Development of a city mobile application

Additional Appendix

Source Code

CityMob

June 27, 2007

Contents

Part I

Client source code - MyCityGuide

CHAPTER 1

MobCanvas.java

```
/*
 * MobCanvas.java
 *
 * Created on 21. mars 2007, 15:52
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package de.unternehmertum.mobile.mycityguide;

import de.unternehmertum.mobile.playtools.event.Event;
import de.unternehmertum.mobile.playtools.gui.GuiComponentImpl;
import de.unternehmertum.playbox.remoting.CityEvent;
import java.io.IOException;
import java.lang.Exception;
import java.util.Calendar;
import java.util.Date;
import javax.microedition.lcdui.*;
import de.unternehmertum.playbox.remoting.Place;

/**
 * Handles all the graphics of the application, and logic that comes with it
 * @author Håkon
 */
```

```
public class MobCanvas extends Canvas implements MobCanvasInterface{

    /** Creates a new instance of MobCanvas */
    public MobCanvas(){
    }

    /**
     * Sets the values that normaly would be set in the constuctor. Also loads
     * the images and recognizes what type of mobile is beeing used and set
     * corresponding values.
     */
    public void initialize() {
        try{
            bildeRest = Image.createImage("/images/essen.png");
            bildeGo = Image.createImage("/images/planen.png");
            bildeTrinken = Image.createImage("/images/trinken.png");
            bildeFilm = Image.createImage("/images/sehen.png");
            bildeTanzen = Image.createImage("/images/tanzen.png");
            bildeHoren = Image.createImage("/images/horen.png");
            bildeMap = Image.createImage("/images/map.png");
            bildeSuche = Image.createImage("/images/suche.png");
            bildeFavoritt = Image.createImage("/images/favoritt.png");
            bildeRest2 = Image.createImage("/images/essen_rollover.png");
            bildeGo2 = Image.createImage("/images/planen_rollover.png");
            bildeTrinken2 = Image.createImage("/images/trinken_rollover.png");
            bildeFilm2 = Image.createImage("/images/sehen_rollover.png");
            bildeTanzen2 = Image.createImage("/images/tanzen_rollover.png");
            bildeHoren2 = Image.createImage("/images/horen_rollover.png");
            bildeMap2 = Image.createImage("/images/map_rollover.png");
            bildeSuche2 = Image.createImage("/images/suche_rollover.png");
            bildeFavoritt2 = Image.createImage("/images/favoritt_rollover.png");
            button = Image.createImage("/images/knapp.png");
            button2 = Image.createImage("/images/knapp_rollover.png");
            was = Image.createImage("/images/was.png");
            warum = Image.createImage("/images/warum.png");
            header = Image.createImage("/images/header.png");
            tab1 = Image.createImage("/images/1_tab.png");
            tab2 = Image.createImage("/images/2_tab.png");
            tab3 = Image.createImage("/images/3_tab.png");
            tab4 = Image.createImage("/images/4_tab.png");
            tab5 = Image.createImage("/images/5_tab.png");
            star = Image.createImage("/images/star.png");
            star2 = Image.createImage("/images/star_2.png");
            vorlage = Image.createImage("/images/vorlage.png");
```

```
} catch (IOException e) {
    System.out.println("lastet ikke bilde");
}

//Width and hight of mobile screen
int theericcsongap=20;
int thenokiagap= 64;
softkeyspace=0;
scrW = getWidth();
int a= getHeight();
hoydeinfoBar=header.getHeight();

if(getHeight()>280){
    scrH=getHeight();
    mobtype=NETBEANS;
}
else if(getHeight()<218){
    softkeyspace=15;
    scrH=getHeight()+thenokiagap-softkeyspace;
    mobtype=NOKIA;
}
else {
    scrH= getHeight()+theericcsongap;
    mobtype=ERICSSON;
}

//Size of pictures on main menu
bildefilmW = bildeFilm.getWidth();
bildefilmH = bildeFilm.getHeight();
bildefilm2W = bildeFilm2.getWidth();
bildefilm2H = bildeFilm2.getHeight();

//Measures for placing the items on the main menu
widthSpacebetweenIcons = ((scrW - (bildefilmW*3))/4);
hightSpacebetweenIcons = (((scrH-hoydeinfoBar) -
        (widthSpacebetweenIcons*4 + bildefilmW*3))/2);
kolonne1 = widthSpacebetweenIcons;
kolonne2 = 2*widthSpacebetweenIcons + bildefilmW;
kolonne3 = 3*widthSpacebetweenIcons + 2*bildefilmW;
rekke1 = hoydeinfoBar + hightSpacebetweenIcons + widthSpacebetweenIcons;
rekke2 = hoydeinfoBar + hightSpacebetweenIcons +
        2*widthSpacebetweenIcons + bildefilmH;
rekke3 = hoydeinfoBar + hightSpacebetweenIcons +
        3*widthSpacebetweenIcons + 2*bildefilmH;
```

```
    //Width of the frame around buttons in main menu
    corrH = (bildefilm2H-bildefilmH)/2;
    corrW = (bildefilm2W-bildefilmW)/2;

    //Width of the frame around buttons in searh menus
    buttonramme= (button2.getHeight()-button.getHeight())/2;

    //Placement of text in buttons according to upper left corner
    textdiffx=12;
    textdiffy= 8;

    //Sets value for level 3, 4 and 5
    maxPopupElementerLevel3();
    maxPopupElementerLevel4();
    maxPopupElementerLevel5();
    resetlevel3values();

    //Sets the temporary values shown on the search pages
    midlEssenCategory=ApplicationData.defaultEssenCategory;
    midlCityDistrictEssen= ApplicationData.defaultCityDistrictEssen;
    midlEventType= ApplicationData.defaultEventType;
    midlDate=ApplicationData.defaultDate;
    midlTrinkenCategory= ApplicationData.defaultTrinkenCategory;
    midlDistrictTrinken= ApplicationData.defaultcityDistrictTrinken;
    midlCinema=ApplicationData.defaultCinema;
    midlCityDistrictTanzen=ApplicationData.defaultCityDistrictTanzen;
}



/**
 * Handels ALL key events. Key codes -6 and -7 seven are left and right
 * softkey.
 * @param keyCode the value of the key pressed
 */
protected void keyPressed( int keyCode ){
    if(currentCanvasPopUpDisplay==POP_UP_NO_RESULTS_FOUND&&
            currentCanvasDisplay!=MULTIPLE_SEARCH_RESULT){
        currentCanvasPopUpDisplay=NO_POP_UP;
    }else if (currentCanvasPopUpDisplay==POP_UP_SEARCHING){
        if (keyCode==-7&&currentCanvasDisplay!=MULTIPLE_SEARCH_RESULT){
            currentCanvasPopUpDisplay=POP_UP_NO_RESULTS_FOUND;
            applicationData.cancelSearch();
        }
    }else{
```

```
if(keyCode==-6){
    if(level==1){
        choosefromlevel1();
    }else if(level==2){
        choosefromundercatagoryorsearch();
    }else if(level==3){
        popUpFire();
    }else if(level==4){
        searchResultFire();
    }else if(level==5){
        elementFire();
    }
}else if (keyCode==-7){
    if(level==1){
        applicationData.startMvvService();
    }else if(level==2){
        level=1;
        currentCanvasDisplay=MAIN_MENU;
        z=1;
        resetdefaults();
    }else if(level==3){
        level=2;
        currentCanvasPopUpDisplay=NO_POP_UP;
        resetlevel3values();
        z=1;
    }else if(level==4){
        backLevel4();
    }else if(level==5){
        level=4;
        currentCanvasPopUpDisplay=NO_POP_UP;
        descriptionString=null;
        retrivelevel4values();
    }
}else{
    switch (getGameAction(keyCode)){
        //Level 1 is main menu, level 2 is searchmenus
        case UP:
            if(level==1){
                if(y==1) y=antHoyde;
                else y--;
            }else if(level==2){
                if(z==1) z=antz;
                else z--;
            }else if(level==3){
                popUpUp();
```

```
    }else if(level==4){
        popUpUp();
    }else if(level==5){
        elementUp();
    }
    break;
case DOWN:
    if(level==1){
        if(y==antHoyde) y=1;
        else y++;
    }else if(level==2){
        if(z==antz) z=1;
        else z++;
    }else if(level==3){
        popUpDown();
    }else if(level==4){
        popUpDown();
    }else if(level==5){
        elementDown();
    }
    break;
case LEFT:
    if(level==1){
        if(x==1){
            x=antBredde;
            if(y==1) y=antHoyde;
            else y--;
        } else x--;
    }else if(level==4){
        if(u==1) u=5;
        else u--;
        resetlevel4values();
        currentCanvasPopUpDisplay=NO_POP_UP;
        doSearch=true;
    }else if(level==5){
        if(w==1) w=2;
        else w=1;
    }else{
    }
    break;
case RIGHT:
    if(level==1){
        if(x==antBredde){
            x=1;
            if(y==antHoyde) y=1;
```

```
                        else y++;
                    } else x++;
                }else if (level==4){
                    if(u==5) u=1;
                    else u++;
                    currentCanvasPopUpDisplay=NO_POP_UP;
                    doSearch=true;
                    resetlevel4values();
                }else if(level==5){
                    if(w==1) w=2;
                    else w=1;
                }else{
                }
                break;
            case FIRE:
                if(level==1){
                    choosefromlevel1();
                }else if(level==2){
                    choosefromundercatagoryorsearch();
                }else if(level==3){
                    popUpFire();
                }else if(level==4){
                    searchResultFire();
                }else if(level==5){
                    elementFire();
                }
                break;
            default:
        }
    }
    repaint();

}


//XXXXXXXXXXXXXXXXXXXX   GENERAL PAINT STUFF   XXXXXXXXXXXXXXXXXXXX

/**
 * Paints the screen according to what the user is doing. Has three levels:
 * currentCanvasDisplay, currentCanvasPopUpDisplay and
 * currentCanvasPopUpDisplay2. These are levels that can be drawn on top
 * of each other.
 * @param g Graphics object
 */
```

```
protected void paint(Graphics g) {
    setFullScreenMode(true);
    g.setColor(0x000000);
    g.fillRect( 0, header.getHeight(), scrW, scrH+softkeyspace);
    displaySoftKeys(g);
    if(currentCanvasDisplay==MAIN_MENU)displayMainMenu(g);
    if(currentCanvasDisplay==MAIN_MENU||currentCanvasDisplay==SEARCH_PAGE||
            currentCanvasDisplay==SEARCH_RESULT||
            currentCanvasDisplay==MULTIPLE_SEARCH_RESULT)displayInfoBar(g);
    if(currentCanvasDisplay==SEARCH_PAGE)displaySearchPage(g);
    if(currentCanvasDisplay==SEARCH_RESULT)displaySearchResults(g);
    if(currentCanvasDisplay==MULTIPLE_SEARCH_RESULT)
        displaymultiplesearchresult(g);
    if(currentCanvasPopUpDisplay==POP_UP)displayPopUpMenu(g);
    if(currentCanvasPopUpDisplay==POP_UP_ELEMENT_INFO)displayElement(g);
    if(currentCanvasPopUpDisplay==POP_UP_SEARCHING)displaySearching(g);
    if(currentCanvasPopUpDisplay==POP_UP_NO_RESULTS_FOUND)
        displayNoResults(g);
    if(currentCanvasPopUpDisplay2==POP_UP2_LOADING)displayLoading(g);
}

/**
 * Draws the search pop-up when a search is made
 * @param g Graphics object
 */
private void displaySearching(Graphics g) {
    int popW= (scrW-vorlage.getWidth())/2;
    int popH=(scrH-vorlage.getHeight())/2;
    g.drawImage(vorlage, popW, popH, 0);
    g.setColor(0xFFFFFF);
    g.setFont(headline);
    g.drawString("Suchen...", popW+35, popH+(vorlage.getHeight()/2)-10, 0);
}

/**
 * Draws the no result pop-up when no results are found
 * @param g Graphics object
 */
private void displayNoResults(Graphics g) {
    int popW= (scrW-vorlage.getWidth())/2;
    int popH=(scrH-vorlage.getHeight())/2;
    g.drawImage(vorlage, popW, popH, 0);
    g.setColor(0xFFFFFF);
    g.setFont(headline);
    g.drawString("Keine Ergebnisse", popW+7,
```

```
            popH+(vorlage.getHeight()/2)-10, 0);
    }

    /**
     * Draws the loading pop-up when a map is loaded
     * @param g Graphics object
     */
    private void displayLoading(Graphics g) {
        int popW= (scrW-vorlage.getWidth())/2;
        int popH=(scrH-vorlage.getHeight())/2;
        g.drawImage(vorlage, popW, popH, 0);
        g.setColor(0xFFFFFF);
        g.setFont(headline);
        g.drawString("Laden...", popW+32, popH+(vorlage.getHeight()/2)-10, 0);
    }

    /**
     * Controls what is to be shown on the screen over the softkeys. Uses the
     * level parameter to place the correct value.
     * @param g Graphics object
     */
    public void displaySoftKeys(Graphics g){
        g.setFont(boldsmall);
        g.setColor(0xFFFFFF);
        int skw1 = 4;
        int skh=scrH+3;

        if(level==1){
            g.drawString("Wählen", skw1, skh, 0);
            g.drawString("MVV", 142, skh, 0);
        } else if(level==2){
            if(currentCanvasPopUpDisplay==POP_UP_SEARCHING){
                g.drawString("Abbrechen", 112, skh, 0);
            }else if(currentCanvasPopUpDisplay==POP_UP_NO_RESULTS_FOUND){
                g.drawString("Schließen", 120, skh, 0);
            }else{
                g.drawString("Wählen", skw1, skh, 0);
                g.drawString("Zurück", 130, skh, 0);
            }
        } else if(level==3){
            g.drawString("Wählen", skw1, skh, 0);
            g.drawString("Schließen", 120, skh, 0);
        } else if(level==4){
            g.drawString("Wählen", skw1, skh, 0);
            g.drawString("Zurück", 130, skh, 0);
```

```
    } else if(level==5){
        if(w==1)g.drawString("Wählen", skw1, skh, 0);
        g.drawString("Schließen", 120, skh, 0);
    }

}

/**
 * Draws the infobar on top of the screen. This inforbar displays
 * information on what part of the application the user is currently at.
 * @param g Graphics object
 */
private void displayInfoBar(Graphics g) {
    g.drawImage(header,0,0,0);
    g.setColor(0xFFFFFF);
    g.setFont(boldmedium);
    g.drawString(catagoryHeadline, infoBarInfoW, infoBarInfoH, 0);

}

/**
 * Starts the paint methode
 * @see paint
 */
public void repaintCanvas() {
    repaint();
}

/**
 * Important methode. If information is to be displayed in a scroll window,
 * this methode handles it. The metode handles scroll windows that look
 * like pop-ups, adjusting width and hight, but also handles scroll windows
 * that take up the whole screen. It handles pages where each line is
 * highlight, for instance a collection of restaurants which can be chosen.
 * But it also handles pages that only are for reading.
 * @param g Graphics object
 * @param elementsOnScreen Array of string elements, where each element
 *      represents one line
 * @param nrOfLines Total elements in array
 * @param maxNrOfLines Maximum number of lines allowed to show in the pop-up
 * @param maxLengthString Maximum length of string allowed
 * @param pixelsToNextline Height between each line
 * @param popUpWidth Width of pop-up if necassary
 * @param popUpHightWithScroll Height of pop-up if pop-up is scrollable
 * @param popUpHightWithoutScroll Hight of pop-up without scrolling (no
```

```
     *      need to scroll, not enough elements)
     * @param startWriteWidth Starting point of letters
     * @param pointerWriteHight Height of the rectangled box highlighting an
     *      entity
     * @param usefont Which font to be used
     * @param highlight highlight or not
     * @param adjustablePopUp adjustable pop-up or not
     * @param differnethighlight highlight type two or not
     * @param color color of writing
     * @param colormarkingbar color of marking bar
     */
    public void drawScrollingPage(Graphics g, String[] elementsOnScreen,
            int nrOfLines, int maxNrOfLines, int maxLengthString,
            int pixelsToNextline, int popUpWidth, int popUpHightWithScroll,
            int popUpHightWithoutScroll, int startWriteWidth,
            int pointerWriteHight, Font usefont, boolean highlight,
            boolean adjustablePopUp, boolean differnethighlight, int color,
            int colormarkingbar){
        int l;
        g.setFont(usefont);
        String writeOnPopUp;
        if(nrOfLines<=maxNrOfLines){
            if(adjustablePopUp){
                popUpHightWithoutScroll=nrOfLines*pixelsToNextline;
                pointerWriteHight = (scrH-popUpHightWithoutScroll)/2;
                g.setColor(0xFFFFFF);
                g.fillRect(startWriteWidth, pointerWriteHight, popUpWidth,
                        popUpHightWithoutScroll);
                g.setColor(0xB9B9B9);
                g.drawRect(startWriteWidth, pointerWriteHight, popUpWidth,
                        popUpHightWithoutScroll);
                g.setColor(0x000000);
                g.drawRect(startWriteWidth-1, pointerWriteHight-1,
                        popUpWidth+2, popUpHightWithoutScroll+2);
            } else pointerWriteHight+=2;

            for(l=0; l<nrOfLines; l++){
                writeOnPopUp=elementsOnScreen[l];
                g.setFont(medium);
                g.setColor(color);
                if(highlight){
                    if (l==t){
                        g.setColor(colormarkingbar);
                        if(adjustablePopUp) g.fillRect(startWriteWidth+1,
                                pointerWriteHight-1, popUpWidth-2,
```

```
                        pixelsToNextline);
                else g.fillRect(startWriteWidth, pointerWriteHight,
                        popUpWidth-2, pixelsToNextline-1);
                g.setColor(color);
                if(differnethighlight){
                    g.setColor(0xFF0000);
                    g.setFont(boldmedium);
                }
            }
        }
        if(writeOnPopUp.length()>maxLengthString){
            writeOnPopUp=writeOnPopUp.substring(0,maxLengthString)+"..";
        }
        g.drawString(writeOnPopUp, startWriteWidth+2,
                pointerWriteHight+1, 0);
        pointerWriteHight+=pixelsToNextline;
    }
}else{
    if(adjustablePopUp){
        popUpHightWithScroll= maxNrOfLines*pixelsToNextline;
        pointerWriteHight = (scrH-popUpHightWithScroll)/2;
        startWriteWidth+=4;

        g.setColor(0xFFFFFF);
        g.fillRect(startWriteWidth, pointerWriteHight-7, popUpWidth,
                popUpHightWithScroll+16);
