);
                errorMessage = Messages.getString("SHASA101");
            }
    InternalErrorException internalError = new InternalErrorException(
                    errorMessage);
            throw internalError;
        }
        /*
         * Close connection
         */
        try {
            systemDBConnection.close();
        } catch (SQLException SQLe) {
            logger.logp(Level.SEVERE, className, methodName,
"SQLException :",
                    SQLe);
            String errorMessage = Messages.getString("SHASA101");
            InternalErrorException internalError = new
InternalErrorException(
                    errorMessage);
            throw internalError;
        }
        closeSystemDBConnection () ;
    }
```

Abbildung 3-6 Code von „deleteEvent“

- deleteSetOfEvent

  Durch diese Methode kann man alle Events oder alle Events in einem bestimmten „Severity“ oder alle Events in einem bestimmen „Area“ löschen.

  Wenn „Area“ und „Severity“ gleichzeitig Null waren, wurden alle Events gelöscht. Wenn „Area“ Null war und die Wert von „Severity“ vom Kunde eingegeben wurde, wurden alle Events in einem bestimmten „Severity“ gelöscht. Sonst wurden alle Events in einem bestimmten „Area“ gelöscht.

Bereich von Delete	Wert von Area	Wert von Severity
Alle Events	0	0
Alle Events in einem bestimmten „Severity“	0	vom Kunde eingegeben
Alle Events in einem bestimmten „Area“.	Vom Kunde eingegeben	

Tabelle 3-7 Einstellung des Wertes von „Area“ und „Severity“ für Delete

```
/**
     * Deletes events
     */

    public void deleteSetOfEvents(short area, short severity)
            throws InternalErrorException {

        final String methodName = "deleteSetOfEvents";
        logger.entering(className, methodName);

        getSystemDBConnection () ;

        try {

            /*
             * Delete all events
             */

            if ((area == 0) && (severity == 0)) {
                String sqlStatement = "DELETE FROM SWOM.SYSTEM_ACTIVITY";
                PreparedStatement pstmt = StatementFactory.getStatement(
                        this.logger, systemDBConnection, sqlStatement);
                boolean result = (pstmt.executeUpdate() == 1);
                pstmt.close();

            } else
            /*
             * Delete all events with a specified severity
             */

            if (area == 0) {
                String sqlStatement = "DELETE FROM SWOM.SYSTEM_ACTIVITY
```

```
WHERE SEVERITY = ?";
                PreparedStatement pstmt = StatementFactory.getStatement(
                        this.logger, systemDBConnection, sqlStatement);
                pstmt.setShort(1, severity);
                boolean result = (pstmt.executeUpdate() == 1);
                pstmt.close();

            }

            /*
             * Delete all events within a certain area
             */
            else {
                String sqlStatement = "DELETE FROM SWOM.SYSTEM_ACTIVITY
WHERE AREA = ?";
                PreparedStatement pstmt = StatementFactory.getStatement(
                        this.logger, systemDBConnection, sqlStatement);
                pstmt.setShort(1, area);
                boolean result = (pstmt.executeUpdate() == 1);
                pstmt.close();
            }
        } catch (SQLException e) {
            logger.logp(Level.SEVERE, className, methodName, "SQL
Exception ",
                    e);
            String errorMessage = Messages.getString("ADMSA103");
            InternalErrorException internalError = new
InternalErrorException(
                    errorMessage);
            throw internalError;
        }

        /*
         * Close connection
         */

        closeSystemDBConnection () ;
    }
```

Abbildung 3-8 Code von „deleteSetOfEvent“

3.2.4 querySetOfEvent

In „SystemActivityMonitorBean“ wird „querySetOfEvent“ Methode definiert. Durch diese Methode kann man die Events im verschiedenen „Area“ und/oder „Severity“ anfragen.

Wenn „Area“ und „Severity“ gleichzeitig „-1“ waren, wurden alle Events gewählt. Wenn „Area“ „-1“ war und der Wert von „Severity“ vom Kunde eingegben wurde, wurden alle Events in einem bestimmten „Severity“ gewählt. Wenn „Severity“ „-1“ war und der Wert von „Area“ vom Kunde eingegben wurde, wurden alle Events in einem bestimmten „Area“ gewählt. Der Kunde kann auch die beide Werte eingeben,um die Events in gewünschten „Area“ und „Severity“ zu wählen.

Bereich von Query	Wert von Area	Wert von Severity
Alle Events	-1	-1
Alle Events in einem bestimmten „Severity“	-1	Vom Kunde eingegeben
Alle Events in einem bestimmten „Area“	Vom Kunde eingegeben	-1
in einem unbekannten „Severity“ und unbekannten „Area“	Vom Kunde eingegeben	Vom Kunde eingegeben

Tabelle 3-9 Einstellung des Wertes von „Area“ und „Severity“ für Query

```
/**
 * Queries events
 */

public Vector<SystemActivityEvent> querySetOfEvents(short area,
        short severity) throws InternalErrorException {

    final String methodName = "querySetOfEvents";
    logger.entering(className, methodName);
```

```
        logger.logp(Level.FINER, className, methodName, "Area " + area
                + "Severity " + severity);