        g.setColor(0xB9B9B9);
        g.drawRect(startWriteWidth, pointerWriteHight-8, popUpWidth,
                popUpHightWithScroll+17);
        g.setColor(0x000000);
        g.drawRect(startWriteWidth-1, pointerWriteHight-9, popUpWidth+2,
                popUpHightWithScroll+19);

        if(uparrow) g.fillTriangle((scrW/2)-3,
                ((scrH-popUpHightWithScroll)/2)-2,(scrW/2)+3,
                ((scrH-popUpHightWithScroll)/2)-2,(scrW/2),
                ((scrH-popUpHightWithScroll)/2)-7);
        if(downarrow) g.fillTriangle((scrW/2)-3,
                ((scrH-popUpHightWithScroll)/2)+popUpHightWithScroll+2,
                (scrW/2)+3,((scrH-popUpHightWithScroll)/2)+
                popUpHightWithScroll+2,(scrW/2),
                ((scrH-popUpHightWithScroll)/2)+
                (popUpHightWithScroll+7));
    }else{
        g.setColor(color);
```

```
        if(uparrow) g.fillTriangle((scrW/2)-3,pointerWriteHight+5,
                (scrW/2)+3,pointerWriteHight+5,(scrW/2),
                pointerWriteHight);
        if(downarrow) g.fillTriangle((scrW/2)-3,scrH-7,
                (scrW/2)+3,scrH-7,(scrW/2),scrH-2);
        if (highlight) pointerWriteHight+=6;
        else pointerWriteHight+=4;
        startWriteWidth-=2;

    }

    if(highlight){
        for(l=top; l<=bottom; l++){
            g.setFont(medium);
            g.setColor(color);
            if (l==t){
                g.setColor(colormarkingbar);
                if(adjustablePopUp)g.fillRect(startWriteWidth+1,
                        pointerWriteHight+1, popUpWidth-2,
                        pixelsToNextline);
                else g.fillRect(startWriteWidth+1, pointerWriteHight+2,
                        popUpWidth-2, pixelsToNextline);
                g.setColor(color);
                if(differnethighlight){
                    g.setColor(0xFF0000);
                    g.setFont(boldmedium);
                }
            }
            writeOnPopUp=elementsOnScreen[l];
            if(writeOnPopUp.length()>maxLengthString){
                writeOnPopUp=writeOnPopUp.
                        substring(0,maxLengthString)+"..";
            }
            g.drawString(writeOnPopUp, startWriteWidth+4,
                    pointerWriteHight+4, 0);
            pointerWriteHight+=pixelsToNextline;
        }
    }else{
        g.setColor(color);

        for(l=t; l<maxNrOfLines+t; l++){
            g.drawString(elementsOnScreen[l], startWriteWidth+2,
                    pointerWriteHight+7, 0);
            pointerWriteHight+=pixelsToNextline;
        }
```

```
            }
        }
    }

//XXXXXXXXXXXXXXXXXXXXX   GENERAL PAINT STUFF END  XXXXXXXXXXXXXXXXXXXXXXXXX


//XXXXXXXXXXXXXXXXXXXXXXXXXXX   LEVEL 1   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


    /**
     * Displays the main menu with its icons. Also some logic rearding which
     * icon is highlighted.
     * @param g Graphics object
     */
    public void displayMainMenu(Graphics g) {
        g.drawImage(bildeRest, kolonne1, rekke1, 0);
        g.drawImage(bildeGo, kolonne2, rekke1, 0);
        g.drawImage(bildeTrinken, kolonne3, rekke1, 0);
        g.drawImage(bildeFilm, kolonne1, rekke2, 0);
        g.drawImage(bildeTanzen, kolonne2, rekke2, 0);
        g.drawImage(bildeHoren, kolonne3, rekke2, 0);
        g.drawImage(bildeMap, kolonne1, rekke3, 0);
        g.drawImage(bildeSuche, kolonne2, rekke3, 0);
        g.drawImage(bildeFavoritt, kolonne3, rekke3, 0);

        if(y==1){
            if(x==1){
                g.drawImage(bildeRest2, kolonne1-corrH, rekke1-corrW, 0);
                catagoryHeadline="Essen";
                ApplicationData.searchType="essen";
                currentSearchDisplay=THEME_ESSEN;
                antz=3;
            }
            if(x==2){
                g.drawImage(bildeGo2, kolonne2-corrH, rekke1-corrW, 0);
                catagoryHeadline="Planen";
                currentSearchDisplay=THEME_PLANEN;
                antz=3;
            }
            if(x==3){
                g.drawImage(bildeTrinken2, kolonne3-corrH, rekke1-corrW, 0);
                catagoryHeadline="Trinken";
                ApplicationData.searchType="trinken";
                currentSearchDisplay=THEME_TRINKEN;
```

```
                antz=3;
            }
        }
        if(y==2){
            if(x==1){
                g.drawImage(bildeFilm2, kolonne1-corrH, rekke2-corrW, 0);
                catagoryHeadline="Sehen";
                ApplicationData.searchType="sehen";
                currentSearchDisplay=THEME_SEHEN;
                antz=4;
            }
            if(x==2){
                g.drawImage(bildeTanzen2, kolonne2-corrH, rekke2-corrW, 0);
                catagoryHeadline="Tanzen";
                ApplicationData.searchType="tanzen";
                currentSearchDisplay=THEME_TANZEN;
                antz=2;
            }
            if(x==3){
                g.drawImage(bildeHoren2, kolonne3-corrH, rekke2-corrW, 0);
                catagoryHeadline="Hören";
                ApplicationData.searchType="konzerte";
                currentSearchDisplay=THEME_HOREN;
                antz=2;
            }
        }
        if(y==3){
            if(x==1){
                g.drawImage(bildeMap2, kolonne1-corrH, rekke3-corrW, 0);
                catagoryHeadline="Karte";
            }
            if(x==2){
                g.drawImage(bildeSuche2, kolonne2-corrH, rekke3-corrW, 0);
                catagoryHeadline="Ergebnisse";
            }
            if(x==3){
                g.drawImage(bildeFavoritt2, kolonne3-corrH, rekke3-corrW, 0);
                catagoryHeadline="Favoriten";
            }
        }
    }

    /**
     * Resets the values displayed in the fields of the search pages. The midl..
     * are the strings that are shown on the buttons. (For instance: "Altstadt"
```

```
     * under city district)
     */
    public void resetdefaults(){
        midlEssenCategory=ApplicationData.defaultEssenCategory;
        midlCityDistrictEssen= ApplicationData.defaultCityDistrictEssen;
        midlEventType= ApplicationData.defaultEventType;
        midlDate=ApplicationData.defaultDate;
        midlTrinkenCategory= ApplicationData.defaultTrinkenCategory;
        midlDistrictTrinken= ApplicationData.defaultcityDistrictTrinken;
        midlCinema=ApplicationData.defaultCinema;
        midlCityDistrictTanzen=ApplicationData.defaultCityDistrictTanzen;
    }

    /**
     * Closing the "loading"-pop-up which is shown when loading the map
     */
    public void loadingPopUpOff(){
        currentCanvasPopUpDisplay2=NO_POP_UP2;
        repaint();
    }

    /**
     * Called when fire/enter is pressed when in the main menu.
     * Makes sure that the right functionality is started.
     */
    public void choosefromlevel1(){
        if(y==antHoyde){
            if(x==1){
                applicationData.showMapWithDevicePosition();
                currentCanvasPopUpDisplay2=POP_UP2_LOADING;
            }else if(x==2){
                currentCanvasDisplay=MULTIPLE_SEARCH_RESULT;
                level=4;
                hoydeinfoBar=header.getHeight()+tab1.getHeight();
                maxPopupElementerLevel4multisearchresult();
                resetlevel4values();
            }
        }else{
            level=2;
            currentCanvasDisplay=SEARCH_PAGE;
        }
    }

//XXXXXXXXXXXXXXXXXXXXXXXXX   LEVEL 1 END  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
//XXXXXXXXXXXXXXXXXXXXXXXXXXXX   LEVEL 2 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

    /**
     * Displayes the page used for searching for entities. This methode calls
     * the different display...ThemePage-methods according to how
     * currentSearchDisplay is set. currentSearchDisplay==Essen means the
     * resaurant search page will be displayed.
     * @param g Graphics object
     * @see displayHorenThemePage
     * @see displayEssenThemePage
     * @see displayPlanenThemePage
     * @see displayTrinkenThemePage
     * @see displaySehenThemePage
     * @see displayTanzenThemePage
     * @see displayHorenThemePage
     */
    public void displaySearchPage(Graphics g){
        sokestrenghoyde = (scrH-button2.getHeight()-hoydeinfoBar)/2;
        for(int i=1; i<antz; i++){
            if(i==z) g.drawImage(button2, 0, sokestrenghoyde-buttonramme, 0);
            else g.drawImage(button, buttonramme/2, sokestrenghoyde, 0);
            sokestrenghoyde = sokestrenghoyde + button.getHeight()+buttonramme;
        }
        //Search button
        if(z==antz) g.drawImage(button2, 0, scrH-button2.getHeight(), 0);
        else g.drawImage(button, buttonramme/2,
                scrH-button2.getHeight()+buttonramme, 0);
        g.setColor(0xFFFFFF);
        g.setFont(boldsmall);
        g.drawString("Suchen", textdiffx,
                scrH-button2.getHeight()+buttonramme+textdiffy,0);
        sokestrenghoyde = (scrH-button2.getHeight()-hoydeinfoBar)/2;
        g.setFont(small);

        if(currentSearchDisplay==THEME_ESSEN)displayEssenThemePage(g);
        if(currentSearchDisplay==THEME_PLANEN)displayPlanenThemePage(g);
        if(currentSearchDisplay==THEME_TRINKEN)displayTrinkenThemePage(g);
        if(currentSearchDisplay==THEME_SEHEN)displaySehenThemePage(g);
        if(currentSearchDisplay==THEME_TANZEN)displayTanzenThemePage(g);
        if(currentSearchDisplay==THEME_HOREN)displayHorenThemePage(g);
    }

    /**
```

```
 * Together with displaySearchPage displays the essen search page.
 * @param g Graphics object
 */
public void displayEssenThemePage(Graphics g) {
    ApplicationData.searchCategory=midlEssenCategory;
    ApplicationData.searchCityDistrict=midlCityDistrictEssen;
    g.drawString("Kategorie: "+shortenStringOnButton(midlEssenCategory, 12),
            textdiffx, sokestrenghoyde+textdiffy, 0);
    g.drawString("Stadtteil: "+shortenStringOnButton(midlCityDistrictEssen,
            11), textdiffx, sokestrenghoyde+button.getHeight()+
            buttonramme+textdiffy, 0);
}

/**
 * Together with displaySearchPage displays the planen search page.
 * @param g Graphics object
 */
public void displayPlanenThemePage(Graphics g) {
    ApplicationData.searchType=midlEventType;
    ApplicationData.searchDate=midlDate;
    g.drawString("Kategorie: "+shortenStringOnButton(midlEventType, 12),
            textdiffx, sokestrenghoyde+textdiffy, 0);
    g.drawString("Tag: "+shortenStringOnButton(midlDate, 5), textdiffx,
            sokestrenghoyde+button.getHeight()+buttonramme+textdiffy, 0);
}

/**
 * Together with displaySearchPage displays the trinken search page.
 * @param g Graphics object
 */
public void displayTrinkenThemePage(Graphics g) {
    ApplicationData.searchCategory=midlTrinkenCategory;
    ApplicationData.searchCityDistrict=midlDistrictTrinken;
    g.drawString(
            "Kategorie: "+shortenStringOnButton(midlTrinkenCategory, 12),
            textdiffx, sokestrenghoyde+textdiffy, 0);
    g.drawString(
            "Stadtteil: "+shortenStringOnButton(midlDistrictTrinken, 11),
            textdiffx, sokestrenghoyde+button.getHeight()+buttonramme+
            textdiffy, 0);
}

/**
 * Together with displaySearchPage displays the sehen search page.
 * @param g Graphics object
```

```
 */
public void displaySehenThemePage(Graphics g) {
    g.drawString("Film: ", textdiffx, sokestrenghoyde+textdiffy, 0);
    g.drawString("Kino: "+shortenStringOnButton(midlCinema, 5), textdiffx,
            sokestrenghoyde+button.getHeight()+buttonramme+textdiffy, 0);
    g.drawString("Tag: "+shortenStringOnButton(midlDate, 4), textdiffx,
            sokestrenghoyde+2*(button.getHeight()+buttonramme)+textdiffy,
            0);
}

/**
 * Together with displaySearchPage displays the tanzen search page.
 * @param g Graphics object
 */
public void displayTanzenThemePage(Graphics g) {
    ApplicationData.searchCityDistrict=midlCityDistrictTanzen;
    g.drawString("Stadtteil:  "+
            shortenStringOnButton(midlCityDistrictTanzen,
            11), textdiffx, sokestrenghoyde+textdiffy, 0);
}

/**
 * Together with displaySearchPage displays the hören search page.
 * @param g Graphics object
 */
public void displayHorenThemePage(Graphics g) {
    ApplicationData.searchDate=midlDate;
    g.drawString("Tag:    "+shortenStringOnButton(midlDate, 5), textdiffx,
            sokestrenghoyde+textdiffy, 0);
}

/**
 * Depending on what search type that has been chosen and what button is
 * highlighted, this methode initiates what is to happen when the wählen
 * softkey or enter key is pressed. z says which button is highlighted.
 * currentSearchDisplay which search type. popUpMenu starts a popUp for
 * specifying a search (for ex. which city district?)
 * performPlaceSearchEvent.fireEvent starts a search with use of the net
 * Result in searchDataAvailableEvent or searchResultEmptyEvent
 * @see popUpMenu
 * @see performPlaceSearchEvent.fireEvent
 * @see searchResultEmptyEvent
 * @see searchDataAvailableEvent
 */
public void choosefromundercatagoryorsearch(){
```

```
if(currentSearchDisplay==THEME_ESSEN){
    if(z==1)popUpMenu(currentSearchDisplay, TYPE_CATEGORY);
    if(z==2)popUpMenu(currentSearchDisplay, TYPE_CITY_DISTRICT);
    if(z==3){
        if(ApplicationData.OFFLINE)performSearch();
        else{
            performPlaceSearchEvent.fireEvent();
            currentCanvasPopUpDisplay=POP_UP_SEARCHING;
        }
        ApplicationData.defaultEssenCategory=midlEssenCategory;
        ApplicationData.defaultCityDistrictEssen=midlCityDistrictEssen;
    }
}
if(currentSearchDisplay==THEME_PLANEN){
    if(z==1)popUpMenu(currentSearchDisplay, TYPE_CATEGORY);
    if(z==2)popUpMenu(currentSearchDisplay, TYPE_DATE);
    if(z==3){
        ApplicationData.searchType=midlEventType;
        performCityEventSearchEvent.fireEvent();
        ApplicationData.defaultEventType=midlEventType;
        ApplicationData.defaultDate=midlDate;
        currentCanvasPopUpDisplay=POP_UP_SEARCHING;
    }
}
if(currentSearchDisplay==THEME_TRINKEN){
    if(z==1)popUpMenu(currentSearchDisplay, TYPE_CATEGORY);
    if(z==2)popUpMenu(currentSearchDisplay, TYPE_CITY_DISTRICT);
    if(z==3){
        performPlaceSearchEvent.fireEvent();
        ApplicationData.defaultTrinkenCategory=midlTrinkenCategory;
        ApplicationData.defaultcityDistrictTrinken=midlDistrictTrinken;
        currentCanvasPopUpDisplay=POP_UP_SEARCHING;
    }
}
if(currentSearchDisplay==THEME_SEHEN){
    if(z==1)popUpMenu(currentSearchDisplay, TYPE_FILM);
    if(z==2)popUpMenu(currentSearchDisplay, TYPE_CINEMA);
    if(z==3)popUpMenu(currentSearchDisplay, TYPE_DATE);
}
if(currentSearchDisplay==THEME_TANZEN){
    if(z==1)popUpMenu(currentSearchDisplay, TYPE_CITY_DISTRICT);
    if(z==2){
        performPlaceSearchEvent.fireEvent();
        ApplicationData.defaultCityDistrictTanzen=
                midlCityDistrictTanzen;
```

```
                currentCanvasPopUpDisplay=POP_UP_SEARCHING;
            }
        }
        if(currentSearchDisplay==THEME_HOREN){
            if(z==1)popUpMenu(currentSearchDisplay, TYPE_DATE);
            if(z==2){
                performCityEventSearchEvent.fireEvent();
                ApplicationData.defaultDate=midlDate;
                currentCanvasPopUpDisplay=POP_UP_SEARCHING;
            }
        }
        resetlevel4values();
    }

//XXXXXXXXX Must be changed for other mobile! XXXXXXXXXXXX
    /**
     * Made to fit a string to a button to make sure it isn't too long
     * Shotens a string according to setMaxStringOnButton which says how long
     * the sting can be. Returns the shortend string
     * @param inarg String to shorten
     * @param otherLength length of string on button in addition to inarg
     * @return the new improved string with a ".." if shortend
     */
    public String shortenStringOnButton(String inarg, int otherLength){
        String retarg=inarg;
        int maxStringOnButton=setMaxStringOnButton();
        if((retarg.length()+otherLength)>maxStringOnButton){
            retarg=retarg.substring(0,maxStringOnButton-otherLength)+"..";
        }
        return retarg;
    }

    /**
     * Sets the string length to buttons.
     * Must be changed for other mobile, when the button size is different
     * @return number of letters allowed
     * @see shortenStringOnButton
     */
    public int setMaxStringOnButton(){
        int temp;
        if(mobtype==ERICSSON) temp=button.getWidth()/6;
        else if(mobtype==NETBEANS) temp=button.getWidth()/6;
        else if (mobtype==NOKIA) temp=button.getWidth()/6;
        else temp=button.getWidth()/6;
        return temp;
```

```
    }

//XXXXXXXXXXXXXXXXXXXXXXXXXXX LEVEL 2 END XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


//XXXXXXXXXXXXXXXXXXXXXXXXXXX   LEVEL 3 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

    /**
     * Starts the displaying of a popup menu, and sets some parameters.
     * drawScrollingPage draws the menu.
     * @param g Graphics object
     * @see drawScrollingPage
     */
    public void displayPopUpMenu(Graphics g){
        int maxStringlength = setMaxStringLengthLevel3();
        int popUpBredde= setPopUpBredde(longestLength, maxStringlength);
        drawScrollingPage(g, generalmenu, antElementerLevel3and4,
                setMaxPopupElementerLevel3, maxStringlength,
                small.getHeight()+2, popUpBredde, -1, -1, (scrW-popUpBredde)/2,
                -1, small, true, true, false, 0x000000, 0xD5D5D5);
    }

    /**
     * Is called from choosefromundercatagoryorsearch
     * Sets the right parameters to start pop-up viewing
     * @param themeID currentSearchDisplay
     * @param type also called z. Says what to search for
     */
    public void popUpMenu(int themeID, int type) {
        this.themeID=themeID;
        this.type=type;
        if(themeID==THEME_ESSEN){
            if(type==TYPE_CATEGORY) {
                generalmenu = ApplicationData.essenCategories;
            }
            if(type==TYPE_CITY_DISTRICT){
                generalmenu = ApplicationData.cityDistrictsEssen;
            }
        }else if(themeID==THEME_PLANEN){
            if(type==TYPE_CATEGORY) generalmenu = ApplicationData.eventTypes;
            if(type==TYPE_DATE) generalmenu = getNextDates();
        }else if(themeID==THEME_TRINKEN){
            if(type==TYPE_CATEGORY) {
                generalmenu = ApplicationData.trinkenCategories;
            }
```

```
            if(type==TYPE_CITY_DISTRICT) {
                generalmenu = ApplicationData.cityDistrictsTrinken;
            }
        }else if(themeID==THEME_SEHEN){
//            if(type==TYPE_FILM)
//            if(type==TYPE_CINEMA)
//            if(type==TYPE_DATE)
        }else if(themeID==THEME_TANZEN){
            if(type==TYPE_CITY_DISTRICT) {
                generalmenu = ApplicationData.cityDistrictsEssen;
            }
        }else if(themeID==THEME_HOREN){
            if(type==TYPE_DATE) generalmenu = getNextDates();
        }
        antElementerLevel3and4 = generalmenu.length;
        longestLength = getLengthLongest(generalmenu, antElementerLevel3and4);
        currentCanvasPopUpDisplay=POP_UP;
        level=3;
    }

    /**
     * Action taken when fire/enter is pressed when a pop-up is viewed
     */
    public void popUpFire(){
        if(themeID==THEME_ESSEN){
            if(type==TYPE_CATEGORY){
                ApplicationData.searchCategory=generalmenu[t];
                midlEssenCategory=generalmenu[t];
            }
            if(type==TYPE_CITY_DISTRICT){
                ApplicationData.searchCityDistrict=generalmenu[t];
                midlCityDistrictEssen=generalmenu[t];
            }
        }else if(themeID==THEME_PLANEN){
            if(type==TYPE_CATEGORY){
                ApplicationData.searchType=generalmenu[t];
                midlEventType=generalmenu[t];
            }
            if(type==TYPE_DATE){
                ApplicationData.searchDate=generalmenu[t];
                midlDate=generalmenu[t];
            }
        }else if(themeID==THEME_TRINKEN){
            if(type==TYPE_CATEGORY){
                ApplicationData.searchCategory=generalmenu[t];
```

```
                midlTrinkenCategory=generalmenu[t];
            }
            if(type==TYPE_CITY_DISTRICT){
                ApplicationData.searchCityDistrict=generalmenu[t];
                midlDistrictTrinken=generalmenu[t];
            }
        }else if(themeID==THEME_SEHEN){
//             if(type==TYPE_FILM)
//             if(type==TYPE_CINEMA)
//             if(type==TYPE_DATE)
        }else if(themeID==THEME_TANZEN){
            if(type==TYPE_CITY_DISTRICT){
                ApplicationData.searchCityDistrict=generalmenu[t];
                midlCityDistrictTanzen=generalmenu[t];
            }
        }else if(themeID==THEME_HOREN){
            if(type==TYPE_DATE){
                ApplicationData.searchDate=generalmenu[t];
                midlDate=generalmenu[t];
            }
        }
        currentCanvasPopUpDisplay=NO_POP_UP;
        level=2;
        resetlevel3values();
    }


    /**
     * Action taken when down is pressed when a pop-up is viewed
     */
    public void popUpDown(){
        if((antElementerLevel3and4>setMaxPopupElementerLevel3||
                antElementerLevel3and4<=setMaxPopupElementerLevel3)&&
                ((t+1)==antElementerLevel3and4)){
        } else if((antElementerLevel3and4>setMaxPopupElementerLevel3)&&
                (t==bottom)){
            top++;
            bottom++;
            t++;
            uparrow=true;
            if((t+1)==antElementerLevel3and4) downarrow=false;
        } else t++;
    }

    /**
```

```
 * Action taken when up is pressed when a pop-up is viewed
 */
public void popUpUp(){
    if(t==0){
    }else if((antElementerLevel3and4>setMaxPopupElementerLevel3)&&(t==top)){
        top--;
        bottom--;
        t--;
        downarrow=true;
        if (t==0) uparrow=false;
    } else t--;
}

/**
 * Resets important level 3 values when leaving a level 3 search page
 */
public void resetlevel3values(){
    t=0;
    top=0;
    bottom=setMaxPopupElementerLevel3-1;
    uparrow = false;
    downarrow = true;
}

/**
 * Sets the width of a pop-up in piksels
 * @param longestLength longest string length in an array of strings
 * @param maxStringlength the maximum allowed string length
 * @return an int of the maximum wdth
 */
public int setPopUpBredde(int longestLength, int maxStringlength){
    int popUpWidth;
    if (longestLength>maxStringlength) popUpWidth=scrW-20;
    else if (longestLength<12) popUpWidth= 78;
    else popUpWidth=(longestLength*6)+4;
    return popUpWidth;
}


/**
 * Sets maximum pop-up elements to be displayed on popups where for instance
 * type of restaurant and city district is chosen
 */
private void maxPopupElementerLevel3() {
    if(mobtype==ERICSSON) {
```

```
            setMaxPopupElementerLevel3=(scrH-40)/(boldmedium.getHeight()+2);
        }
        else if(mobtype==NETBEANS){
            setMaxPopupElementerLevel3=(scrH-40)/(boldmedium.getHeight()+2);
        }
        else if(mobtype==NOKIA) setMaxPopupElementerLevel3=9;
        else setMaxPopupElementerLevel3=(scrH-40)/(boldmedium.getHeight()+2);
    }

    /**
     * Sets the maximum string length allowed on a pop-up according to what
     * mobile is being used.
     * Must be changed for other mobiles
     * @return the maximum length
     */
    public int setMaxStringLengthLevel3(){
        int temp;
        if(mobtype==ERICSSON) temp=scrW/7;
        else if(mobtype==NETBEANS) temp=scrW/7;
        else if(mobtype==NOKIA) temp=scrW/7;
        else temp=scrW/7;
        return temp;
    }

//XXXXXXXXXXXXXXXXXXXXXXXXX   LEVEL 3 END   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//XXXXXXXXXXXXXXXXXXXXXXXXX   LEVEL 4   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

    /**
     * Displays search results according to what kind of search has been made.
     * @param g Graphics object
     * @see drawScrollingPage
     */
    public void displaySearchResults(Graphics g){
        g.setColor(0x000000);
        g.fillRect( 0, hoydeinfoBar, scrW, scrH-hoydeinfoBar);
        if(ApplicationData.OFFLINE){
            drawScrollingPage(g, searchResultNamesLevel4,
                    antElementerLevel3and4,
                    setMaxPopupElementerLevel4, setMaxStringLengthLevel4(),
                    medium.getHeight(), scrW, scrH-hoydeinfoBar,
                    scrH-hoydeinfoBar, 0, hoydeinfoBar+3, medium, true, false,
                    true, 0xFFFFFF, 0xFFFFFF);
        }else if (ApplicationData.searchTheme.compareTo("placeSearch")==0) {
            drawScrollingPage(g, placeResultNamesLevel4, antElementerLevel3and4,
```

```
                setMaxPopupElementerLevel4, setMaxStringLengthLevel4(),
                medium.getHeight(), scrW, scrH-hoydeinfoBar,
                scrH-hoydeinfoBar, 0, hoydeinfoBar+3, medium, true, false,
                true, 0xFFFFFF, 0xFFFFFF);
        }else{
            drawScrollingPage(g, cityEventResultNamesLevel4,
                    antElementerLevel3and4, setMaxPopupElementerLevel4,
                    setMaxStringLengthLevel4(), medium.getHeight(), scrW,
                    scrH-hoydeinfoBar, scrH-hoydeinfoBar, 0, hoydeinfoBar+3,
                    medium, true, false, true, 0xFFFFFF, 0xFFFFFF);
        }
    }

    /**
     * Sets values if search is of the type place
     * @param placeResult array with place objects
     */
    public void startDisplayPlaceResults(Place[] placeResult) {
        placeResultLevel4=placeResult;
        antElementerLevel3and4= placeResult.length;
        placeResultNamesLevel4 = new String[antElementerLevel3and4];
        for(int k=0; k<antElementerLevel3and4; k++){
            placeResultNamesLevel4[k]=placeResultLevel4[k].getName();
        }
    }
    /**
     * Sets values if search is of the type city event
     * @param cityEventResult array with city event objects
     */
    public void startDisplayCityEventResults(CityEvent[] cityEventResult) {
        cityEventResultLevel4=cityEventResult;
        antElementerLevel3and4= cityEventResult.length;
        cityEventResultNamesLevel4 = new String[antElementerLevel3and4];
        for(int k=0; k<antElementerLevel3and4; k++){
            cityEventResultNamesLevel4[k]=cityEventResultLevel4[k].getName();
        }
    }


    /**
     * Offline methode. Sets values if search is of the type searchresult
     * @param searchResult Array of searchresult objects
     */
    public void startDisplaySearchResultsalt(SearchResult[] searchResult) {
        searchResultLevel4=searchResult;
```

```
    antElementerLevel3and4= searchResult.length;
    searchResultNamesLevel4 = new String[antElementerLevel3and4];
    for(int k=0; k<antElementerLevel3and4; k++){
        searchResultNamesLevel4[k]=searchResultLevel4[k].getName();
    }
}

/**
 * Resets important level 4 values when leaving a level 4 search result page
 */
public void resetlevel4values(){
    t=0;
    top=0;
    bottom=setMaxPopupElementerLevel4-1;
}

/**
 * Saves level 4 value (search result page), when leaving the page and
 * entering an entity. Collected by retrivelevel4values
 */
public void savelevel4values(){
    savedt=t;
    savedtop=top;
    savedbottom=bottom;
    savetoparrow=uparrow;
    savebottomarrow=downarrow;
}

/**
 * Retrieves the values saved in savelevel4values
 */
public void retrivelevel4values(){
    t=savedt;
    top=savedtop;
    bottom=savedbottom;
    uparrow=savetoparrow;
    downarrow=savebottomarrow;
}

/**
 * Starts action when the fire/enter button has been pressed when having
 * chosen an entity from the search result menu
 */
public void searchResultFire(){
    currentCanvasPopUpDisplay= POP_UP_ELEMENT_INFO;
```

```
        if(ApplicationData.OFFLINE){
            searchResultElementForLevel5=searchResultLevel4[t];
        }
        else if(ApplicationData.searchTheme.compareTo("placeSearch")==0||
                ApplicationData.OFFLINE) {
            placeResultElementForLevel5=placeResultLevel4[t];
        }
        else cityEventElementForLevel5=cityEventResultLevel4[t];
        level=5;
        savelevel4values();
        resetlevel5values();
    }

    /**
     * Retrives information if user navigates back to level 4
     */
    private void backLevel4() {
        if(currentCanvasDisplay==SEARCH_RESULT){
            level=2;
            currentCanvasDisplay=SEARCH_PAGE;
            resetlevel3values();
        }else{
            level=1;
            currentCanvasDisplay=MAIN_MENU;
            maxPopupElementerLevel4();
            resetlevel4values();
            hoydeinfoBar=header.getHeight();
            currentCanvasPopUpDisplay=NO_POP_UP;
            doSearch=true;
            u=1;
        }
    }

    /**
     * Used for drawing five searchresults. Used when viewing the last search
     * alternative availible from the main menu
     * @param g Graphics object
     */
    public void displaymultiplesearchresult(Graphics g){
        if(u==1)g.drawImage(tab1, 0, header.getHeight(), 0);
        if(u==2)g.drawImage(tab2, 0, header.getHeight(), 0);
        if(u==3)g.drawImage(tab3, 0, header.getHeight(), 0);
        if(u==4)g.drawImage(tab4, 0, header.getHeight(), 0);
        if(u==5)g.drawImage(tab5, 0, header.getHeight(), 0);
```

```
    ApplicationData.lastSearchNumber=u;
    if(doSearch){
        if(ApplicationData.OFFLINE){
            SearchResult[] searchResultLevel4 = getearliersearches();
            startDisplaySearchResultsalt(searchResultLevel4);
        }else{
            restoreLastSearchData.fireEvent();
            doSearch=false;
            currentCanvasPopUpDisplay=POP_UP_SEARCHING;
            repaint();
        }
    }else if(currentCanvasPopUpDisplay==POP_UP_NO_RESULTS_FOUND||
            currentCanvasPopUpDisplay==POP_UP_SEARCHING){
    }else{
        displaySearchResults(g);
    }
}


/**
 * Offline methode. Gathers information to be displayed on last search list
 * @return Returns an array of searchresult objects
 */
private SearchResult[] getearliersearches() {
    SearchResult[] searchResultLevel4=null;
    if(u==1){
        searchResultLevel4 = searchService.performSearch
                ("Essen", " "," ",Calendar.getInstance().getTime());
    }
    if(u==2){
        searchResultLevel4 = searchService.performSearch
                ("Trinken", " "," ",Calendar.getInstance().getTime());
    }
    if(u==3){
        searchResultLevel4 = searchService.performSearch
                ("Horen", " "," ",Calendar.getInstance().getTime());
    }
    if(u==4){
        searchResultLevel4 = searchService.performSearch
                ("Sehen", " "," ",Calendar.getInstance().getTime());
    }
    if(u==5){
        searchResultLevel4 = searchService.performSearch
                ("Planen", " "," ",Calendar.getInstance().getTime());
    }
```

```
        return searchResultLevel4;
    }

    /**
     * Sets the maximum elements to be shown on the last search result list
     * according to type of phone
     */
    private void maxPopupElementerLevel4multisearchresult() {
        if(mobtype==ERICSSON) setMaxPopupElementerLevel4=6;
        else if(mobtype==NETBEANS) setMaxPopupElementerLevel4=15;
        else if(mobtype==NOKIA) setMaxPopupElementerLevel4=8;
        else setMaxPopupElementerLevel4=6;
    }


    /**
     * Sets the maximum elements to be shown on the search result list
     * according to type of phone
     */
    private void maxPopupElementerLevel4() {
        if(mobtype==ERICSSON) setMaxPopupElementerLevel4=7;
        else if(mobtype==NETBEANS) setMaxPopupElementerLevel4=16;
        else if(mobtype==NOKIA) setMaxPopupElementerLevel4=9;
        else setMaxPopupElementerLevel4=7;
    }

    /**
     * Sets the maximum string length allowed on search result page (level 4)
     * to what mobile is being used.
     * Must be changed for other mobiles
     * @return maximum string length level 4
     */
    public int setMaxStringLengthLevel4(){
        int temp;
        if(mobtype==ERICSSON) temp=scrW/7;
        else if(mobtype==NETBEANS) temp=scrW/7;
        else if(mobtype==NOKIA) temp=30;
        else temp=scrW/7;
        return temp;
    }

//XXXXXXXXXXXXXXXXXXXXXXXX   LEVEL 4 END   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

//XXXXXXXXXXXXXXXXXXXXXXXXXXXX   LEVEL 5   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
/**
 * Displays the content of an entity. Shows all the important information
 * that needs to be handled in connection with an entity. Also starts the
 * functionality it uses.
 * Uses drawScrollingPage to display the description
 * @param g Graphics object
 */
public void displayElement(Graphics g){
    g.setColor(0xFFFFFF);
    g.fillRect( elementPopUpGapSide, elementPopUpGapTop+was.getHeight()-2,
            scrW-elementPopUpGapSide,
            scrH-elementPopUpGapTop-was.getHeight()+2);
    g.setColor(0xB9B9B9);
    g.drawRect(elementPopUpGapSide+1, elementPopUpGapTop+was.getHeight()-1,
            scrW-elementPopUpGapSide-2,
            scrH-elementPopUpGapTop-was.getHeight());
    g.setColor(0x000000);
    g.drawRect(elementPopUpGapSide, elementPopUpGapTop+was.getHeight()-2,
            scrW-elementPopUpGapSide,
            scrH-elementPopUpGapTop-was.getHeight()+2);
    boolean place=false;
    if(ApplicationData.searchTheme.compareTo("placeSearch")==0||
            ApplicationData.OFFLINE) place=true;
    if(descriptionString==null){
        if(ApplicationData.OFFLINE){
            descriptionString =
                    searchResultElementForLevel5.getDescription();
            descriptionStringArray = makeNewStringArray(descriptionString,
                    nrOfLettersPrLineOnPopUpElement());
        }else if(place){
            descriptionString =
                    placeResultElementForLevel5.getDescription();
            descriptionStringArray = makeNewStringArray(descriptionString,
                    nrOfLettersPrLineOnPopUpElement());
        }else{
            descriptionString = cityEventElementForLevel5.getDescription();
            descriptionStringArray = makeNewStringArray(descriptionString,
                    nrOfLettersPrLineOnPopUpElement());
        }
    }
    switch(w){
        case 1:
            g.drawImage(was, elementPopUpGapSide, elementPopUpGapTop, 0);
            yHight = elementPopUpGapTop+was.getHeight()+textgap;
```

```
g.setFont(headline);
if(place){//For place
    if(ApplicationData.OFFLINE) g.drawString(
            searchResultElementForLevel5.getName(),
            elementPopUpGapSide+textgap, yHight, 0);
    else g.drawString(placeResultElementForLevel5.getName(),
            elementPopUpGapSide+textgap, yHight, 0);
    yHight = yHight + textgap + headline.getHeight();
    int nrofstars= Integer.parseInt(String.valueOf(
            placeResultElementForLevel5.getPoll().getValue()));
    int maxnrofstars=6;
    for(int q=0; q<maxnrofstars; q++){
        if(q<nrofstars) g.drawImage(star,
                elementPopUpGapSide+textgap+9*q, yHight+1, 0);
        else g.drawImage(star2, elementPopUpGapSide+textgap+9*q,
                yHight+1, 0);
    }
    g.setFont(medium);
    yHight+=medium.getHeight();
     if(!ApplicationData.OFFLINE){
        g.drawString(placeResultElementForLevel5.getCategory(),
                elementPopUpGapSide+textgap, yHight, 0);
        yHight+=medium.getHeight();
    }
    if(ApplicationData.OFFLINE){
        g.drawString(searchResultElementForLevel5.getAddress(),
                elementPopUpGapSide+textgap, yHight, 0);
    }
    else {