        /*
         * Get connection
         */

        getSystemDBConnection () ;

        try {

            /*
             * Get all events
             */

            if ((area == -1) && (severity == -1)) {
                String sqlStatement = "SELECT  SYSTEM_ACTIVITY_ID, TIME,
SEVERITY, AREA, MESSAGE_ID, MESSAGE "
                        + "FROM SWOM.SYSTEM_ACTIVITY " + "ORDER BY TIME";
                PreparedStatement pstmt = StatementFactory.getStatement(
                        this.logger, systemDBConnection, sqlStatement);
                ResultSet set = pstmt.executeQuery();
                Vector<SystemActivityEvent> result = new
Vector<SystemActivityEvent>();
                while (set.next()) {
                    SystemActivityEvent systemActivityEvent = new
SystemActivityEvent();
                    systemActivityEvent.setSystemActivityEventID(set
                            .getString(1));
                    systemActivityEvent.setTime(set.getTimestamp(2));
                    systemActivityEvent.setArea(set.getShort(3));
                    systemActivityEvent.setSeverity(set.getShort(4));
                    systemActivityEvent.setMessageID(set.getString(5));
                    systemActivityEvent.setMessage(set.getString(6));
                    result.add(systemActivityEvent);
                }
                pstmt.close();
                return result;

            } else

            /*
             * Select all events with a specified severity
             */
```

```
            if (area == -1) {
                String sqlStatement = "SELECT  SYSTEM_ACTIVITY_ID, TIME,
SEVERITY, AREA, MESSAGE_ID, MESSAGE "
                        + "FROM SWOM.SYSTEM_ACTIVITY "
                        + "WHERE SEVERITY = ? "
                        + "ORDER BY TIME";
                PreparedStatement pstmt = StatementFactory.getStatement(
                        this.logger, systemDBConnection, sqlStatement);
                pstmt.setShort(1, severity);
                ResultSet set = pstmt.executeQuery();
                Vector<SystemActivityEvent> result = new
Vector<SystemActivityEvent>();
                while (set.next()) {

                    SystemActivityEvent systemActivityEvent = new
SystemActivityEvent();
                    systemActivityEvent.setSystemActivityEventID(set
                            .getString(1));
                    systemActivityEvent.setTime(set.getTimestamp(2));
                    systemActivityEvent.setArea(set.getShort(3));
                    systemActivityEvent.setSeverity(set.getShort(4));
                    systemActivityEvent.setMessageID(set.getString(5));
                    systemActivityEvent.setMessage(set.getString(6));
                    result.add(systemActivityEvent);
                }

                pstmt.close();
                closeSystemDBConnection () ;
                return result;
            }

            /*
             * Get all events within a certain area
             */
            if (severity == -1) {
                String sqlStatement = "SELECT  SYSTEM_ACTIVITY_ID, TIME,
SEVERITY, AREA, MESSAGE_ID, MESSAGE "
                        + "FROM SWOM.SYSTEM_ACTIVITY "
                        + "WHERE AREA = ? "
                        + "ORDER BY TIME";
                PreparedStatement pstmt = StatementFactory.getStatement(
                        this.logger, systemDBConnection, sqlStatement);
                pstmt.setShort(1, area);
```

```
                ResultSet set = pstmt.executeQuery();
                Vector<SystemActivityEvent> result = new
Vector<SystemActivityEvent>();
                while (set.next()) {
                    SystemActivityEvent systemActivityEvent = new
SystemActivityEvent();
                    systemActivityEvent.setSystemActivityEventID(set
                            .getString(1));
                    systemActivityEvent.setTime(set.getTimestamp(2));
                    systemActivityEvent.setArea(set.getShort(3));
                    systemActivityEvent.setSeverity(set.getShort(4));
                    systemActivityEvent.setMessageID(set.getString(5));
                    systemActivityEvent.setMessage(set.getString(6));
                    result.add(systemActivityEvent);

        }

                pstmt.close();
                closeSystemDBConnection () ;
                return result;
            }

            /*
             * Area and severity specified
             */

            logger.logp(Level.FINER, className, methodName,
                    "Selecting area and severity");
            String sqlStatement = "SELECT  SYSTEM_ACTIVITY_ID, TIME,
SEVERITY, AREA, MESSAGE_ID, MESSAGE "
                    + "FROM SWOM.SYSTEM_ACTIVITY "
                    + "WHERE AREA = ? AND SEVERITY  = ? " + "ORDER BY TIME";
            PreparedStatement pstmt = StatementFactory.getStatement(
                    this.logger, systemDBConnection, sqlStatement);
            pstmt.setShort(1, area);
            pstmt.setShort(2, severity);
            ResultSet set = pstmt.executeQuery();
            Vector<SystemActivityEvent> result = new
Vector<SystemActivityEvent>();
            while (set.next()) {
                SystemActivityEvent systemActivityEvent = new
SystemActivityEvent();

    systemActivityEvent.setSystemActivityEventID(set.getString(1));
```

```
                systemActivityEvent.setTime(set.getTimestamp(2));
                systemActivityEvent.setArea(set.getShort(3));
                systemActivityEvent.setSeverity(set.getShort(4));
                systemActivityEvent.setMessageID(set.getString(5));
                systemActivityEvent.setMessage(set.getString(6));
                result.add(systemActivityEvent);
            }

            pstmt.close();
            closeSystemDBConnection () ;
            return result;