        g.drawString(placeResultElementForLevel5.
                getPlaceAddress().getStreet()+", "+
                placeResultElementForLevel5.getPlaceAddress().
                getCity(), elementPopUpGapSide+textgap, yHight,
                0);
    }
    g.setColor(0xD5D5D5);
    g.fillRect(elementPopUpGapSide+2,
            yHight+medium.getHeight()*v-1,
            scrW-elementPopUpGapSide*2-4, medium.getHeight());
    g.setColor(0x483d8b);
    g.setFont(mediumunderlined);
    if(ApplicationData.OFFLINE){
        g.drawString("Kontakt "+searchResultElementForLevel5.
                getTelephone(), elementPopUpGapSide+textgap,
                yHight + medium.getHeight(), 0);
```

```
            }
            else g.drawString("Kontakt "+placeResultElementForLevel5.
                    getPhone_number(), elementPopUpGapSide+textgap,
                    yHight + medium.getHeight(), 0);
            g.drawString("Standort", elementPopUpGapSide+textgap,
                    yHight + mediumunderlined.getHeight()*2, 0);
            g.drawString("Bewertung ", elementPopUpGapSide+textgap,
                    yHight + mediumunderlined.getHeight()*3, 0);
            g.drawString("Zu Favoriten hinzufügen",
                    elementPopUpGapSide+textgap, yHight +
                    mediumunderlined.getHeight()*4, 0);
        } else{//For cityevent
            g.drawString(cityEventElementForLevel5.getName(),
                    elementPopUpGapSide+textgap, yHight, 0);
            yHight = yHight + textgap + headline.getHeight();
            g.setFont(medium);
            g.drawString(cityEventElementForLevel5.getPlace().getName(),
                    elementPopUpGapSide+textgap, yHight, 0);
            yHight = yHight + medium.getHeight();
            g.drawString(cityEventElementForLevel5.getPlace().
                    getPlaceAddress().getStreet()+", "+
                    cityEventElementForLevel5.getPlace().
                    getPlaceAddress().getCity(),
                    elementPopUpGapSide+textgap, yHight, 0);
            yHight = yHight + medium.getHeight();
            g.drawString("Open: "+cityEventElementForLevel5.
                    getOpening_data(), elementPopUpGapSide+textgap,
                    yHight, 0);
            g.setColor(0xD5D5D5);
            g.fillRect(elementPopUpGapSide+2,
                    yHight+medium.getHeight()*v,
                    scrW-elementPopUpGapSide*2-4, medium.getHeight());
            g.setColor(0x483d8b);
            g.setFont(mediumunderlined);
            g.drawString("Standort", elementPopUpGapSide+textgap,
                    yHight + mediumunderlined.getHeight()*1, 0);
            g.drawString("Zu Favoriten hinzufügen",
                    elementPopUpGapSide+textgap, yHight +
                    mediumunderlined.getHeight()*2, 0);
        }
        break;
    case 2:
        g.drawImage(warum, elementPopUpGapSide, elementPopUpGapTop, 0);
        g.setColor(0x000000);
        antElementerLevel5=descriptionStringArray.length;
```

```
            drawScrollingPage(g, descriptionStringArray, antElementerLevel5,
                    setMaxPopupElementerLevel5, 100, small.getHeight(), -1,
                    scrH-was.getHeight()-elementPopUpGapTop,
                    scrH-was.getHeight()-elementPopUpGapTop, 3,
                    was.getHeight()+elementPopUpGapTop, small, false, false,
                    false, 0x000000, 0xD5D5D5);
            break;
    }
}

/**
 * Logic for when the up-key is pressed in a scroll window
 */
private void elementUp() {
    if(w==1){
        if(ApplicationData.searchTheme.compareTo("placeSearch")==0||
                ApplicationData.OFFLINE){
            if(v==1) v=4;
            else v--;
        }else{
            if(v==1) v=2;
            else v--;
        }
    }else{
        if(t==0){
        }else if (antElementerLevel5>(setMaxPopupElementerLevel5+t)){
            t--;
            downarrow=true;
            if (t==0) uparrow=false;
        }else{
            t--;
            downarrow=true;
        }
    }
}

/**
 * Logic for when the down-key is pressed in a scroll window
 */
private void elementDown() {
    if(w==1){
        if(ApplicationData.searchTheme.compareTo("placeSearch")==0||
                ApplicationData.OFFLINE){
            if(v==4) v=1;
            else v++;
```

```
        }else{
            if(v==2) v=1;
            else v++;
        }
    }else{
        if(antElementerLevel5==setMaxPopupElementerLevel5+t){
        }else if(antElementerLevel5>setMaxPopupElementerLevel5+t){
            t++;
            uparrow=true;
            if(antElementerLevel5==setMaxPopupElementerLevel5+t)
                downarrow=false;
        }else{
            t++;
            uparrow=true;
        }
    }
}

/**
 * Resets important level 5 values when leaving a level 5 entity page
 */
public void resetlevel5values(){
    t=0;
    v=1;
    w=1;
    uparrow=false;
    downarrow=true;
}

/**
 * Logic for when the enter-key is pressed in a scroll window
 */
private void elementFire(){
    if((ApplicationData.searchTheme.compareTo("placeSearch")==0&&v==2)||
            (ApplicationData.searchTheme.compareTo("eventSearch")==0&&v==1)){
        if (ApplicationData.OFFLINE){
        }else if(ApplicationData.searchTheme.compareTo("placeSearch")==0){
            applicationData.showMapWithPOI(placeResultElementForLevel5.
                    getPlaceAddress().getLatitude(),
                    placeResultElementForLevel5.getPlaceAddress().
                    getLongitude());
            currentCanvasPopUpDisplay2=POP_UP2_LOADING;
        }else if (ApplicationData.searchTheme.compareTo("eventSearch")==0){
            applicationData.showMapWithPOI(cityEventElementForLevel5.
                    getPlace().getPlaceAddress().getLatitude(),
```

```
                        cityEventElementForLevel5.getPlace().getPlaceAddress().
                        getLongitude());
                currentCanvasPopUpDisplay2=POP_UP2_LOADING;
            }
        }
        if((ApplicationData.searchTheme.compareTo("placeSearch")==0||
                ApplicationData.OFFLINE)&&v==1){
            applicationData.callNumber("");
        }
    }

    /**
     * Maximum number of lines in the description tab of the element. Set
     * according to what phone is in use
     */
    private void maxPopupElementerLevel5() {
        if(mobtype==ERICSSON) setMaxPopupElementerLevel5=8;
        else if(mobtype==NETBEANS) setMaxPopupElementerLevel5=20;
        else if(mobtype==NOKIA) setMaxPopupElementerLevel5=9;
        else  setMaxPopupElementerLevel5=8;
    }

    /**
     * Sets the number of letters allowed to be displayed on an element on one
     * line. Must be changed for other mobiles
     * @return max number of letters
     */
    public int nrOfLettersPrLineOnPopUpElement(){
        int temp;
        if(mobtype==ERICSSON) temp=35;
        else if(mobtype==NETBEANS) temp=35;
        else if(mobtype==NOKIA) temp=31;
        else temp=35;
        return temp;
    }

//XXXXXXXXXXXXXXXXXXXXXXXXXXXX  LEVEL 5 END   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


//XXXXXXXXXXXXXXXXXXXXXXXXXXXX   GENERAL METHODES   XXXXXXXXXXXXXXXXXXXXXXXXXXXX

    /**
     * Takes a string and splits it into several smaller strings that are
     * returned in an array. The strings are split into parts that are to be
     * smaller than a certain max. The method makes sure that words aren't split
```

```
 * in two.
 * @param menuString The string that is to split up
 * @param limitWidth Maximum number of letters allowed for each smaller
 * string
 * @return an array with each element in the array representing one line
 * in the description field
 */
protected String[] makeNewStringArray(String menuString, int limitWidth){
    //limitWidth says how long the string can be.
    int length = menuString.length();
    int fulllength = menuString.length();
    String lineString;
    int lineX = 0;
    int startIndex = 0;
    int endIndex = 0;
    int numberoflines=0;
    while(length > 0){
        endIndex = startIndex + getLineBreak(menuString, startIndex,
                fulllength, limitWidth);
        length -= (endIndex - startIndex);
        startIndex = endIndex;
        numberoflines++;
    }
    String[] returnString = new String[numberoflines];
    startIndex = 0;
    endIndex = 0;
    length=fulllength;
    while(length > 0){
        endIndex = startIndex + getLineBreak(menuString, startIndex,
                fulllength, limitWidth);
        lineString = menuString.substring(startIndex, endIndex);
        returnString[lineX]=lineString;
        lineX++;
        length -= (endIndex - startIndex);
        startIndex = endIndex;
    }
    return returnString;
}

/**
 * Returns the next index for a line break. Takes in a string. Uses a
 * startindex and maxlength. Used by makeNewStringArray and does not split
 * up words.
 * @param menuString The string that the part is extracted from
 * @param startIndex where in the menuString the new string will start
```

```
 * @param length Length of the string
 * @param limitWidth Max length of string on one line
 * @return Index for next line break
 */
public int getLineBreak(String menuString, int startIndex, int length,
        int limitWidth){
    int returnarg;
    String newMenuString = menuString.substring(startIndex, length);
    if (limitWidth < length-startIndex) returnarg=
            newMenuString.lastIndexOf(' ', limitWidth)+1;
    else returnarg= length-startIndex;
    return returnarg;
}

/**
 * Get the longest string from an array of strings
 * @param generalmenu stringarray where we want to find the longest string
 * @param finnAntElementer Number of elements in the string
 * @return returns the length of the longest string
 */
public int getLengthLongest(String[] generalmenu, int finnAntElementer){
    int longest=0;
    for(int k=0; k<finnAntElementer; k++){
        if (generalmenu[k].length()>longest){
            longest=generalmenu[k].length();
        }
    }
    return longest;
}

/**
 * Gets the dates for the next 14 days. To be displayed in a pop-up when the
 * user chooses date for an event
 */
private String[] getNextDates() {
    int nrofdays=14;
    String[] calendarreturn= new String[nrofdays];
    Calendar cal = Calendar.getInstance();
    Date date = cal.getTime();
    cal.setTime(date);
    String outDate;
    long time = date.getTime();
    long _1day= 24L * 60L * 60L * 1000L;
    calendarreturn[0]="Heute";
```

```
    for(int m=1; m<nrofdays; m++){
        time += _1day;
        date = new Date(time);
        calendarreturn[m]=date.toString().substring(0, date.toString().
                lastIndexOf(' ', 14));
    }
    return calendarreturn;
}



/**
 * When a search has been initiated, this methode is called when a result
 * has been received from the Net.
 */
public void searchDataAvailableEvent() {
    if(currentCanvasPopUpDisplay!=POP_UP_NO_RESULTS_FOUND){
        level=4;
        if(currentCanvasDisplay!=MULTIPLE_SEARCH_RESULT){
            currentCanvasDisplay=SEARCH_RESULT;
        }
        if(ApplicationData.searchTheme.compareTo("placeSearch")==0){
            startDisplayPlaceResults(ApplicationData.placeSearchResult);
        }
        else{
            startDisplayCityEventResults(
                    ApplicationData.cityEventsSearchReslut);
        }
        currentCanvasPopUpDisplay=NO_POP_UP;
        repaint();
    }
}

/**
 * When a search has been initiated, this methode is called when an empty
 * result has been received from the Net.
 */
public void searchResultEmptyEvent() {
    if(currentCanvasPopUpDisplay!=POP_UP_NO_RESULTS_FOUND){
        currentCanvasPopUpDisplay=POP_UP_NO_RESULTS_FOUND;
        repaint();
    }
}
```

```
/**
 * Sets an ApplicationData object
 * @param applicationData application data object
 */
public void setApplicationData(ApplicationData applicationData){
    this.applicationData = applicationData;
}


/**
 * A methode for displaying search results when using the offline
 * functionality
 */
public void performSearch(){
    SearchResult[] searchResultLevel4 = searchService.performSearch("Essen",
            " "," ",Calendar.getInstance().getTime());
    level=4;
    currentCanvasDisplay=SEARCH_RESULT;
    startDisplaySearchResultsalt(searchResultLevel4);
}

/**
 * Sets a place search element in mobcanvas. Used when starting a place
 * search
 * @param performPlaceSearchEvent Place search element
 */
public void setperformPlaceSearchEvent(Event performPlaceSearchEvent) {
    this.performPlaceSearchEvent=performPlaceSearchEvent;
}
/**
 * Sets a city event search element in mobcanvas. Used when starting a city
 * event search
 * @param performCityEventSearchEvent city event search element
 */
public void setperformCityEventSearchEvent(
        Event performCityEventSearchEvent){
    this.performCityEventSearchEvent=performCityEventSearchEvent;
}

/**
 * Sets a restore last search event object in mobcanvas.
 * @param restoreLastSearchData the event last search object
 */
public void setrestoreLastSearchData(Event restoreLastSearchData) {
    this.restoreLastSearchData=restoreLastSearchData;
```

```
    }

//XXXXXXXXXXXXXXXXXXXXXXXXXXXX   GENERAL METHODES END  XXXXXXXXXXXXXXXXXXXXXXXX


//XXXXXXXXXXXXXXXXXXXXXXXXXXXX   VARIABLES   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

    public final static int ERICSSON=0;
    public final static int NETBEANS=1;
    public final static int NOKIA=2;

    public final static int THEME_ESSEN = 1;
    public final static int THEME_PLANEN = 2;
    public final static int THEME_TRINKEN = 3;
    public final static int THEME_SEHEN = 4;
    public final static int THEME_TANZEN = 5;
    public final static int THEME_HOREN = 6;

    public final static int MAIN_MENU = 0;
    public final static int SEARCH_PAGE = 1;
    public final static int SEARCH_RESULT = 2;
    public final static int MULTIPLE_SEARCH_RESULT=3;

    public final static int NO_POP_UP = 0;
    public final static int POP_UP = 1;
    public final static int POP_UP_ELEMENT_INFO = 2;
    public final static int POP_UP_SEARCHING = 3;
    public final static int POP_UP_NO_RESULTS_FOUND = 4;

    public final static int NO_POP_UP2=0;
    public final static int POP_UP2_LOADING =1;

    public final static int TYPE_CATEGORY = 1;
    public final static int TYPE_CITY_DISTRICT  = 2;
    public final static int TYPE_DATE = 3;
    public final static int TYPE_FILM = 4;
    public final static int TYPE_CINEMA = 5;

    private int currentSearchDisplay, currentCanvasDisplay=0,
            currentCanvasPopUpDisplay=0, currentCanvasPopUpDisplay2=0;
    private int currentPopUpType;
```

```
/*
 * Level 1
 */
int x=1, y=1;
Image header, bildeFilm, bildeGo, bildeTanzen, bildeMap, bildeFavoritt,
        bildeHoren, bildeRest, bildeSuche, bildeTrinken;
Image bildeFilm2, bildeGo2, bildeFavoritt2, bildeMap2, bildeTanzen2,
        bildeHoren2, bildeRest2, bildeSuche2, bildeTrinken2;
Image tab1, tab2, tab3, tab4, tab5, star, star2, vorlage;
int antHoyde=3, antBredde=3;
int widthSpacebetweenIcons, hightSpacebetweenIcons, kolonne1, kolonne2,
        kolonne3, rekke1, rekke2, rekke3;
public static int scrW, scrH, bildefilmW, bildefilmH, bildefilm2W,
        bildefilm2H, corrH, corrW;

/*
 * Level 2
 */
int z=1, antz;
int sokestrenghoyde, sokeknapphoyde, buttonramme, textdiffx, textdiffy;

/*
 * Level 3
 */
int t=0;
String[] generalmenu;
int longestLength, top, bottom;
int setMaxPopupElementerLevel3, antElementerLevel3and4;
boolean usetoday=true;

/*
 *Level 4
 */
int searchResultsGapTop=10, setMaxPopupElementerLevel4;

MyCityGuideSearchService searchService = new MyCityGuideSearchService();

Place[]  placeResultLevel4;
Place placeResultElementForLevel5;
String[] placeResultNamesLevel4;

CityEvent[] cityEventResultLevel4;
CityEvent cityEventElementForLevel5;
String[] cityEventResultNamesLevel4;
```

```
SearchResult[]  searchResultLevel4;
SearchResult searchResultElementForLevel5;
String[] searchResultNamesLevel4;

int savedt, savedtop, savedbottom;
int u=1;

/*
 * Level 5
 */
int w=1, v=1;
int yHight, textgap= 3;
Image was, warum;
int elementPopUpGapSide=0, elementPopUpGapTop=18;
int setMaxPopupElementerLevel5, antElementerLevel5;
String descriptionString=null;
String[] descriptionStringArray;

/*
 * General
 */
private GuiComponentImpl guiComponentImpl;
private ApplicationData applicationData;
private Event performPlaceSearchEvent;
private Event performCityEventSearchEvent;
private Event restoreLastSearchData;

public final static int softkey =-6;
int softkeyspace;
int mobtype;
String searchInput;
String catagoryHeadline;
int level=1;

Image button, button2;
int themeID, type;
int hoydeinfoBar, infoBarInfoH=5, infoBarInfoW=19;
boolean uparrow, downarrow, savetoparrow, savebottomarrow;
boolean doSearch=true;
String midlCityDistrictEssen, midlEssenCategory, midlDistrictTrinken,
        midlTrinkenCategory, midlEventType, midlCinema,
        midlCityDistrictTanzen, midlDate;
Font headline =
        Font.getFont(Font.FACE_SYSTEM, Font.STYLE_BOLD, Font.SIZE_LARGE);
Font medium =
```

```
            Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_MEDIUM);
    Font boldmedium =
            Font.getFont(Font.FACE_SYSTEM, Font.STYLE_BOLD, Font.SIZE_MEDIUM);
    Font boldmediumunderlined =
            Font.getFont(Font.FACE_SYSTEM, Font.STYLE_BOLD |
            Font.STYLE_UNDERLINED, Font.SIZE_MEDIUM);
    Font mediumunderlined =
            Font.getFont(Font.FACE_SYSTEM, Font.STYLE_UNDERLINED,
            Font.SIZE_MEDIUM);
    Font small =
            Font.getFont(Font.FACE_SYSTEM, Font.STYLE_PLAIN, Font.SIZE_SMALL);
    Font boldsmall =
            Font.getFont(Font.FACE_SYSTEM, Font.STYLE_BOLD, Font.SIZE_SMALL);
}
```