        } catch (SQLException e) {
            logger.logp(Level.SEVERE, className, methodName, "SQL

Exception ",
                    e);
            String errorMessage = Messages.getString("ADMSA104");
            InternalErrorException internalError = new
InternalErrorException(
                    errorMessage);
            throw internalError;
        }

    }
```

Abbildung 3-10 Code von „querySetOfEvent”

3.2.5 Verbindung mit der Dantenbank

Alle Daten über System Activity Event werden in der Datenbank gespeichert. Wenn man mit solcher Daten bearbeiten möchten, muss man zuerst eine Verbindung mit der Datenbank darstellen. Nach der Bearbeitung muss auch die Verbindung gelöscht werden.

- Verbinden mit der Dantenbank

```
    /**
         * Gets a connection to system database
         */

        private void getSystemDBConnection() throws
    InternalErrorException {

            final String methodName = "getSystemDBConnection" ;
            logger.entering(className, methodName) ;

            if (systemDBConnection == null) {
                systemDBConnection = systemSDS.getConnection();

            }
        }
```

Abbildung 3-11 Code von „getSystemDBConnection”

- Löschen die Verbindung mit der Dantenbank

```
/**
     * Closes connection to system database
     */
    private void closeSystemDBConnection() throws InternalErrorException
{
                final String methodName = "closeSystemDBConnection" ;
        logger.entering (className, methodName) ;

        if (systemDBConnection == null) {
            return ;
        }
        try {
            systemDBConnection.close() ;
            systemDBConnection = null ;
        } catch (SQLException SQLe) {
            String errorMessage = Messages.getString("SHASA101") ;
            InternalErrorException internalError = new
InternalErrorException (errorMessage) ;
            throw internalError ;
        }
            }
```

Abbildung 3-12 Code von „closeSystemDBConnection”

3.3 SystemActivityMonitoringBean

Nach dem Entwurf der Struktur, wurde eine Stateless Session Bean erzeugt, um die Datenbank zuzugreifen. Dann wurde das Session Bean als Web Service freigegeben .

Zuerst wurde ein Projekt „SystemActivityMonitoring“ dargestellt, und dann produziere ein stateless Session Bean:SystemActivityMonitoringBean.

JavaBeans sind Java Klassen,in der die bestimmten Codierungskonventionen gibt. Es bietet Getter- und Setter-Methoden für den Zugriff seiner Eigenschaften. Setter-Mothode ist auch als mutator Methode genannt. Es wird angewendet, um die Änderungen der Variablen under Kontrolle zu bringen.

Getter-Methode ist auch als accessor Methode genannt. Seine Funktion ist der Wert des privaten Membervariable zurückzugeben.

SystemActivityMonitoringBean.java

```
package iaas.swom.admininterface;

import iaas.swom.admin.buildtime.ProcessModelInfo;
import iaas.swom.admin.systemmanagement.SystemManagementLocal;
import
iaas.swom.admin.systemsactivitymonitoring.SystemActivityMonitoringLoc
al;
import iaas.swom.shared.exception.ConfigurationException;
import iaas.swom.shared.exception.DatabaseException;
import iaas.swom.shared.exception.InternalErrorException;
import iaas.swom.shared.exception.UserException;

import iaas.swom.sharedejbs.StatisticsManager.PMStatsBaseInfo;
import
iaas.swom.sharedejbs.systemActivityMonitor.SystemActivityEvent;
import
iaas.swom.sharedejbs.systemActivityMonitor.SystemActivityMonitorLocal
;

import java.util.Vector;
import java.util.logging.Logger;

import javax.ejb.EJB;
import javax.faces.context.FacesContext;
```

```
public class SystemActivityMonitoringBean {

    static String className =
SystemActivityMonitoringBean.class.getName();
    public static Logger logger = Logger.getLogger(className);

    @EJB
    private SystemActivityMonitoringLocal systemActivityMonitoring;

    private Boolean error = false;
    private Boolean success = false;
    private String errorMessage;
    private String credential = null;

    private int[] selectedEvent;
    public int[] getSelectedEvent() {
        return selectedEvent;
    }

    public void setSelectedEvent(int[] selectedEvent) {
        this.selectedEvent = selectedEvent;
    }
    private Vector <SystemActivityEvent> systemActivityEvents ;


    public Boolean getError() {
        return error;
    }

    public void setError(Boolean error) {
        this.error = error;
    }

    public Boolean getSuccess() {
        return success;
    }

    public void setSuccess(Boolean success) {
        this.success = success;
    }

    public String getErrorMessage() {
        return errorMessage;
    }
```

```
    public void setErrorMessage(String errorMessage) {
        this.errorMessage = errorMessage;
    }

    public String getCredential() {
        if (credential == null) {
            SystemsManagementBean sysManagementBean =
(SystemsManagementBean)
FacesContext.getCurrentInstance().getExternalContext().getSessionMap(
).get("systemsManagementBean");
            this.credential = sysManagementBean.getCredential();
        }
        return credential;
    }

    public void setCredential(String credential) {
        this.credential = credential;
    }

    public Vector<SystemActivityEvent> getSystemActivityEvents() {
        try {
            this.systemActivityEvents = systemActivityMonitoring
                    .getSystemActivityEvents(getCredential());
        } catch (InternalErrorException e) {
            errorMessage = e.getLocalizedMessage();
            error = true;
        } catch (UserException e) {
            errorMessage = e.getLocalizedMessage();
            error = true;
        }
        return systemActivityEvents;
    }

    public void reset() {
        this.error=false;
        this.errorMessage = "";
        this.success = false;

    }

    public void resetAll() {
        this.credential = null;
        reset();
```

Abbildung 3-13 Code von SystemActivityMonitoringBean

Im generierte „SystemActivityMonitoringLocal“ Schnittstelle wurden die notwendige Code gefügt. Durch die Anwendung dieser Schnittstelle könnte ein Interface erstellt werden,um die Daten zu lesen und zu operieren.

SystemActivityMonitoringLocal.java

```
package iaas.swom.admin.systemsactivitymonitoring;
import iaas.swom.shared.exception.InternalErrorException;
import iaas.swom.shared.exception.UserException;
import iaas.swom.sharedejbs.systemActivityMonitor.SystemActivityEvent;
import java.util.Vector;
import javax.ejb.Local;

public interface SystemActivityMonitoringLocal {

   public Vector<SystemActivityEvent> getSystemActivityEvents(String
 credentials)
    throws InternalErrorException, UserException;

    public void deleteSystemActivityEvents(String credential)
    throws InternalErrorException,UserException;

    public void querySystemActivityEvents(String credential)
    throws InternalErrorException,UserException;
}
```

Abbildung 3-14 Code von SystemActivityMonitoringLocal

In der Abbildung 3-2 wurde die Struktur der Klassen im Projekt „SystemActivityMonitoring“ gezeigt.

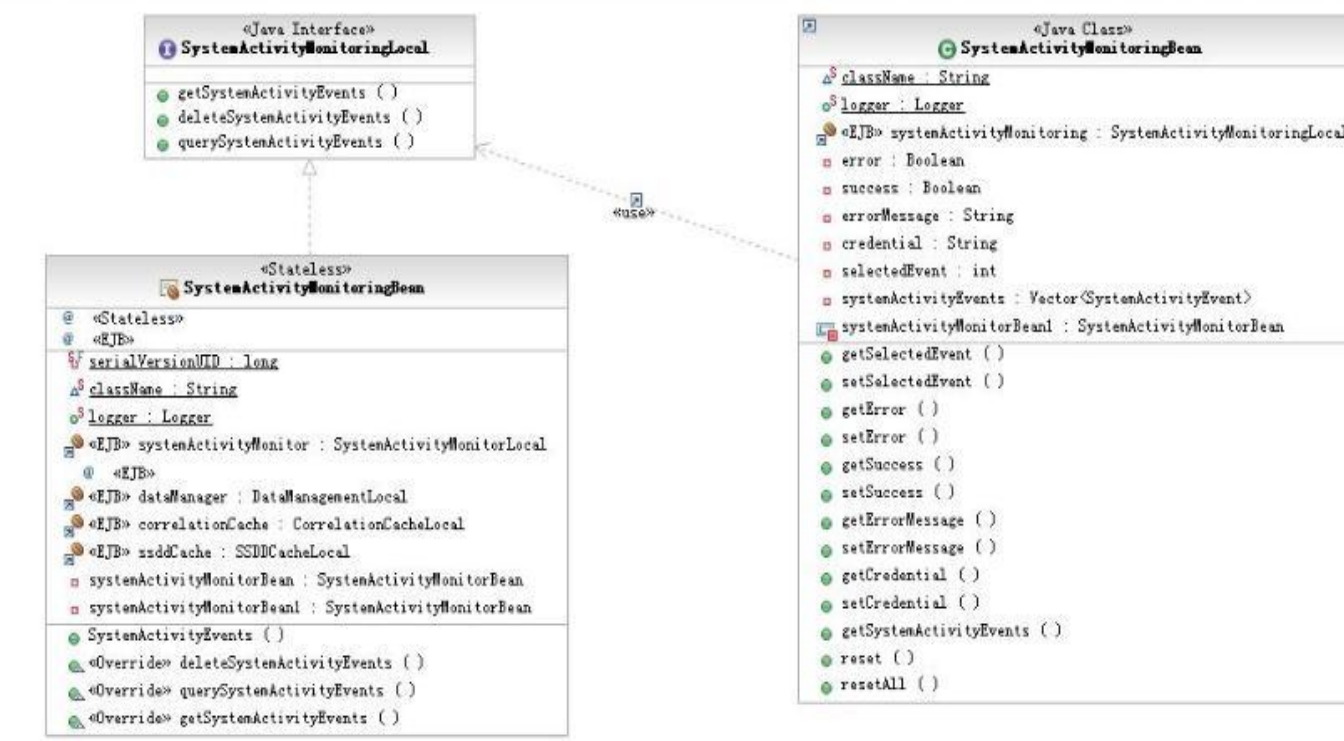

Abbildung 3-15 Struktur der Klassen

3.4 Design des Interfaces

Um das Interface darzustellen, wurde die „JavaSever Faces“(kurz JSF) Technologie benutzt.

Was ist „JSF“?

JSF ist eine Java-basierte Web-Applikation-Framework,um die Entwicklung der Intergration von web-basierten Benutzerinterface zu vereinfachen. Basierend auf Servlets und JSP-Technologie, gehört JSF zu den Webtechnologien der Java Platform, Enterprise Edition (Java EE). Bei JavaServer Faces sind die Model-Komponenten sogenannte Java-Beans (Java-Klassen), die vom Container verwaltet werden. [5]

3.4.1 Data Tabelle

Um die alle Informationen über System Activity Event anzuzeigen, benutze ich die „DataTabelle“. Wenn die DataTabelle richtig benutzt wird, kann nicht nur der Prozess vereinfacht werden,sondern kann auch die Leistung zu

verbessern.

„JavaSever Faces“ (auch „JSF“) bietet eine leistungsstarke „tag“ für die Formatierung der Tabellen während der Iteration wird das Data „***hx:dataTableEx***“ für die Iteration über die Kollektion oder Array von Werten verwendet. Die folgende Codes wurde es gezeigt, wie „***hx:dataTableEx***“ verwendet wurde, um die Daten aus den Backing Bean anzuzeigen. Die folgenden Dateien wurden in diesem Beispiel verwendet:

- system_activity_event.jsp —— um die aus managed-Bean (SystemActivityMonitoringBean) genommenen Daten anzuzeigen
- systemactivitymonitoringbean.java (in der Seite 24-26 gezeigt)
- faces-config.xml —— die Deklaration der Navigationsregel für JSF. SystemActivityMonitoringBean muss in faces-config.xml als managed-Bean registrieren.

system_activity_event.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@taglib uri="http://www.ibm.com/jsf/html_extended" prefix="hx"%>

<html>
    <head>
        <title>SWoM - Stuttgarter Workflow Maschine</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
        <link rel="stylesheet"
href="${pageContext.request.contextPath}/theme/stylesheet.css"
type="text/css">
        <!--[if IE]> <style type="text/css">table
```

```
{width:auto}</style><![endif]-->
    </head>
    <f:view>
        <body>
            <hx:scriptCollector>
                <div id="pageContainer">
                    <jsp:include page="header.jsp" />
                    <div id="contentContainer">