CHAPTER 2

ApplicationData.java

```
/*
 * ApplicationData.java
 *
 * Created on 28. März 2007, 11:43
 *
 */

package de.unternehmertum.mobile.mycityguide;

import de.unternehmertum.mobile.playtools.MainMidlet;
import de.unternehmertum.mobile.playtools.event.Event;
import de.unternehmertum.mobile.playtools.gui.GuiCmdListener;
import de.unternehmertum.mobile.playtools.gui.GuiCommand;
import de.unternehmertum.mobile.playtools.gui.GuiCommandResult;
import de.unternehmertum.mobile.playtools.gui.GuiComponentImpl;
import de.unternehmertum.mobile.playtools.util.jsr172.rmi.RemoteException;
import de.unternehmertum.playbox.remoting.Cinema;
import de.unternehmertum.playbox.remoting.CityEvent;
import de.unternehmertum.playbox.remoting.Movie;
import de.unternehmertum.playbox.remoting.Place;
import de.unternehmertum.playbox.remoting.Position;
import de.unternehmertum.playbox.remoting.ServiceException;
import de.unternehmertum.mobile.playtools.logging.Log;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Item;
```

```
import javax.microedition.rms.RecordStore;
import javax.microedition.rms.RecordStoreException;
import javax.microedition.rms.RecordStoreNotFoundException;

/**
 *
 * This class is responsible for the storing the data that the application
 needs. It uses
 * the RecordStore class of J2ME giving the application persistent storage
 possibilities.
 * The class also serves as a logical unit that coordinates the different
 classes, also called
 * a controller in the MVC design pattern.
 * It also ensures server communication with the CityMob Server: for security
 aspects, the
 * ApplicationData class uses the SecurityCmdListener; for service requests it
 uses the
 * MyCityGuideService (implemented on the client, not to be mistaken with the
 * MyCityGuideService implemented on the server).
 *
 * @author lefebure
 */
public class ApplicationData extends GuiCmdListener{

    /** Creates a new instance of ApplicationData */
    public ApplicationData() {
    }


    /**
     * First run initialization, app start initialization and store data
     methods. Initializes the application
     * data when it is first started. Stores the new data to persistence
     memory when the application is closed.
     * This in order to reduce te number of write and read operations on
     RecordStores.
     **/

    private void firstRunInitialization(){
        try {

            // Create and initialize defaultVariables_RS
            rs = RecordStore.openRecordStore("defaultVariables_RS", true);
            byte[] _data = "Alle".getBytes();
            for (int i=0; i< 7; i++)rs.addRecord(_data, 0, _data.length);
```

```
            _data = "Heute".getBytes();
            rs.addRecord(_data, 0, _data.length);
            _data = "undiscovered".getBytes();
            rs.addRecord(_data, 0, _data.length);
            rs.closeRecordStore();

            // Create lastSearches_RS
            rs = RecordStore.openRecordStore("lastSearches_RS", true);
            rs.closeRecordStore();

            // Create lastSearches_RS
            rs = RecordStore.openRecordStore("favorites_RS", true);
            rs.closeRecordStore();

        } catch (RecordStoreException ex) {
            ex.printStackTrace();
        }
    }
    /**
     * This method initializes all the data that the application needs. This
     data is read from a record store, ensuiring
     * persistence data storage. This method also initializes the MobCanvas
     class, as well as it initiates the
     *  GPS discovery process.
     **/
    public void initializeApplicationData(){

        byte[] _data;

        //Load the default data
        try {
            rs = RecordStore.openRecordStore("defaultVariables_RS", false);
            _data = rs.getRecord(1);
            defaultCityDistrictEssen = new String(_data,0,_data.length);
            _data = rs.getRecord(2);
            defaultEssenCategory = new String(_data,0,_data.length);
            _data = rs.getRecord(3);
            defaultcityDistrictTrinken = new String(_data,0,_data.length);
            _data = rs.getRecord(4);
            defaultTrinkenCategory = new String(_data,0,_data.length);
            _data = rs.getRecord(5);
            defaultEventType = new String(_data,0,_data.length);
            _data = rs.getRecord(6);
            defaultCinema = new String(_data,0,_data.length);
            _data = rs.getRecord(7);
```

```
            defaultCityDistrictTanzen = new String(_data,0,_data.length);
            _data = rs.getRecord(8);
            defaultDate = new String(_data,0,_data.length);
            _data = rs.getRecord(9);
            bluetoothDeviceUrl = new String(_data,0,_data.length);
            rs.closeRecordStore();

            rs = RecordStore.openRecordStore("lastSearches_RS", false);
            numbOfStoredSearches = rs.getNumRecords();
            rs.closeRecordStore();

        } catch (RecordStoreException ex) {
            // In case the record stores have been deleted by the user or if
            it is the first time the user runs the application
            firstRunInitialization();
        }
        //Controller tasks
        mobCanvas.initialize();
        if(USE_GPS)initializeBluetoothAndGPS();
        else this.guiComponentImpl.setActiveDisplayable(mobCanvas);
        if(DEBUG_TEST)performServiceTest();
    }

    /**
     * This method must be called before the application is closed, otherwise
     the default data may get lost.
     * It stores the variable data to the persistent storage. Uses a record
     store.
     **/
    public void storeApplicationData(){
        try {
            rs = RecordStore.openRecordStore("defaultVariables_RS", true);
            byte[] _data = defaultCityDistrictEssen.getBytes();
            rs.setRecord(1, _data, 0, _data.length);
            _data = defaultEssenCategory.getBytes();
            rs.setRecord(2, _data, 0, _data.length);
            _data = defaultcityDistrictTrinken.getBytes();
            rs.setRecord(3, _data, 0, _data.length);
            _data = defaultTrinkenCategory.getBytes();
            rs.setRecord(4, _data, 0, _data.length);
            _data = defaultEventType.getBytes();
            rs.setRecord(5, _data, 0, _data.length);
            _data = defaultCinema.getBytes();
            rs.setRecord(6, _data, 0, _data.length);
            _data = defaultCityDistrictTanzen.getBytes();
```

```
            rs.setRecord(7, _data, 0, _data.length);
            _data = defaultDate.getBytes();
            rs.setRecord(8, _data, 0, _data.length);
            _data = bluetoothDeviceUrl.getBytes();
            rs.setRecord(9, _data, 0, _data.length);
            rs.closeRecordStore();
        } catch (RecordStoreException ex) {
            ex.printStackTrace();
        }
    }

    /**
     * Method to display the last search. This method actually performs a new
     search after having retrieved the
     *  search parameters from the record store. lastSearchNumber goes from 1
     to unlimited, the smaller the number,
     * the closer in time.
     **/

    public void restoreLastSearchData(){
        //verifies the input number...
        if(numbOfStoredSearches==0){
            searchResultEmptyEvent.fireEvent();
            return;
        }
        // in case the number of the last search is too high .. does not
        exist...
        if(numbOfStoredSearches < lastSearchNumber|| lastSearchNumber<0){
            searchResultEmptyEvent.fireEvent();
            return;
        }

        // Retrieve from lastSearches_RS
        String[] lastSearchData = retrieveLastSearchData(lastSearchNumber);
        if(lastSearchData==null){searchResultEmptyEvent.fireEvent();return;}

        // Performs the new search, according to the search data
        if(lastSearchData[0].compareTo("placeSearch")==0){
            searchTheme="placeSearch";
            searchType=lastSearchData[1];
            searchCityDistrict=lastSearchData[2];
            searchCategory=lastSearchData[3];
            storeSearch=false;
            performPlaceSearch();
            storeSearch=true;
```

```
    } else if(lastSearchData[0].compareTo("eventSearch")==0){
        searchTheme="eventSearch";
        searchType=lastSearchData[1];
        searchDate=lastSearchData[2];
        storeSearch=false;
        performCityEventSearch();
        storeSearch=true;
    }
}

/**
 * Stores the parameters for the last search as a string.
 * example string: "placeSearch##essen##amerikanisch##"scwabing"
 *
 **/

private void storeLastSearch(String searchData) {
    try {
        rs = RecordStore.openRecordStore("lastSearches_RS", false);
        byte[] _data = searchData.getBytes();
        rs.addRecord(_data, 0, _data.length);
        rs.closeRecordStore();
        numbOfStoredSearches++;
    } catch (RecordStoreException ex) {
        ex.printStackTrace();
    }
}
/*
 * Help method for retrieving the last search data.
 **/
private String[] retrieveLastSearchData(int lastSearchNumber) {

    String[] lastSearchData=null;

    try {

        rs = RecordStore.openRecordStore("lastSearches_RS", false);
        byte[] _data = rs.getRecord(numbOfStoredSearches -
        lastSearchNumber +1);
        rs.closeRecordStore();
        String searchString = new String(_data,0,_data.length);
        lastSearchData = split(searchString);
    } catch (RecordStoreException ex) {
        ex.printStackTrace();
        return null;
```

```
    }
    return lastSearchData;
}

/*
 * This functionality was never implemented.
 **/
public void setNewFavorite(String newFavorite) {
}

public String getUsername(){return username;}
public String getPassword(){return password;}


/**
 * CONTROLLER METHODS. Methods that serve the purpose of a controller in
 an MVC design.
 **/

 /*
 * Method required by the play.Tools framework.
 **/
public void setGuiComponent(GuiComponentImpl guiComponentImpl) {
    this.guiComponentImpl = guiComponentImpl;
}

 /*
 * Method required by the play.Tools framework. Takes input from the
 keypad of the mobile phone, in order
 * to deal accordingly.
 **/
public int cmdAction(GuiCommand guiCommand, Displayable displayable, Item
item) {
     if (guiCommand.getName().equalsIgnoreCase("closeMap")){
         this.guiComponentImpl.getcommand_closeMap().addCommandResult(new
         GuiCommandResult(0, mobCanvas, "Close map"  ));
         this.guiComponentImpl.setActiveDisplayable(mobCanvas);
         mobCanvas.loadingPopUpOff();
     }
     if (guiCommand.getName().equalsIgnoreCase("zoomIn")){
       mapCanva.zoomIn();
    }
     if (guiCommand.getName().equalsIgnoreCase("zoomOut")){
         mapCanva.zoomOut();
    }
```

```
    return 0;
}

 /*
 * Gives a referance to this class to the MapCanva class.
 **/
public void setMapCanva(Map2CmdListener mapCanva){
    this.mapCanva=mapCanva;
}

 /*
 * Gives a referance to this class to the MobCanvas class.
 **/
public void setmobCanvas(MobCanvas mobCanvas){
    this.mobCanvas=mobCanvas;
}

 /*
 * Gives a referance to this class to the BluetoothCmdListener class.
 **/
public void setBluetoothListener(BluetoothCmdListener bluetoothListener){
    this.bluetoothListener=bluetoothListener;
}


/**
 * Web Service search methods
 *
 * Methods called when a search is to be performed. When the search is
 finished and the new data
 * is loaded into ApplicationData class, it sends an event to its
 listeners so that they can update their info.
 **/

/**
 * search method for place searches.
 **/
public void performPlaceSearch(){

    //performServiceTest();
    verifySearchSettings();
    printSearchData();
    searchTheme= "placeSearch";
    searchType = searchType.toLowerCase();
    searchCityDistrict = searchCityDistrict.toLowerCase();
```

```
    searchCategory = searchCategory.toLowerCase();


    try {
        placeSearchResult =
        restaurantService.service.placeSearch(searchType,
        searchCityDistrict, searchCategory).getPlace();

        if(!cancelSearch){


         if(storeSearch)storeLastSearch("4#"+searchTheme+"#"+searchType+"#"
         +searchCityDistrict+"#"+searchCategory);
            storeApplicationData();
            searchDataAvailableEvent.fireEvent();
        }
        cancelSearch=false;

    } catch (RemoteException ex) {
        //ex.printStackTrace();
        searchResultEmptyEvent.fireEvent();
    } catch (ServiceException ex) {
        searchResultEmptyEvent.fireEvent();
        //ex.printStackTrace();
    }
}

/**
 * search method for event searches.
 **/
public void performCityEventSearch(){
    verifySearchSettings();

    searchTheme = "eventSearch";
    searchType = searchType.toLowerCase();
    searchDate = searchDate.toLowerCase();

    try {
        cityEventsSearchReslut =
        restaurantService.service.searchEvents(searchType,
        searchDate).getCityEvent();

        if(!cancelSearch){
```

```
            if(storeSearch)storeLastSearch("3#"+searchTheme+"#"+searchType+"#"
               +searchDate);storeApplicationData();
               searchDataAvailableEvent.fireEvent();
            }
            cancelSearch=false;

        } catch (RemoteException ex) {
            //ex.printStackTrace();
            searchResultEmptyEvent.fireEvent();
        } catch (ServiceException ex) {
            searchResultEmptyEvent.fireEvent();
            //ex.printStackTrace();
        }
    }

     /**
     * Cancels an ongoing search
     **/
    public void cancelSearch(){
        cancelSearch=true;
    }

    /**
     * Verifies the input settings for a search
     **/
    private void verifySearchSettings(){
        if(searchType.compareTo("Alle")==0)searchType="%";
        if(searchCityDistrict.compareTo("Alle")==0)searchCityDistrict="%";
        if(searchCategory.compareTo("Alle")==0)searchCategory="%";
        if(searchDate.compareTo("Heute")!=0 &&
        searchDate.compareTo("heute")!=0)searchDate="%";
    }

    public void setRestaurantService(RestaurantService restaurantService){
        this.restaurantService=restaurantService;
    }

    public void setsearchDataAvailableEvent(Event searchDataAvailableEvent) {
        this.searchDataAvailableEvent=searchDataAvailableEvent;
    }
    public void setsearchResultEmptyEvent(Event searchResultEmptyEvent) {
        this.searchResultEmptyEvent=searchResultEmptyEvent;
    }
```

```
/**
 * Login methods
 *
 **/
public void setperformLoggInEvent(Event performLoggInEvent) {
    this.performLoggInEvent=performLoggInEvent;
}

/**
 * Method called when the user is logged in
 **/
public void loggedInEvent() {
    Log.println(this, "Logged In succesfull");
    isLoggedIn=true;
    if(displayMap){
        displayMap=false;
        displayMap();
    }
}

/**
 * Method called when the logg in failed
 **/
public void loggedInFailedEvent() {
    Log.println(this, "Loggin failed");
    isLoggedIn=false;
}

/**
 * GPS and Bluetooth methods
 *
 **/

public void initializeBluetoothAndGPS(){
    if(!USE_GPS)return;
    if(bluetoothDeviceUrl.compareTo("undiscovered")==0){

        this.guiComponentImpl.setActiveDisplayable(this.guiComponentImpl.
        getdisplayable_bluetoothForm());
        bluetoothListener.doDeviceDiscovery();
    }else {
        this.guiComponentImpl.setActiveDisplayable(mobCanvas);
        bluetoothListener.connectGPS(bluetoothDeviceUrl);
    }
}
```

```
public void positionAvailableEvent() {
    Log.println(this, "GPS position available");
    s("New position: "+devicePosition);
    //storeApplicationData();
}

// Called when the connection to the GPS failed.
public void gpsConnectFailedEvent() {
    Log.println(this, "GPS connection failed");
    //bluetoothDeviceUrl="undiscovered";
    //storeApplicationData();
    initializeBluetoothAndGPS();
}

/**
 * Map methods
 **/

public void showMapWithPOI(Double latitude, Double longitude){
    defaultGPSPosition = new Position(latitude,longitude);
    displayMap();
    mapCanva.addPOI(defaultGPSPosition,"");
}
public void showMapWithDevicePosition(){
     defaultGPSPosition = devicePosition;
     displayMap();
}
private void displayMap(){
    if(isLoggedIn){
        mobCanvas.loadingPopUpOff();
        mapCanva.showMap();
    }else{
        displayMap=true;
        performLoggInEvent.fireEvent();
    }
}

/**
 * Method called to dial a number directly from the application
 **/
public void callNumber(String number) {
    boolean b;
    if(number.compareTo("")==0)number="004915771961408";
    try{
```

```
        MainMidlet.getMidlet().pauseApp();
        MainMidlet.getMidlet().notifyPaused();
        b = MainMidlet.getMidlet().platformRequest("tel:"+number);
        MainMidlet.getMidlet().resumeRequest();
    }
    catch(Exception e){
        s(""+e);
    }
}

/**
 * Method called to display a wap page in the default browser ofthe mobile
 phone
 **/
 public void startMvvService() {
    boolean b;
    try{
        MainMidlet.getMidlet().pauseApp();
        MainMidlet.getMidlet().notifyPaused();
        b =
        MainMidlet.getMidlet().
        platformRequest("http://efa.mvv-muenchen.de/mobile/index_de.html");
        MainMidlet.getMidlet().resumeRequest();
    }

    catch(Exception e){
        s(""+e);
    }
}

/**
 * END CONTROLLER METHODS
 */


/**
 *  Splits a search string. The search string must be uder the form:
 *  "4#placeSearch#"+searchType+"#"+searchCityDistrict+"#"+searchCategory"
 *
 **/
private String [] split(String string) {

    // First get rid of whitespace at start and end of the string
    // String string = source_string.trim();
    // If string contains no tokens, return a zero length array.
```

```
    if( string.length() == 0) return (new String [0]);
    s(string);
    // Use a Vector to collect the unknown number of tokens.
    String token="";
    String[] resultString = null;
    int index_a = 0;
    int index_b = 0;

    // Finds the number of elemts to find, hence the size of the return
    array.

    index_b = string.indexOf('#', index_a);
    token = string.substring(index_a, index_b);
    resultString = new String[Integer.valueOf(token).intValue()];
    index_a = index_b + 1;
    int t=0;
    while(true){
        index_b = string.indexOf('#', index_a);
        if (index_b == -1) {
            token = string.substring(index_a);
            resultString[t] = token;
            break;
        }
        token = string.substring(index_a, index_b);
        resultString[t] = token;
        index_a = index_b + 1;
        t++;
    }
    return resultString;
} // split


/**
 * Variables
 **/

private final String username="al";
private final String password="al";


/**
 * Controller variables
 **/
private GuiComponentImpl guiComponentImpl;
private Event searchDataAvailableEvent;
```

```
private Event searchResultEmptyEvent;
private Event performLoggInEvent;

private RestaurantService restaurantService;

private Map2CmdListener mapCanva;
private MobCanvas mobCanvas;
private BluetoothCmdListener bluetoothListener;
/**
 * Static variables that are set for a specific city.
 **/

// Category specific data
public static String[] cityDistrictsEssen =
{"Alle","Allach-Untermenzing","Altstadt-Lehel","Au-Haidhausen","Berg am
Laim","Bogenhausen","Freimann","Giesing-Harlaching","Hadern","Ludwigsvorstadt-I
sarvorstadt","Milbertshofen","Moosach","Neuhausen-Nymphenburg","Obersendling-Th
alkirchen-Solln","Pasing-Obermenzing","Ramersdorf-Perlach","Riem","Schwabing-Ma
xvorstadt","Schwanthalerhöhe-Laim","Sendling-Westpark","Trudering"};
public static String[] essenCategories =
{"Alle","Amerikanisch","Asiatisch","Bayerisch","Chinesisch","Edel /
Gourmet","Fisch / Seafood","Französisch","Frühstück /
Brunch","Griechisch","Indisch","International","Italienisch","Japanisch /
Sushi","Mediterran","Nachtlokal","Orientalisch","Portugisisch","sonstiges","Spa
nisch","Tex-Mex","Thailändisch","Türkisch","Vegetarisch","Vietnamesisch"};
public static String[] cityDistrictsTrinken
={"Alle","Altstadt-Lehel","Au-Haidhausen","Berg am
Laim","Bogenhausen","Giesing-Harlaching","Hadern","Ludwigsvorstadt-Isarvorstadt
","Milbertshofen","Moosach","Neuhausen-Nymphenburg","Obersendling-Thalkirchen-S
olln","Ramersdorf-Perlach","Schwabing-Maxvorstadt","Schwanthalerhöhe-Laim","Sen
dling-Westpark"};
public static String[] trinkenCategories =
{"Alle","Biergarten","Cafe","Cocktailbar","Edel","Kneipe","Live-Club","Lounge",
"Pub","Sportsbar","Tagesbar"};
public static String[] eventTypes =
{"Alle","Partys","Konzerte","Comedy","Theater","Klassik","Kunst","Sport","Messe
"};


/**
 *  Default variables
 *
 * RecordStore used as persistence storage.
 * Following recordStores are used:
 *      defaultVariables_RS
```

```
 *      lastSearches_RS
 *      favorites_RS
 **/

private RecordStore rs;

public static String defaultCityDistrictEssen="Alle";
//defaultVariables_RS record_id = 1
public static String defaultEssenCategory="Alle"; //defaultVariables_RS
record_id = 2
public static String defaultcityDistrictTrinken="Alle";
//defaultVariables_RS record_id = 3
public static String defaultTrinkenCategory="Alle"; //defaultVariables_RS
record_id = 4
public static String defaultEventType="Alle"; //defaultVariables_RS
record_id = 5
public static String defaultCinema="Alle"; //defaultVariables_RS record_id
= 6
public static String defaultCityDistrictTanzen="Alle";
//defaultVariables_RS record_id = 7
public static String defaultDate="Heute"; //defaultVariables_RS record_id
= 8
public static String bluetoothDeviceUrl="undiscovered";
//defaultVariables_RS record_id=9

private int numbOfStoredSearches=0; //number of last searches stored

public static String searchTheme="";
public static String searchType="%";
public static String searchCityDistrict="%";
public static String searchCategory="%";
public static String searchDate="%";

public static int lastSearchNumber=0;
private boolean storeSearch=true;
private boolean cancelSearch=false;

public static Position devicePosition = new Position(new
Double(48.161900),new Double(11.585800));
public static Position defaultGPSPosition  = new Position(new
Double(48.161900),new Double(11.585800));

// Map display variables
private boolean isLoggedIn=false;
private boolean displayMap=false;
```

```
public static CityEvent[] cityEventsSearchReslut;
public static Place[] placeSearchResult;

public static boolean DEBUG_TEST = false;
public static boolean DEBUG_PRINT = false;
public static boolean USE_GPS = false;
public static boolean OFFLINE = false;

/**
 * Debug and Web Service Test Methods.
 *
 **/

private void performApplicationLifeTest(){
    s("##### DEBUG Verify default data");
    s("defaultCityDistrictEssen:    "+defaultCityDistrictEssen);
    s("defaultEssenCategory:        "+defaultEssenCategory);
    s("defaultcityDistrictTrinken:  "+defaultcityDistrictTrinken);
    s("defaultTrinkenCategory:      "+defaultTrinkenCategory);
    s("defaultPlanenCategory:       "+defaultEventType);
    s("defaultCinema:               "+defaultCinema);
    s("defaultCityDistrictTanzen:   "+defaultCityDistrictTanzen);
    s("defaultDate:                 "+defaultDate);
    s("bluetoothDeviceUrl:          "+bluetoothDeviceUrl);
    s("numbStoredLastSearch:        "+numbOfStoredSearches);
    s("##### DEBUG end");
}

private void printSearchData(){
    s("##### DEBUG printSearchData");
    s("numbStoredLastSearch: "+numbOfStoredSearches);
    s("lastSearchNumber: "+lastSearchNumber);
    s("");
    s("searchTheme: "+searchTheme);
    s("searchType: "+searchType);
    s("searchCityDistrict: "+searchCityDistrict);
    s("searchCategory: "+searchCategory);
    s("searchDate: "+searchDate);
    s("##### DEBUG end");

}

private void printAllLastSearches(){
    s("#### DEBUG printAllLastSearches ####");
```

```
        try {
            rs = RecordStore.openRecordStore("lastSearches_RS", false);
            for (int i=1; i<rs.getSize()-1; i++){
                byte[] _data = rs.getRecord(i);
                String searchString = new String(_data,0,_data.length);
                s("Element "+i+" "+searchString);
            }
            rs.closeRecordStore();
        } catch (RecordStoreException ex) {
            ex.printStackTrace();
        }

        s("#### DEBUG printAllLastSearches END ####");
    }

    private void performServiceTest(){
        testCityEvent();
        testListAllCinemas();
        testListAllMovies();
        testPlace();
    }

    private void testCityEvent(){
        try {
            s(" ##### WS Query test #####");
            s(" ##### Service searchEvents()  #####");
            CityEvent[] events =
            restaurantService.service.searchEvents("%","%").getCityEvent();
            for(int i=0; i<events.length; i++){
                s("New Event: "+events[i].getName());
                s("Event date: "+events[i].getOpening_data());
            }


        } catch (RemoteException ex) {
            ex.printStackTrace();
        } catch (ServiceException ex) {
            ex.printStackTrace();
        }
        s(" ##### Query test END #####");
    }
    private void testPlace(){
        try {
            s(" ##### WS Query test #####");
            s(" ##### Service placeSearch() #####");
```

```
        Place[] places =
        restaurantService.service.placeSearch("%","%","%").getPlace();
        for(int i=0; i<places.length; i++)System.out.println("New Place:
        "+places[i].getName());
    } catch (RemoteException ex) {
        ex.printStackTrace();
    } catch (ServiceException ex) {
        ex.printStackTrace();
    }

    s(" ##### Query test END #####");
}
private void testListAllMovies(){
    try {
        s(" ##### WS Query test #####");
        s(" ##### Service ListAllMovies()  #####");
        Movie[] movies =
        restaurantService.service.listAllMovies().getMovie();

        for(int i=0; i<movies.length; i++){
            s("New Movie: "+movies[i].getTitle());
            s("Movie Poll: "+movies[i].getPoll().getValue());
        }
    } catch (RemoteException ex) {
        ex.printStackTrace();
    } catch (ServiceException ex) {
        ex.printStackTrace();
    }
    s(" ##### Query test END #####");
}
private void testListAllCinemas(){
    try {
        s(" ##### WS Query test #####");
        s(" ##### Service ListAllCinemas()  #####");
        Cinema[] cinemas =
        restaurantService.service.listAllCinemas("%").getCinema();
        for(int i=0; i<cinemas.length; i++)System.out.println("New Cinema:
        "+cinemas[i].getPlace().getName());
    } catch (RemoteException ex) {
        ex.printStackTrace();
    } catch (ServiceException ex) {
        ex.printStackTrace();
    }
    s(" ##### Query test END #####");
}
```

```
    private void s(String
    string){if(DEBUG_PRINT)System.out.println(""+string);}

}
```