                        <jsp:include page="navigation.jsp" />
                        <div id="mainContent">
                       <h1><h:outputText value="Display All System
Activity Events" /></h1>
                            <p class="bold marginb"><h:outputText
value="Please select a system activity event you want to delete:" /></p>
                            <h:form>

                                <hx:dataTableEx
value="#{systemActivityMonitoringBean.systemActivityEvents}"
border="0" cellpadding="2" cellspacing="0" columnClasses="columnClass1,
columnClass2, columnClass3,columnclass4,clumnclass5"
styleClass="dataTableEx" id="tableEx1" headerClass="tableHeader"
var="systemActivityEvent">
                                    <hx:columnEx id="columnEx1">
                                    <f:facet name="header">
                                    <h:outputText styleClass="outputText"
value="Time" />
                                    </f:facet>
                                    <h:outputText id="time"
value="#{systemActivityEvents.time}" styleClass="outputText" />

                                    </hx:columnEx>
                                    <h:column id="columnEx2">
                                    <f:facet name="header">
                                           <h:outputText
styleClass="outputText" value="Message" />
                                    </f:facet>
                                    <h:outputText id="message"
value="#{systemActivityEvents.message}" styleClass="outputText" />

                                    </h:column>

                                    <h:column id="columnEx3">
                                    <f:facet name="header">
                                           <h:outputText
```

```
styleClass="outputText" value="Area" />
                                    </f:facet>
                                    <h:outputText id="Area"
value="#{systemActivityEvents.area}" styleClass="outputText" />
                                    </h:column>

                                    <h:column id="columnEx4">
                                    <f:facet name="header">
                                    <h:outputText styleClass="outputText"
value="Severity" />
                                    </f:facet>
                                    <h:outputText id="severity"
value="#{systemActivityEvents.severity}" styleClass="outputText" />

                                    </h:column>
                                    <hx:columnEx>
                                        <hx:inputRowSelect id="syid"
value="#{systemActivityMonitoringBean.selectedEvent}"
selectOne="true" />
                                    </hx:columnEx>
                                </hx:dataTableEx>
                                <h:commandButton
action="#{systemActivityMonitoringBean.deleteEvent}" title="Delete"
value="Delete" styleClass="submit" />
                                </h:form>
                            <h:outputText
rendered="#{systemActivityMonitoringBean.error}"
value="#{systemActivityMonitoringBean.errorMessage}"
styleClass="error" escape="false" />
                                <h:outputText
rendered="#{systemActivityMonitorBean.success}" value="Delete
successful" styleClass="success" />
                            </div>
                             <div class="clearfloat"></div>
                        </div>
                       <div class="clearfloat"></div>
                    </div>
                </hx:scriptCollector>
            </body>
        </f:view>
    </html>
```