CHAPTER 3

BluetoothCmdListener.java

```
/*
 * BluetoothCmdListener.java

 * This class is responsible for searching bluetooth devices that are in the
 surroundings of the device. When all the devices
 * are found, these are displayed as a list. The user is then able to select
 the device that corersponds to the GPS.
 * When the item is selected, a connection to the GPS is made.
 *
 * Created on 28. März 2007, 17:08
 * @author lefebure
 *
 */

package de.unternehmertum.mobile.mycityguide;

import de.unternehmertum.mobile.playtools.bluetooth.BluetoothComponent;
import de.unternehmertum.mobile.playtools.bluetooth.BluetoothDevice;
import de.unternehmertum.mobile.playtools.bluetooth.BluetoothListener;
import de.unternehmertum.mobile.playtools.event.Event;
import de.unternehmertum.mobile.playtools.gps.GpsComponent;
import de.unternehmertum.mobile.playtools.gui.GuiCmdListener;
import de.unternehmertum.mobile.playtools.gui.GuiCommand;
import de.unternehmertum.mobile.playtools.gui.GuiCommandResult;
import de.unternehmertum.mobile.playtools.gui.GuiComponentImpl;
```

```
import de.unternehmertum.mobile.playtools.logging.Log;
import java.io.InputStream;
import java.util.Vector;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Item;

public class BluetoothCmdListener extends GuiCmdListener implements
BluetoothListener{

    String bluetoothDeviceUrl;

    private BluetoothComponent bluetooth;

    private GuiComponentImpl guiComponentImpl;

    private GpsComponent gpsComponent;

    private Event gpsConnectFailedEvent;

    private MobCanvas mobCanvas;

    private int blueTSearchAttemps=3;

    private boolean cancelBT=false;

    /** Creates a new instance of BluetoothCmdListener */
    public BluetoothCmdListener() {

    }

    public int cmdAction(GuiCommand c, Displayable d, Item item) {



        if (c.getName().equalsIgnoreCase("bluetoothDiscoverDevices")){
            this.guiComponentImpl.getitem_bluetoothForm_blueTMessage1().setLabel("
            Searchin for a GPS device");

            wait.. ");
            doDeviceDiscovery();
        }else if (c.getName().equalsIgnoreCase("connectBTdevice")){
            int selectedDevice =
```

```
        this.guiComponentImpl.getitem_bluetoothForm_bluetoothDevicelist().
        getSelectedIndex();
        BluetoothDevice btd =
        (BluetoothDevice)bluetooth.getBluetoothDevices().elementAt(selectedDevice);
        bluetoothDeviceUrl = btd.getBtsppURL();
        ApplicationData.bluetoothDeviceUrl=bluetoothDeviceUrl;
        connectGPS(bluetoothDeviceUrl);

    }else if (c.getName().equalsIgnoreCase("skipGPS")){
        cancelBT=true;
        bluetooth.cancelDiscover();
        ApplicationData.USE_GPS=false;
        this.guiComponentImpl.getcommand_skipGPS().addCommandResult(new
        GuiCommandResult(0, mobCanvas, "Skip GPS discovery"  ));
    }
    return 0;
}

public void setBluetoothComponent(BluetoothComponent bluetooth) {
    this.bluetooth = bluetooth;
}

public void notifyBluetoothListener(int code, Object o) {
    if (code == BluetoothComponent.DISCOVERY_COMPLETE){
        int complete = ((Integer)o).intValue();
        deviceSearchComplete(complete);
    }else if (code == BluetoothComponent.DISCOVERY_CANCELED){
        this.guiComponentImpl.setActiveDisplayable(mobCanvas);
    }
}

private void deviceSearchComplete(int complete){
    if(cancelBT){cancelBT=false;return;}
    this.guiComponentImpl.getitem_bluetoothForm_blueTMessage1().setLabel("
    Bluetooth GPS device search complete");
    this.guiComponentImpl.getitem_bluetoothForm_blueTMessage2().setLabel("Please
    select your GPS device: ");

    ChoiceGroup g =
    this.guiComponentImpl.getitem_bluetoothForm_bluetoothDevicelist();
    g.deleteAll();

    if(bluetooth.getBluetoothDevices().size()<1){
        g.append("no devices",null);
```

```
        } else {
            for (int i = 0; i < bluetooth.getBluetoothDevices().size();i++){
                Vector v = bluetooth.getBluetoothDevices();
                BluetoothDevice bluetooth = (BluetoothDevice)v.elementAt(i);
                String name = "unknown";
                try{
                    name = bluetooth.getRemoteDevice().getFriendlyName(false)
                    + ", " + bluetooth.getBtsppURL();
                }catch (Exception e){
                    name = "unknown";
                }

                g.addCommand(this.guiComponentImpl.getcommand_connectBTdevice());
                g.append(name,null);
            }
        }

        this.guiComponentImpl.setActiveDisplayable(this.guiComponentImpl.
        getdisplayable_bluetoothForm());
    }

    public void setGuiComponent(GuiComponentImpl gui) {
        this.guiComponentImpl = gui;
    }

    public void setGpsComponent(GpsComponent gps){
        this.gpsComponent = gps;
    }

    public void connectGPS(String bluetoothDeviceUrl) {
        Log.println(this, "connect to " + bluetoothDeviceUrl);
        final String url = bluetoothDeviceUrl;

        Thread t = new Thread(){
            public void run(){
                try {
                    StreamConnection conn =
                    (StreamConnection)Connector.open(url,
                    Connector.READ_WRITE, true);
                    InputStream in = conn.openInputStream();
                    gpsComponent.setConnection(in);
                }catch (Exception ex) {
                    Log.println(BluetoothCmdListener.this, "ERROR: " +
                    ex.toString());
                    gpsConnectFailedEvent.fireEvent();
```

```
                }
            }
        };

        t.start();
    }

    public void doDeviceDiscovery() {
        this.bluetooth.setBluetoothListener(this);
        this.bluetooth.doDiscoverDevice();
    }

    public void setgpsConnectFailedEvent(Event gpsConnectFailedEvent) {
        this.gpsConnectFailedEvent=gpsConnectFailedEvent;
    }
    public void setmobCanvas(MobCanvas mobCanvas){
        this.mobCanvas=mobCanvas;
    }
}
```

CHAPTER 4

GpsCmdListener.java

```
/*
 * GpsCmdListener.java
 *
 * This class permits a connection to a GPS device. It gives uses events
 * to notify of positions. Listeners to this class then receives updates on ^
 the new position.
 *
 * Created on 28. März 2007, 14:40
 * @author lefebure
 */

package de.unternehmertum.mobile.mycityguide;

import de.unternehmertum.mobile.playtools.event.Event;
import de.unternehmertum.mobile.playtools.gps.GpsComponent;
import de.unternehmertum.mobile.playtools.gps.GpsData;
import de.unternehmertum.mobile.playtools.gps.GpsListener;
import de.unternehmertum.mobile.playtools.logging.Log;

public class GpsCmdListener implements GpsListener{


    private GpsComponent gpsComponent;

    private Event positionAvailableEvent;
```

```
    private Event positionLostEvent;

    private boolean positionAvailableEventFired = false;

    /** Creates a new instance of GpsCmdListener */
    public GpsCmdListener() {
    }

    public void notifyGpsListener(int code, final GpsData o) {

        ApplicationData.devicePosition.setLatitude(new Double(o.getLatitude()));
        ApplicationData.devicePosition.setLongitude(new Double(o.getLongitude()));

        if (!positionAvailableEventFired){
            positionAvailableEventFired = true;
            this.positionAvailableEvent.fireEvent();
        }
    }

    public void connectionError(Throwable t) {
        positionAvailableEventFired = false;
        this.positionLostEvent.fireEvent();
        Log.println(this, "connectionError " + t.toString());
    }

    public void setGpsComponent(GpsComponent playtoolsGps) {
        this.gpsComponent = playtoolsGps;
    }

    public void setpositionAvailableEvent(Event event) {
        this.positionAvailableEvent = event;
    }

    public void setpositionLostEvent(Event event) {
        this.positionLostEvent = event;
    }

}
```

CHAPTER 5

Map2CmdListener.java

```
/*
 * MapCanvasCmdListener.java
 *
 * This class is a the class responsible for displaying a map. It uses a class
 defined in the play.Tools framework
 * that extends the Canva class. It permits the display of a map with GPS
 coordinates in the center.
 * It is possible to add points of interst, and images on the top of the map.
 * Created on 28. März 2007 2007, 09:15
 *
 * @author lefebure
 */

package de.unternehmertum.mobile.mycityguide;

import de.unternehmertum.mobile.playtools.ComponentException;
import de.unternehmertum.mobile.playtools.gps.GpsComponent;
import de.unternehmertum.mobile.playtools.gui.GuiCmdListener;
import de.unternehmertum.mobile.playtools.gui.GuiCommand;
import de.unternehmertum.mobile.playtools.gui.GuiCommandResult;
import de.unternehmertum.mobile.playtools.gui.GuiComponentImpl;
import de.unternehmertum.mobile.playtools.logging.Log;

import de.unternehmertum.mobile.playtools.map2.MapCanvas;
import de.unternehmertum.mobile.playtools.map2.MapController;
```

```
import de.unternehmertum.mobile.playtools.map2.MapDisplayable;
import de.unternehmertum.mobile.playtools.server.service.BinaryService;
import de.unternehmertum.mobile.playtools.server.service.MapService;
import de.unternehmertum.playbox.remoting.Position;
import java.util.Enumeration;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.Item;

public class Map2CmdListener extends GuiCmdListener{

    private BinaryService binaryService;
    private MapService mapService;
    private GuiComponentImpl guiComponentImpl;
    private MapController mapController;
    private GpsComponent positionComponent;


    public int cmdAction(GuiCommand c, Displayable d, Item item) {

        if (c.getName().equalsIgnoreCase("showMap")){
            Log.println(this, "showMap");
            showMap();
        }
        return 0;
    }

    public void setBinaryService(BinaryService remoteBinaryService) {
        this.binaryService = remoteBinaryService;
    }

    public void setMapService(MapService remoteMapService) {
        this.mapService = remoteMapService;
    }

    public void setGuiComponent(GuiComponentImpl guiComponentImpl) {
        this.guiComponentImpl = guiComponentImpl;
    }

    public void checkInit() throws ComponentException {
        if (this.binaryService == null){
            throw new ComponentException("binaryService is missing");
        }

        if (this.mapService == null){
```

```
            throw new ComponentException("mapService is missing");
        }

        MapCanvas mapCanvas = this.guiComponentImpl.getdisplayable_mapCanvas();
        mapCanvas.setBinaryService(binaryService);
        mapCanvas.setMapService(mapService);
        mapCanvas.checkInit();
        this.mapController = mapCanvas.getMapController();

        this.guiComponentImpl.getdisplayable_mapCanvas().setFullScreenMode(true);


    }
    /**
     * Show a map with the GPS coordinates as the center of the map.
     **/
    public void showMap() {

        //For testing purposes
        //Position center = new Position(new
        Double(this.positionComponent.getPosition().getLatitude()),new
        Double(this.positionComponent.getPosition().getLongitude()));

        //Position center = new Position(new Double(48.161900),new
        Double(11.585800));




     this.mapController.setMapCircularArea(ApplicationData.defaultGPSPosition,
     800);

        //Show map

this.guiComponentImpl.setActiveDisplayable(this.guiComponentImpl.
getdisplayable_mapCanvas());

    }

    public void updatePosition(Position position){
         MapDisplayable posDisp = new MapDisplayable();
         this.mapController.addMapDisplayable(posDisp);
         posDisp.setPosition(position);
    }
```

```
    public void addPOI(Position position, String description){
         MapDisplayable posDisp = new MapDisplayable();
         this.mapController.addMapDisplayable(posDisp);
         posDisp.setPosition(position);
         posDisp.setImage(guiComponentImpl.getimage_poiImage());
         posDisp.setVisible(true);
         posDisp.setDescription(description);
    }

    public void addPosition(Position position, Image image, String
    description) {
         MapDisplayable posDisp = new MapDisplayable();
         this.mapController.addMapDisplayable(posDisp);
         posDisp.setPosition(position);
         posDisp.setImage(image);
         posDisp.setVisible(true);
         posDisp.setDescription(description);
     }
     public void zoomIn() {
         this.mapController.zoomIn();
     }
     public void zoomOut(){
         this.mapController.zoomOut();
     }
}
```

CHAPTER 6

MyCityGuideService.java

```
/*
 * RestaurantService.java
 *
 * This class is responsible for acces to the web service that is implmented
 on the server.
 * It permits several method calls that is defined in the web service, such as
 event search or place search.
 *
 * Created on 2. April 2007, 15:28
 *
 * @author lefebure
 */

package de.unternehmertum.mobile.mycityguide;

import de.unternehmertum.mobile.playtools.server.AbstractService;
import de.unternehmertum.mobile.playtools.server.Call;
import de.unternehmertum.mobile.playtools.server.RemoteServiceListener;
import de.unternehmertum.mobile.playtools.server.ServerException;
import de.unternehmertum.playbox.remoting.RestaurantServicePortType_Stub;
import de.unternehmertum.playbox.remoting.UserData;
import de.unternehmertum.playbox.remoting.UserServicePortType_Stub;

public class RestaurantService extends AbstractService{
```

```
RestaurantServicePortType_Stub service;

/**
 * Creates a new instance of RestaurantService
 */
public RestaurantService() {
    super();
    this.setWebServiceName("RestaurantService");
    service = new RestaurantServicePortType_Stub();
}

public UserData userLogin(String loginName, String passwordHash) throws
ServerException {
    try{
        return service.userLogin(loginName, passwordHash);
    } catch (Exception ex) {
        handleException(ex);
    }
    return null;
}

/**
 * asynchron
 * returns via Callback Boolean true/false
 */
public int userLogin(final String loginName, final String passwordHash,
final RemoteServiceListener callback) {
    return asynchronousCall(new Call(){
        public Object call() throws ServerException{

            UserData result = userLogin(loginName, passwordHash);
            return result;

        }
    }, callback);
}

 protected String getEndpoint_Adress_Property() {
    return service.ENDPOINT_ADDRESS_PROPERTY;
}

protected String getSession_Maintain_Property() {
    return service.SESSION_MAINTAIN_PROPERTY;
}
```

```
    protected void _setProperty(String name, Object value) {
         service._setProperty(name, value);
     }

     protected Object _getProperty(String name) {
         return service._getProperty(name);
     }
}
```

CHAPTER 7

SecurityCmdListener.java

```
/*
 * SecurityCmdListener.java
 *
 * This class is responsible for logg in to the server. It provides a username
 / password mechanism
 * giving access to web services provided by the database. This class notifies
 other classes wether the logg in
 * was succesfull or not.
 *
 * Created on 1. April 2007, 17:09
 *
 * @author lefebure
 */

package de.unternehmertum.mobile.mycityguide;

import de.unternehmertum.mobile.playtools.event.Event;
import de.unternehmertum.mobile.playtools.security.SecurityComponent;
import de.unternehmertum.mobile.playtools.security.SecurityException;
import de.unternehmertum.mobile.playtools.security.SecurityComponentListener;
import de.unternehmertum.mobile.playtools.security.UserPassword;
import de.unternehmertum.mobile.playtools.util.encoders.MD5;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Item;
```

```
/**
 *
 *
 */
public class SecurityCmdListener implements SecurityComponentListener{

    private SecurityComponent securityComponent;

    private Event loggedInEvent;
    private Event loggedInFailedEvent;

    private RestaurantService restaurantService;

    /** Creates a new instance of SecurityCmdListener */
    public SecurityCmdListener() {
    }

    public void loggIn(){
        String userName="user";
        String pwd="CityMobUser";
        try {
               this.securityComponent.login(new
               UserPassword(userName,MD5.calcMD5(pwd)));
            } catch (SecurityException ex) {
                ex.printStackTrace();
            }
    }

    public void loginSucessful(de.unternehmertum.playbox.remoting.UserData user) {

        this.loggedInEvent.fireEvent();
    }

    public void loginFailed(Object o) {
        this.loggedInFailedEvent.fireEvent();
    }

    public void setSecurityComponent(SecurityComponent securityComponent) {
        this.securityComponent = securityComponent;
    }

     public void setloggedInEvent(Event loggedInEvent) {
        this.loggedInEvent = loggedInEvent;
    }
      public void setloggedInFailedEvent(Event loggedInFailedEvent) {
```

```
        this.loggedInFailedEvent = loggedInFailedEvent;
    }


}
```

Part II

Server source code - MunichX Server

CHAPTER 8

model.Cinema.java

```
/*
 * Cinema.java
 *
 * This represents the Cinema object that permits the storage of cinema
 objects to the          * database.
 *
 * Created on 16. April 2007, 17:01
 *
 *  @author lefebure
 */

package de.unternehmertum.playbox.model;
import de.unternehmertum.playbox.model.util.SoapTypeConverter;

/**
 *
 *
 */

@SoapTypeConverter.ConverterFlags(
transientFields = {},
        linkedClassName = "de.unternehmertum.playbox.model.remote.Cinema"
        )

        public class Cinema implements BaseModel {
```

```
private long id = NOT_SPECIFIED;
private String opening_data = null;
private long place_id = -1;

private Place place = new Place();
private Movie[] movies = new Movie[0];

/** Creates a new instance of Cinema */
public Cinema() {
}
public Cinema(Cinema original) {
    this.setId(original.getId());
    this.setPlace(original.getPlace());
    this.setPlace_id(original.getPlace_id());
    Movie[] movies = new Movie[original.getMovies().length];
    System.arraycopy(original.getMovies(), 0, movies, 0, movies.length);
    setMovies(movies);

    this.setOpening_data(original.getOpening_data());
}
public void setId(long l) {
    this.id=l;
}
public long getId() {
    return id;
}

public void setPlace(Place place){
    this.place=place;
}
public Place getPlace(){
    return place;
}

public void setMovies(Movie[] movies){
    this.movies=movies;
}
public Movie[] getMovies(){
    return movies;
}

public void setOpening_data(String string) {
    this.opening_data=string;
}
```

```
    public String getOpening_data() {
        return opening_data;
    }

    public void setPlace_id(long l) {
        this.place_id=l;
    }
    public long getPlace_id() {
        return place_id;
    }
}
```

CHAPTER 9

model.Cinema.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="de.unternehmertum.playbox.model.Cinema" table="cinema"
    lazy="false">
        <id name="id" column="id">
            <generator class="native"/>
        </id>
        <property name="opening_data"/>

        <many-to-one name="place" column="place_id" update="false"
        unique="true" class="de.unternehmertum.playbox.model.Place"/>

        <array name="movies" cascade="all" batch-size="10"
        element-class="de.unternehmertum.playbox.model.Movie" >
            <key column="id"/>
            <list-index column="poll_id" />
            <one-to-many entity-name="AMovie"/>
        </array>

    </class>
</hibernate-mapping>
```

CHAPTER 10

model.CityEvent.java

```
/*
 * CityEvent.java
 *
 * This represents the CityEvent object that permits the storage of CityEvent
 objects to the database.
 * Created on 11. April 2007, 13:23
 *
 * @author lefebure
 */

package de.unternehmertum.playbox.model;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;
import java.util.Date;


/**
 *
 *
 */
@SoapTypeConverter.ConverterFlags(
transientFields = {},
linkedClassName = "de.unternehmertum.playbox.model.remote.CityEvent")
public class CityEvent implements BaseModel{
  private long id = NOT_SPECIFIED;
```

```
private String type = null;
private String name = null;
private String description = null;
private String price = null;
private String opening_data = null;

private long place_id = -1;
private long poll_id = -1;

private Place place = new Place();
private Poll poll = new Poll();
  /**
   * Creates a new instance of CityEvent
   */
  public CityEvent() {
  }
  public CityEvent(CityEvent original) {
     this.setId(original.getId());
     this.setType(original.getType());
     this.setName(original.getName());
     this.setDescription(original.getDescription());
     this.setPrice(original.getPrice());
     this.setOpening_data(original.getOpening_data());
     this.setPoll(original.getPoll());
     this.setPlace_id(original.getPlace_id());
     this.setPoll_id(original.getPoll_id());
     this.setPlace(original.getPlace());
  }

  public long getId() {
      return id;
  }

  public String getType() {
      return type;
  }

  public String getName() {
      return name;
  }

  public String getDescription() {
      return description;
  }
```

```
public String getPrice() {
    return price;
}
public void setPoll(Poll l) {
    this.poll=l;
}
 public Poll getPoll() {
    return poll;
}

public void setId(long l) {
    this.id=l;
}

public void setType(String string) {
    this.type=string;
}

public void setName(String string) {
    this.name=string;
}

public void setDescription(String string) {
    this.description=string;
}

public void setPrice(String string) {
    this.price=string;
}



public Place getPlace(){
    return place;
}
public void setPlace(Place place){
    this.place=place;
}
public String getOpening_data() {
    return opening_data;
}

public void setOpening_data(String date) {
    this.opening_data=date;
}
```

```
    public void setPlace_id(long place_id){
        this.place_id=place_id;
    }
    public long getPlace_id(){
        return place_id;
    }
    public void setPoll_id(long poll_id){
        this.poll_id=poll_id;
    }
    public long getPoll_id(){
        return poll_id;
    }
}
```

CHAPTER 11

model.CityEvent.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
<class name="de.unternehmertum.playbox.model.CityEvent" table="city_event"
lazy="false">
<id name="id" column="id">
<generator class="native"/>
</id>
                <property name="type"/>
                <property name="name"/>
                <property name="description"/>
                <property name="price"/>
                <property name="opening_data"/>

                <many-to-one name="poll" column="poll_id" update="false"
                unique="true" class="de.unternehmertum.playbox.model.Poll"/>
                <many-to-one name="place" column="place_id" update="false"
                unique="true" class="de.unternehmertum.playbox.model.Place"/>

</class>
</hibernate-mapping>
```

CHAPTER 12

model.Movie.java

```
/*
 * Movie.java
 *
 * This represents the Movie object that permits the storage of Movie objects
 to the database.
 * Created on 11. April 2007, 13:23
 *
 * @author lefebure
 */

package de.unternehmertum.playbox.model;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;

/**
 *
 * @author lefebure
 */
@SoapTypeConverter.ConverterFlags(
transientFields = {},
linkedClassName = "de.unternehmertum.playbox.model.remote.Movie")
public class Movie implements BaseModel{
     private long id = NOT_SPECIFIED;
     private String title = null;
     private long year = -1;
```

```
 private String description = null;
 private String genre = null;
 private String info = null;

 private long poll_id = -1;
 private Poll poll = new Poll();

 //private long place_id = -1;

/** Creates a new instance of Movie */
public Movie() {
}
public Movie(Movie original) {
   this.setId(original.getId());
   this.setTitle(original.getTitle());
   this.setYear(original.getYear());
   this.setDescription(original.getDescription());
   this.setGenre(original.getGenre());
   this.setInfo(original.getInfo());
   this.setPoll_id(original.getPoll_id());
   this.setPoll(original.getPoll());
   //this.setPlace_id(original.getPlace_id());

}
public long getId() {
    return id;
}
public void setId(long l) {
    this.id=l;
}

public void setPoll(Poll l) {
    this.poll=l;
}
public Poll getPoll() {
    return poll;
}

public void setPoll_id(long l) {
    this.poll_id=l;
}
public long getPoll_id() {
    return poll_id;
}
```

```
    public void setDescription(String string) {
        this.description=string;
    }
    public String getDescription() {
        return description;
    }

    public String getTitle() {
        return title;
    }
    public void setTitle(String string) {
        this.title=string;
    }

    public long getYear() {
        return year;
    }
    public void setYear(long l) {
        this.year=l;
    }

    public String getGenre() {
        return genre;
    }
    public void setGenre(String string) {
        this.genre=string;
    }

    public String getInfo() {
        return info;
    }
    public void setInfo(String string) {
        this.info=string;
    }
     /*
    public void setPlace_id(long l) {
        this.place_id=l;
    }
    public long getPlace_id() {
        return place_id;
    }
     */
}
```

CHAPTER 13

model.Movie.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
<class name="de.unternehmertum.playbox.model.Movie" entity-name="AMovie"
table="movie" lazy="false">
<id name="id" column="id">
<generator class="native"/>
</id>
                <property name="title"/>
                <property name="year"/>
                <property name="description"/>
                <property name="genre"/>
                <property name="info"/>

                <many-to-one name="poll" column="poll_id" update="false"
                unique="true" class="de.unternehmertum.playbox.model.Poll"/>
</class>
</hibernate-mapping>
```

CHAPTER 14

model.Place.java

```
/*
 * Place.java
 *
 * Created on 11. April 2007, 13:23
 *
 * This represents the Place object that permits the storage of Place objects
 to the database.
 * Created on 11. April 2007, 13:23
 *
 * @author lefebure
 */

package de.unternehmertum.playbox.model;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;

@SoapTypeConverter.ConverterFlags(
transientFields = {},
        linkedClassName = "de.unternehmertum.playbox.model.remote.Place")
        public class Place implements BaseModel{
    private long id = NOT_SPECIFIED;
    private String name = null;
    private String description = null;
    private String type = null;
    private String category = null;
```

```
private String price = null;
private String city_district = null;
private String phone_number = null;
private String opening_data = null;

private long placeAddress_id = -1;
private PlaceAddress placeAddress = new PlaceAddress();

private long poll_id = -1;
private Poll poll = new Poll();

/** Creates a new instance of Place */
public Place(){
}

public Place(Place original) {
    this.setId(original.getId());
    this.setType(original.getType());
    this.setCategory(original.getCategory());
    this.setDescription(original.getDescription());
    this.setPrice(original.getPrice());
    this.setPoll_id(original.getPoll_id());
    this.setPoll(original.getPoll());
    this.setName(original.getName());
    this.setCity_district(original.getCity_district());
    this.setPhone_number(original.getPhone_number());
    this.setOpening_data(original.getOpening_data());
    this.setPlaceAddress_id(original.getPlaceAddress_id());
    this.setPlaceAddress(original.getPlaceAddress());


}
public long getId() {
    return id;
}
public void setId(long l) {
    this.id=l;
}

public String getType() {
    return type;
}
public void setType(String string) {
    this.type=string;
}
```

```
public String getCategory() {
    return category;
}
public void setCategory(String string) {
    this.category=string;
}

public String getDescription() {
    return description;
}
public void setDescription(String string) {
    this.description=string;
}

public String getPrice() {
    return price;
}
public void setPrice(String string) {
    this.price=string;
}

public void setPoll_id(long poll_id){
    this.poll_id=poll_id;
}
public long getPoll_id(){
    return poll_id;
}

public Poll getPoll() {
    return poll;
}
public void setPoll(Poll l) {
    this.poll=l;
}

public String getName() {
    return name;
}
public void setName(String string) {
    this.name=string;
}

public String getCity_district() {
    return city_district;
```

```
    }
    public void setCity_district(String string) {
        this.city_district=string;
    }

    public String getPhone_number() {
        return phone_number;
    }
    public void setPhone_number(String string) {
        this.phone_number=string;
    }

    public void setOpening_data(String string) {
        this.opening_data=string;
    }
    public String getOpening_data() {
        return opening_data;
    }

    public PlaceAddress getPlaceAddress(){
        return placeAddress;
    }
    public void setPlaceAddress(PlaceAddress placeAddress){
        this.placeAddress=placeAddress;
    }
    public void setPlaceAddress_id(long placeAddress_id){
        this.placeAddress_id=placeAddress_id;
    }
    public long getPlaceAddress_id(){
        return placeAddress_id;
    }
}
```

CHAPTER 15

model.Place.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="de.unternehmertum.playbox.model.Place" table="place"
    lazy="false">
        <id name="id" column="id">
            <generator class="native"/>
        </id>
        <property name="type"/>
        <property name="category"/>
        <property name="price"/>
        <property name="name"/>
        <property name="description"/>
        <property name="city_district"/>
        <property name="phone_number"/>
        <property name="opening_data"/>

        <many-to-one name="placeAddress" column="placeAddress_id"
        unique="true" update="false"
        class="de.unternehmertum.playbox.model.PlaceAddress"/>
        <many-to-one name="poll" column="poll_id" unique="true" update="false"
        class="de.unternehmertum.playbox.model.Poll"/>
```

```
    </class>
</hibernate-mapping>
```

CHAPTER 16

model.PlaceAddress.java

```
/*
 * PlaceAddress.java
 *
 * This represents the PlaceAddress object that permits the storage of
 PlaceAddress objects to the database.
 * Created on 11. April 2007, 13:23
 *
 * @author lefebure
 */

package de.unternehmertum.playbox.model;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;


@SoapTypeConverter.ConverterFlags(
transientFields = {},
linkedClassName = "de.unternehmertum.playbox.model.remote.PlaceAddress")
public class PlaceAddress implements BaseModel{
  private long id = NOT_SPECIFIED;
  private String street = null;
  private String city = null;
  private long zip = -1;
  private String country = null;
  private double latitude = -1;
```

```
private double longitude = -1;

  /** Creates a new instance of Location */
  public PlaceAddress() {
  }
  public PlaceAddress(PlaceAddress original) {
     this.setId(original.getId());
     this.setStreet(original.getStreet());
     this.setCity(original.getCity());
     this.setZip(original.getZip());
     this.setCountry(original.getCountry());
     this.setLatitude(original.getLatitude());
     this.setLongitude(original.getLongitude());
  }

  public long getId() {
      return id;
  }

  public String getStreet() {
      return street;
  }

  public String getCity() {
      return city;
  }

  public long getZip() {
      return zip;
  }

  public String getCountry() {
      return country;
  }

  public double getLatitude() {
      return latitude;
  }

  public double getLongitude() {
      return longitude;
  }

  public void setId(long l) {
      this.id=l;
```

```
    }

   public void setStreet(String string) {
        this.street=string;
    }

    public void setCity(String string) {
        this.city=string;
    }

    public void setZip(long l) {
        this.zip=l;
    }

    public void setCountry(String string) {
        this.country=string;
    }

    public void setLatitude(double d) {
        this.latitude=d;
    }

    public void setLongitude(double d) {
        this.longitude=d;
    }
}
```

CHAPTER 17

model.PlaceAddress.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
<class name="de.unternehmertum.playbox.model.PlaceAddress"
table="place_address" lazy="false">
<id name="id" column="id">
<generator class="native"/>
</id>
                <property name="street"/>
                <property name="city"/>
                <property name="zip"/>
                <property name="country"/>
                <property name="latitude"/>
                <property name="longitude"/>

</class>
</hibernate-mapping>
```

CHAPTER 18

model.Poll.java

```
/*
 * Poll.java
 *
 * This represents the Poll object that permits the storage of Poll objects to
 the database.
 * Created on 11. April 2007, 13:23
 *
 * @author lefebure
 */

package de.unternehmertum.playbox.model;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;

/**
 *
 * @author lefebure
 */
@SoapTypeConverter.ConverterFlags(
transientFields = {},
linkedClassName = "de.unternehmertum.playbox.model.remote.Poll")
public class Poll implements BaseModel{
     private long id = NOT_SPECIFIED;
     //private long poll_id = -1;
     private long value = -1;
```

```
     private long number_polls = -1;
    /** Creates a new instance of Poll */
    public Poll() {
    }
    public Poll(Poll original) {
         this.setId(original.getId());
         //this.setPoll_id(original.getPoll_id());
         this.setValue(original.getValue());
         this.setNumber_polls(original.getNumber_polls());
    }

    public long getId() {
        return id;
    }



    public long getValue() {
        return value;
    }

    public long getNumber_polls() {
        return number_polls;
    }

    public void setId(long l) {
        this.id=l;
    }
/*
    public void setPoll_id(long l) {
        this.poll_id=l;
    }
    public long getPoll_id() {
        return poll_id;
    }
 **/
    public void setValue(long l) {
        this.value=l;
    }

    public void setNumber_polls(long l) {
        this.number_polls=l;
    }

}
```

CHAPTER 19

model.Poll.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
<class name="de.unternehmertum.playbox.model.Poll" table="poll" lazy="false">
<id name="id" column="id">
<generator class="native"/>
</id>
                <property name="value"/>
                <property name="number_polls"/>
</class>
</hibernate-mapping>
```

CHAPTER 20

model.remote.Cinema.java

```
/*
 * Cinema.java
 *
 * Created on 16. April 2007, 17:03
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package de.unternehmertum.playbox.model.remote;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;
/**
 *
 * @author lefebure
 */
@SoapTypeConverter.ConverterFlags(
transientFields = {},
linkedClassName = "de.unternehmertum.playbox.model.Cinema",
    isRemote = true
)
public class Cinema {

    private long id;
    private long place_id;
```

```
    private String opening_data = null;

    private Place place;
    private Movie[] movies;
    /** Creates a new instance of Cinema */

    public void setId(long l) {
        this.id=l;
    }
    public long getId() {
        return id;
    }
     public void setPlace(Place place){
        this.place=place;
    }
    public Place getPlace(){
        return place;
    }
     public void setMovies(Movie[] movies){
        this.movies=movies;
    }
    public Movie[] getMovies(){
        return movies;
    }
    public void setOpening_data(String string) {
        this.opening_data=string;
    }
     public String getOpening_data() {
        return opening_data;
    }
      public void setPlace_id(long l) {
        this.place_id=l;
    }
      public long getPlace_id() {
        return place_id;
    }
}
```

CHAPTER 21

model.remote.CityEvent.java

```
/*
 * CityEvent.java
 *
 * Created on 13. April 2007, 12:01
 *s
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package de.unternehmertum.playbox.model.remote;
import de.unternehmertum.playbox.model.util.SoapTypeConverter;
import java.util.Date;

/**
 *
 * @author lefebure
 */
@SoapTypeConverter.ConverterFlags(
transientFields = {},
linkedClassName = "de.unternehmertum.playbox.model.CityEvent",
isRemote = true)
public class CityEvent{
  private long id;
  private String type;
  private String name;
```

```
private String description;
private String price;
private String opening_data;
private long place_id;
private long poll_id;

private Place place;
private Poll poll;

  public long getId() {
      return id;
  }

  public String getType() {
      return type;
  }

  public String getName() {
      return name;
  }

  public String getDescription() {
      return description;
  }

  public String getPrice() {
      return price;
  }

  public Poll getPoll() {
      return poll;
  }

  public void setId(long l) {
      this.id=l;
  }

  public void setType(String string) {
      this.type=string;
  }

  public void setName(String string) {
      this.name=string;
  }
```

```
    public void setDescription(String string) {
        this.description=string;
    }

    public void setPrice(String string) {
        this.price=string;
    }

    public void setPoll(Poll l) {
        this.poll=l;
    }

    public Place getPlace(){
        return place;
    }
    public void setPlace(Place place){
        this.place=place;
    }

    public String getOpening_data() {
        return opening_data;
    }

    public void setOpening_data(String date) {
        this.opening_data=date;
    }
    public void setPlace_id(long place_id){
        this.place_id=place_id;
    }
    public long getPlace_id(){
        return place_id;
    }
    public void setPoll_id(long poll_id){
        this.poll_id=poll_id;
    }
    public long getPoll_id(){
        return poll_id;
    }
}
```