Abbildung 3-16 Code von system_activity_event.jsp

faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<faces-config
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd"
    version="1.2">
    <application>

    <state-manager>com.ibm.faces.application.DevelopmentStateManager<
/state-manager>
    <locale-config>
        <supported-locale>en</supported-locale>
    </locale-config>

<message-bundle>iaas.swom.admininterface.messages</message-bundle>
        </application>
    <factory>

    <faces-context-factory>com.ibm.faces.context.AjaxFacesContextFact
ory</faces-context-factory>

    <render-kit-factory>com.ibm.faces.renderkit.AjaxRenderKitFactory<
/render-kit-factory>
    </factory>



.....
<managed-bean>

    <managed-bean-name>systemActivityMonitoringBean</managed-bean-nam
e>

    <managed-bean-class>iaas.swom.admininterface.SystemActivityMonito
ringBean</managed-bean-class>

        <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
        <lifecycle>
```

```
    <phase-listener>iaas.swom.admininterface.CheckCredentialListener<
/phase-listener>
    </lifecycle>
    <navigation-rule>

        <from-view-id>*</from-view-id>

            <navigation-case>
            <from-outcome>systemactivityevents</from-outcome>
            <to-view-id>/system_activity_events.jsp</to-view-id>
            </navigation-case>
            <navigation-case>
            <from-outcome>querysystemactivityevents</from-outcome>
            <to-view-id>/selected_event.jsp</to-view-id>
           </navigation-case>

  </navigation-rule>
</faces-config>
```

Abbildung 3-17 Code von faces-config.xml

3.4.2 Drop-down List

Um verschiedene Werte von „Area“ und „Servity“ zu wählen, benutzte ich die „Drop-down“ Liste.

„**selectOneMenu**“ tag wurde verwendet,um die Elemente anzuzeigen. Der Benutzer kann nur eine unter der Liste der verfügbaren Optionen auswählen.

„**f:selectItem**“ tag wurde verwendet, um die Auswahlliste anzuzeigen.

In der folgende Code wurde es gezeigt, wie eine Drop-down Liste dargestellt wurde.

selected_event.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@taglib uri="http://www.ibm.com/jsf/html_extended" prefix="hx"%>
```

```
<html>
    <head>
        <title>SWoM - Stuttgarter Workflow Maschine</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
        <link rel="stylesheet"
href="${pageContext.request.contextPath}/theme/stylesheet.css"
type="text/css">
        <!--[if IE]> <style type="text/css">table
{width:auto}</style><![endif]-->
    </head>
    <f:view>
        <body>
            <hx:scriptCollector>
                <div id="pageContainer">
                    <jsp:include page="header.jsp" />
                    <div id="contentContainer">
                        <jsp:include page="navigation.jsp" />
                        <div id="mainContent">
                       <h1><h:outputText value="Display Selected System
Activity Events" /></h1>
                            <p class="bold marginb"><h:outputText
value="Please select the value of the area or/and severity:" /></p>
                            <h:form>
                            <h:panelGrid columns="3">
                         <h:outputLabel  value="Area: "/>
                           <h:selectOneMenu
value="#{systemActivityEvents.selectedArea}"  required="true">
                            <f:selectItem itemLabel="---"
itemValue="-1"/>
                            <f:selectItem itemLabel="0" itemValue="0"/>
                            <f:selectItem itemLabel="1" itemValue="1"/>
                            <f:selectItem itemLabel="2" itemValue="2"/>
                            <f:selectItem itemLabel="3" itemValue="3"/>
                            <f:selectItem itemLabel="4" itemValue="4"/>
                           </h:selectOneMenu>
                    </h:panelGrid>
                    <h:panelGrid columns="3">
                    <h:outputLabel value="Severity: "/>
                              <h:selectOneMenu
value="#{systemActivityEvents.selectedSeverity}"  required="true">
                            <f:selectItem itemLabel="---"
itemValue="-1"/>
                            <f:selectItem itemLabel="0" itemValue="0"/>
```

```
                        <f:selectItem itemLabel="4" itemValue="4"/>
                        <f:selectItem itemLabel="8" itemValue="8"/>
                        <f:selectItem itemLabel="12" itemValue="12"/>
                        <f:selectItem itemLabel="16" itemValue="16"/>
                       </h:selectOneMenu>
                </h:panelGrid>
        <h:commandButton
action="#{systemActivityMonitoringBean.querySetOfEvents}"
title="Query" value="Query" styleClass="submit" />
        <hx:dataTableEx
value="#{systemActivityMonitoringBean.systemActivityEvents}"
                        border="0" cellpadding="2" cellspacing="0"
columnClasses="columnClass1, columnClass2,
columnClass3,columnclass4,clumnclass5"
                        styleClass="dataTableEx" id="tableEx1"
headerClass="tableHeader" var="querysystemActivityEvent">
                            <hx:columnEx id="columnEx1">
                            <f:facet name="header">
                                <h:outputText
styleClass="outputText" value="Time" />
                            </f:facet>
                            <h:outputText id="time"
value="#{systemActivityEvents.time}" styleClass="outputText" />

                            </hx:columnEx>
                            <h:column id="columnEx2">
                            <f:facet name="header">
                                <h:outputText
styleClass="outputText" value="Message" />
                            </f:facet>
                            <h:outputText id="message"
value="#{systemActivityEvents.message}" styleClass="outputText" />

                            </h:column>
                            <h:column id="columnEx3">
                            <f:facet name="header">
                                <h:outputText
styleClass="outputText" value="Area" />
                            </f:facet>
                            <h:outputText id="Area"
value="#{systemActivityEvents.area}" styleClass="outputText" />

                            </h:column>
                            <h:column id="columnEx4">
```

```
                                    <f:facet name="header">
                                        <h:outputText
styleClass="outputText" value="Severity" />
                                    </f:facet>
                                    <h:outputText id="severity"
value="#{systemActivityEvents.severity}" styleClass="outputText" />

                                </h:column>
                                <hx:columnEx>
                                    <hx:inputRowSelect id="syid"
value="#{systemActivityMonitoringBean.selectedEvent}"
selectOne="true" />
                                </hx:columnEx>


                            </hx:dataTableEx>

                            <h:commandButton
action="#{systemActivityMonitoringBean.deleteEvent}" title="Delete"
value="Delete" styleClass="submit" />
                        </h:form>
                        <h:outputText
rendered="#{systemActivityMonitoringBean.error}"
value="#{systemActivityMonitoringBean.errorMessage}"
styleClass="error" escape="false" />
                        <h:outputText
rendered="#{systemActivityMonitorBean.success}" value="Delete
successful" styleClass="success" />
                    </div>
                     <div class="clearfloat"></div>
                </div>
                <div class="clearfloat"></div>
            </div>
        </hx:scriptCollector>
    </body>
</f:view>
</html>
```

Abbildung 3-18 Code von selected_event.jsp

In der folgenden Abbildung wurde der Effekt der Interfaces am IE gezeigt.

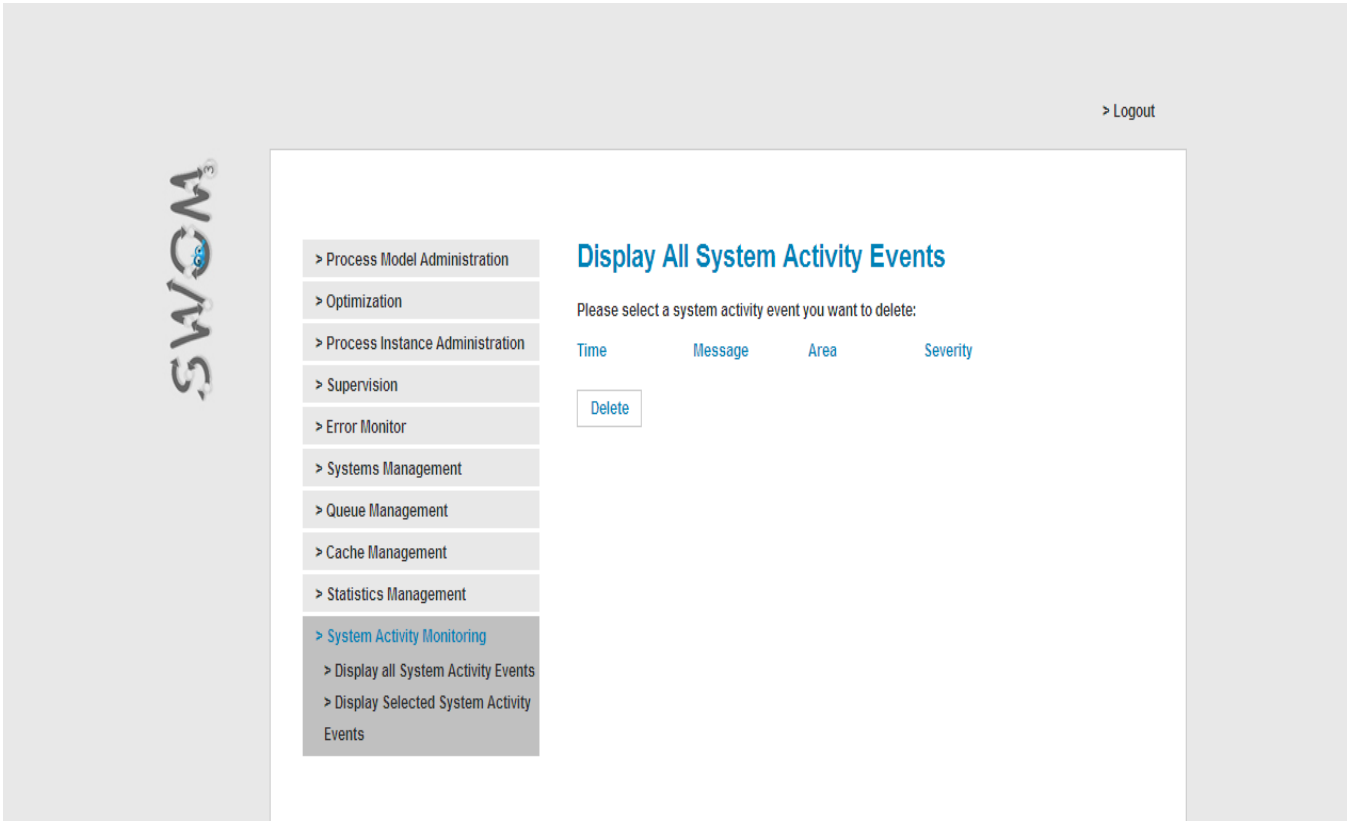

Abbildung 3-19 Interface „Display All System Activity Events“

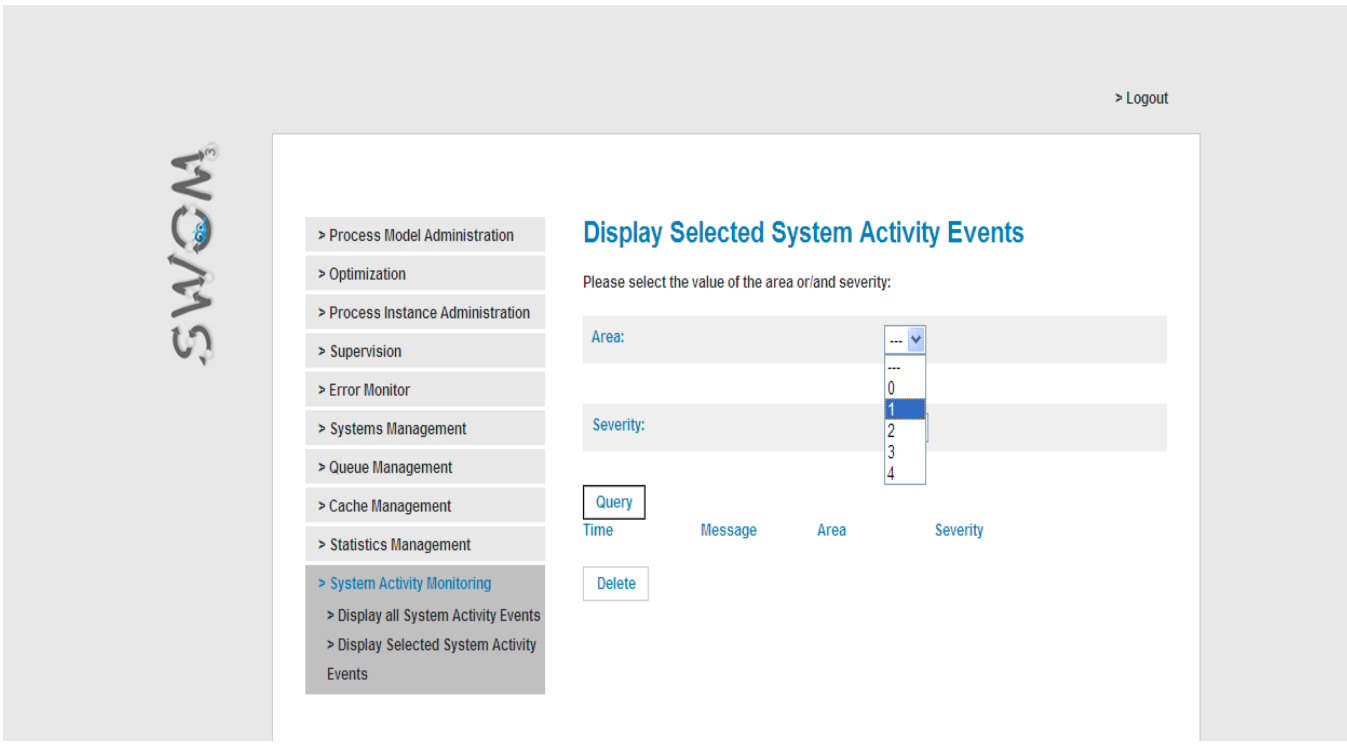

Abbildung 3-20 Interface „Display Selected System Activity Events“

4. Zusammenfassung

Das Ziel dieser Studierenarbeit war eine Anwendung zu entwickeln,um die entsprechende Informationen über die Anforderungen von anderen komponenten von SWoM zu sammeln. In dieser Studierenarbeit wurde solche Technologie erklärt:

- Wie ein EJB im RAD V7 erzeugt wurde;
- Wie das Web Service veröffentlicht wurde;
- Wie eine Ajax-basierte JSF-Anwendung dargestellt wurde.

5. Literaturverzeichnis

[1]: Wikipedia WebSphere:

http://de.wikipedia.org/wiki/WebSphere

[2]: Database Design – SWoM3 Team

[3]: SWoM Projekt

http://www.iaas.uni-stuttgart.de/forschung/projects/swom/

[4]: SWoM 3 Team

[5]: JavaServer Faces

http://de.wikipedia.org/wiki/JavaServer_Faces

6. Abbildungsverzeichnis

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift: Qi Qin

Stuttgart, 25.08.2010