CHAPTER 22

model.remote.Movie.java

```
/**
 * Movie.java
 *
 * Created on 11. April 2007, 13:23
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package de.unternehmertum.playbox.model.remote;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;

/**
 *
 * @author lefebure
 */
@SoapTypeConverter.ConverterFlags(
transientFields = {},
        linkedClassName = "de.unternehmertum.playbox.model.Movie",
        isRemote = true)

        public class Movie{
    private long id;
    private String title;
```

```
private long year;
private String description;
private String genre;
private String info;


//private long place_id;

private long poll_id;
private Poll poll;



public long getId() {
    return id;
}
public void setId(long l) {
    this.id=l;
}

public void setPoll(Poll l) {
    this.poll=l;
}
public Poll getPoll() {
    return poll;
}

public void setPoll_id(long l) {
    this.poll_id=l;
}
public long getPoll_id() {
    return poll_id;
}

public void setDescription(String string) {
    this.description=string;
}
public String getDescription() {
    return description;
}

public String getTitle() {
    return title;
}
public void setTitle(String string) {
```

```
        this.title=string;
    }

    public long getYear() {
        return year;
    }
    public void setYear(long l) {
        this.year=l;
    }

    public String getGenre() {
        return genre;
    }
    public void setGenre(String string) {
        this.genre=string;
    }

    public String getInfo() {
        return info;
    }
    public void setInfo(String string) {
        this.info=string;
    }

    /*
    public void setPlace_id(long l) {
        this.place_id=l;
    }
    public long getPlace_id() {
        return place_id;
    }*/
}
```

CHAPTER 23

model.remote.Place.java

```
/*
 * Place.java
 *
 * Created on 11. April 2007, 13:23
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package de.unternehmertum.playbox.model.remote;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;

/**
 *
 * @author lefebure
 */
@SoapTypeConverter.ConverterFlags(
transientFields = {},
linkedClassName = "de.unternehmertum.playbox.model.Place",
isRemote = true)

public class Place{
  private long id;
  private String name;
```

```
private String description;
private String type;
private String category;
private String price;
private String city_district;
private String phone_number;
private String opening_data;

private long placeAddress_id;
private PlaceAddress placeAddress;

private long poll_id;
private Poll poll;

  public long getId() {
      return id;
  }

  public String getType() {
      return type;
  }

  public String getCategory() {
      return category;
  }

  public String getDescription() {
      return description;
  }

  public String getPrice() {
      return price;
  }
  public void setId(long l) {
      this.id=l;
  }

  public void setType(String string) {
      this.type=string;
  }

  public void setCategory(String string) {
      this.category=string;
  }
```

```
public void setDescription(String string) {
    this.description=string;
}

public void setPrice(String string) {
    this.price=string;
}

public void setPoll(Poll l) {
    this.poll=l;
}
public Poll getPoll() {
    return poll;
}
public String getName() {
    return name;
}

public String getCity_district() {
    return city_district;
}

public String getPhone_number() {
    return phone_number;
}

public void setName(String string) {
    this.name=string;
}

public void setCity_district(String string) {
    this.city_district=string;
}

public void setPhone_number(String string) {
    this.phone_number=string;
}

public void setOpening_data(String string) {
    this.opening_data=string;
}
public String getOpening_data() {
    return opening_data;
}
```

```
    public PlaceAddress getPlaceAddress(){
        return placeAddress;
    }
    public void setPlaceAddress(PlaceAddress placeAddress){
        this.placeAddress=placeAddress;
    }
     public void setPoll_id(long poll_id){
        this.poll_id=poll_id;
    }

    public long getPoll_id(){
        return poll_id;
    }
     public void setPlaceAddress_id(long placeAddress_id){
        this.placeAddress_id=placeAddress_id;
    }

    public long getPlaceAddress_id(){
        return placeAddress_id;
    }
}
```

CHAPTER 24

model.remote.PlaceAddress.java

```
/*
 * PlaceAddress.java
 *
 * Created on 11. April 2007, 13:23
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package de.unternehmertum.playbox.model.remote;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;
/**
 *
 * @author lefebure
 */
@SoapTypeConverter.ConverterFlags(
transientFields = {},
        linkedClassName = "de.unternehmertum.playbox.model.PlaceAddress",
        isRemote = true)

        public class PlaceAddress{
    private long id;
    private String street;
```

```
private String city;
private long zip;
private String country;
private double latitude;
private double longitude;

public long getId() {
    return id;
}

public String getStreet() {
    return street;
}

public String getCity() {
    return city;
}

public long getZip() {
    return zip;
}

public String getCountry() {
    return country;
}

public double getLatitude() {
    return latitude;
}

public double getLongitude() {
    return longitude;
}

public void setId(long l) {
    this.id=l;
}

public void setStreet(String string) {
    this.street=string;
}

public void setCity(String string) {
    this.city=string;
}
```

```
    public void setZip(long l) {
        this.zip=l;
    }

    public void setCountry(String string) {
        this.country=string;
    }

    public void setLatitude(double d) {
        this.latitude=d;
    }

    public void setLongitude(double d) {
        this.longitude=d;
    }
}
```

CHAPTER 25

model.remote.Poll.java

```
/*
 * Poll.java
 *
 * Created on 11. April 2007, 13:23
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package de.unternehmertum.playbox.model.remote;

import de.unternehmertum.playbox.model.util.SoapTypeConverter;

/**
 *
 * @author lefebure
 */
@SoapTypeConverter.ConverterFlags(
transientFields = {},
linkedClassName = "de.unternehmertum.playbox.model.Poll",
isRemote = true)
public class Poll{
     private long id;
     //private long poll_id;
```

```
     private long value;
     private long number_polls;

    public long getId() {
        return id;
    }



    public long getValue() {
        return value;
    }

    public long getNumber_polls() {
        return number_polls;
    }

    public void setId(long l) {
        this.id=l;
    }
/*
    public void setPoll_id(long l) {
        this.poll_id=l;
    }
     public long getPoll_id() {
        return poll_id;
    }*/
    public void setValue(long l) {
        this.value=l;
    }

    public void setNumber_polls(long l) {
        this.number_polls=l;
    }
}
```

CHAPTER 26

service.external.MyCityGuideServiceExternalBean.java

```
 /*
  * MyCityGuideServiceExternalBean.java
  *
  * Created on 29. März 2007, 13:27
  *
  * To change this template, choose Tools | Template Manager
  * and open the template in the editor.
  */

package de.unternehmertum.playbox.service.external;
import de.unternehmertum.playbox.model.remote.Cinema;
import de.unternehmertum.playbox.model.remote.Movie;
import de.unternehmertum.playbox.model.remote.Place;
import de.unternehmertum.playbox.model.remote.CityEvent;
import de.unternehmertum.playbox.model.remote.PlaceAddress;
import de.unternehmertum.playbox.service.remote.MyCityGuideService;
import de.unternehmertum.playbox.service.exception.ServiceException;
import de.unternehmertum.playbox.service.impl.MyCityGuideServiceBean;
import de.unternehmertum.playbox.model.util.SoapTypeConverter;
public class MyCityGuideServiceExternalBean extends AuthorizingService implements
        MyCityGuideService {
    private MyCityGuideServiceBean restaurantService = null;

    public CityEvent[] searchEvents(final String type, final String
    opening_data ) throws ServiceException {
```

```
        return (CityEvent[])

        SoapTypeConverter.convertToRemote(restaurantService.
        searchEvents(type,opening_data));
    }

    public Place[] placeSearch(final String type, final String city_district,
    final String category)throws ServiceException {
        return (Place[])
        SoapTypeConverter.convertToRemote(restaurantService.
        placeSearch(type, city_district,category));
    }
    public Movie[] listAllMovies() throws ServiceException {
        return (Movie[])
        SoapTypeConverter.convertToRemote(restaurantService.listAllMovies());
    }
    public Cinema[] listAllCinemas(final String
    movieName) throws ServiceException {
        return (Cinema[])

        SoapTypeConverter.convertToRemote(restaurantService.
        listAllCinemas(movieName));
    }

    /*
    public Movie[] movieSearch(final String title, final long place_id, final
    String date) throws ServiceException{
     return (Movie[])
        SoapTypeConverter.convertToRemote(restaurantService.movieSearch(title,
        place_id, date));
    }
     */

    public MyCityGuideServiceBean getRestaurantService() {
        return restaurantService;
    }
    public void setRestaurantService(MyCityGuideServiceBean restaurantService) {
        this.restaurantService = restaurantService;
    }
}
```

CHAPTER 27

service.impl.MyCityGuideServiceBean.java

```
/*
 * MyCityGuideServiceBean.java
 *
 * Created on 01.03.2006
 */
package de.unternehmertum.playbox.service.impl;
import de.unternehmertum.playbox.model.Cinema;
import de.unternehmertum.playbox.model.CityEvent;
import de.unternehmertum.playbox.model.Movie;
import de.unternehmertum.playbox.model.Place;
//import de.unternehmertum.playbox.model.PlaceAddress;
//import de.unternehmertum.playbox.model.Poll;
import de.unternehmertum.playbox.service.internal.AuthorizingServiceInternal;
import de.unternehmertum.playbox.service.internal.MyCityGuideServiceInternal;
import de.unternehmertum.playbox.service.exception.ServiceException;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.hibernate.Session;
import org.hibernate.HibernateException;
import org.hibernate.Query;
import java.util.Collection;

public class MyCityGuideServiceBean extends AbstractServiceBean implements
        MyCityGuideServiceInternal{

    private AuthorizingServiceInternal authorizingService;
```

```
public CityEvent[] searchEvents(final String type, final String
opening_data ) throws ServiceException {
    Collection resultCol = (Collection) executeHibernateCall(new
    HibernateCallback() {
        public Object doInHibernate(Session session) throws
                HibernateException {
            Query query = session.createQuery("from " +
                    CityEvent.class.getName() + " WHERE type LIKE :var and
                    opening_data LIKE :var2");
            query.setString("var",type);
            query.setString("var2",opening_data);
            return query.list();
        }
    });
    if (resultCol.size() == 0) {
        throw new ServiceException("Result of query empty.");
    } else {
        CityEvent[] result = new CityEvent[resultCol.size()];
        result = (CityEvent[]) resultCol.toArray(result);
        return result;
    }
}

public Place[] placeSearch(final String type, final String city_district,
final String category) throws ServiceException {
    Collection resultCol = (Collection) executeHibernateCall(new
    HibernateCallback() {
        public Object doInHibernate(Session session) throws
                HibernateException {
            Query query = session.createQuery("from " +
                    Place.class.getName() + " WHERE type LIKE :var1 and
                    city_district LIKE :var2 and category LIKE :var3");
            query.setString("var1",type);
            query.setString("var2",city_district);
            query.setString("var3",category);
            return query.list();
        }
    });
    if (resultCol.size() == 0) {
        throw new ServiceException("Result of query empty.");
    } else {
        Place[] result = new Place[resultCol.size()];
        result = (Place[]) resultCol.toArray(result);
        return result;
```

```
    }
}


public Cinema[] listAllCinemas(final String movieName) throws
ServiceException {
    Collection resultCol = (Collection) executeHibernateCall(new
    HibernateCallback() {
        public Object doInHibernate(Session session) throws
                HibernateException {
            Query query = session.createQuery("from " +
                    Cinema.class.getName());
            return query.list();
        }
    });
    if (resultCol.size() == 0) {
        throw new ServiceException("Result of query empty.");
    } else {
        Cinema[] result = new Cinema[resultCol.size()];
        result = (Cinema[]) resultCol.toArray(result);
        return result;
    }
}

public Movie[] listAllMovies() throws ServiceException {
    Collection resultCol = (Collection) executeHibernateCall(new
    HibernateCallback() {
        public Object doInHibernate(Session session) throws
                HibernateException {
            Query query = session.createQuery("from " +
                    Movie.class.getName());
            return query.list();
        }
    });
    if (resultCol.size() == 0) {
        throw new ServiceException("Result of query empty.");
    } else {
        Movie[] result = new Movie[resultCol.size()];
        result = (Movie[]) resultCol.toArray(result);
        return result;
    }
}

 /*
public Movie[] movieSearch(final String title, final long place_id, final
```

```
    String date) throws ServiceException {
         Collection resultCol = (Collection) executeHibernateCall(new
         HibernateCallback() {
           public Object doInHibernate(Session session) throws
                   HibernateException {
               Query query = session.createQuery("from " +
                       Movie.class.getName() + " WHERE title LIKE :var1 and
                       place_id= :var2 and date= :var"); //
                            query.setString("var1",title);
                            query.setLong("var2",place_id);
                            query.setString("var3",date);
               return query.list();

           }
         });
         if (resultCol.size() == 0) {
           throw new ServiceException("Result of query empty.");
       } else {
       Movie[] result = new Movie[resultCol.size()];
       result = (Movie[]) resultCol.toArray(result);
       return result;
       }
   }*/

    /**
     * Authorization methods
     **/
    public void setAuthorizingService(AuthorizingServiceInternal
    authorizingService) {
        this.authorizingService = authorizingService;
    }
    protected AuthorizingServiceInternal getAuthorizingService() {
        return authorizingService;
    }
}
```

CHAPTER 28

service.internal.MyCityGuideServiceInternal.java

```
/*
 * MyCityGuideServiceInternal.java
 *
 * Created on 29. März 2007, 10:53
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package de.unternehmertum.playbox.service.internal;
import de.unternehmertum.playbox.model.Cinema;
import de.unternehmertum.playbox.model.CityEvent;
import de.unternehmertum.playbox.model.Movie;
import de.unternehmertum.playbox.model.Place;
import de.unternehmertum.playbox.model.PlaceAddress;
import de.unternehmertum.playbox.service.exception.ServiceException;

public interface MyCityGuideServiceInternal {

    public CityEvent[] searchEvents(String type, String opening_data ) throws
    ServiceException;
    public Place[] placeSearch(String type, String city_district, String
    category)throws ServiceException;
    public Movie[] listAllMovies() throws ServiceException;
    public Cinema[] listAllCinemas(String movieName) throws ServiceException;
```

```
    //public Movie[] movieSearch(String title, long place_id, String date)
    throws ServiceException;


}
```

CHAPTER 29

service.remote.MyCityGuideService.java

```
 /*
 * MyCityGuideService.java
 *
 * Created on 29. März 2007, 13:21
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package de.unternehmertum.playbox.service.remote;
import de.unternehmertum.playbox.service.exception.ServiceException;
/**
 *
 * @author lefebure
 */
public interface MyCityGuideService extends AuthorizingService {

public de.unternehmertum.playbox.model.remote.CityEvent[] searchEvents(
String type, String opening_data ) throws ServiceException;
public de.unternehmertum.playbox.model.remote.Place[] placeSearch(
String type, String city_district, String category)throws ServiceException;
public de.unternehmertum.playbox.model.remote.Movie[] listAllMovies()
throws ServiceException;
public de.unternehmertum.playbox.model.remote.Cinema[] listAllCinemas(
```

```
String movieName) throws ServiceException;
//public de.unternehmertum.playbox.model.remote.Movie[] movieSearch(
String title, long place_id, String date) throws ServiceException;

}
```

CHAPTER 30

MyCityGuideService Web Service Specification

```
bean id="MyCityGuideServiceExternalWS" class="org.codehaus.xfire.spring.ServiceBean">
        <property name="serviceBean">
            <bean id="myCityGuideServiceExternalBean"
                  class="de.unternehmertum.playbox.service.external.
                  MyCityGuideServiceExternalBean">
                <property name="myCityGuideService" ref="myCityGuideService"/>
                <property name="authorizingService" ref="authorizingService"/>
            </bean>
        </property>
        <property name="serviceClass"

                  value="de.unternehmertum.playbox.service.remote.MyCityGuideService"/>
        <property name="inHandlers">
            <list>
                <ref bean="addressingHandler"/>
            </list>
        </property>
        <property name="faultHandlers">
            <ref bean="throwableToFaultHandler"/>
        </property>
    </bean
```

CHAPTER 31

MySQL Database Tables

```
--
-- Table structure for table ‘place‘
--
CREATE TABLE ‘place‘ (
  ‘id‘ int NOT NULL auto_increment,
  ‘name‘ varchar(50) default NULL,
  ‘description‘ text default NULL,
  ‘type‘ varchar(50) default NULL,
  ‘category‘ varchar(50) default NULL,
  ‘price‘ varchar(50) default NULL,
  ‘city_district‘ varchar(50) default NULL,
  ‘phone_number‘ varchar(255) default NULL,
  ‘opening_data‘ varchar(255) default NULL,
  ‘address_id‘ int default NULL,
  ‘poll_id‘ int default NULL,
  PRIMARY KEY  (‘id‘),
  KEY (‘place_id‘,‘type‘,‘category‘,‘city_district‘,‘poll_id‘)
) TYPE=MyISAM;

--
-- Dumping data for table ‘place‘
--

--
```

```
-- Table structure for table ‘event‘
--

CREATE TABLE ‘city_event‘ (
  ‘id‘ int NOT NULL auto_increment,
  ‘type‘ varchar(50) default NULL,
  ‘name‘ varchar(50) default NULL,
  ‘description‘ text default NULL,
  ‘price‘ varchar(50) default NULL,
  ‘opening_data‘ date default NULL,
  ‘place_id‘ int default NULL,
  ‘poll_id‘ int default NULL,
  PRIMARY KEY  (‘id‘),
  KEY (‘poll_id‘)
) TYPE=MyISAM;

--
-- Dumping data for table ‘event‘
--

--
-- Table structure for table ‘place_address‘
--

CREATE TABLE ‘place_address‘ (
  ‘id‘ int(11) NOT NULL auto_increment,
  ‘street‘ varchar(255) default NULL,
  ‘city‘ varchar(255) default NULL,
  ‘state‘ varchar(255) default NULL,
  ‘zip‘ varchar(10) default NULL,
  ‘country‘ varchar(255) default NULL,
  ‘longitude‘ double default NULL,
  ‘latitude‘ double default NULL,
  PRIMARY KEY  (‘id‘),
  KEY (‘place_id‘)
) TYPE=MyISAM;

--
-- Dumping data for table ‘user_address‘
--

--
-- Table structure for table ‘poll‘
--
CREATE TABLE ‘poll‘ (
```

```
  `id` int NOT NULL auto_increment,
  `poll_id` int default NULL,
  `value` tinyint default NULL,
  `number_polls` int default NULL,
  PRIMARY KEY  (`id`),
  KEY (`poll_id`)
) TYPE=MyISAM;
--
-- Dumping data for table `poll`
--

--
-- Table structure for table `movie`
--
CREATE TABLE `movie` (
  `id` int NOT NULL auto_increment,
  `title` varchar(255) default NULL,
  `year` tinyint default NULL,
  `description` text default NULL,
  `genre` varchar(50) default NULL,
  `info` varchar(50) default NULL,
  `place_id` int default NULL,
  `poll_id` int default NULL,
  PRIMARY KEY  (`id`),
  KEY (`poll_id`,`place_id`)
) TYPE=MyISAM;
--
-- Dumping data for table `movie`
--
```