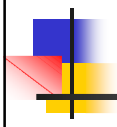


Programowanie w języku Java

Wykład 12: Java Platform, Micro Edition (Java ME)

Konfiguracje

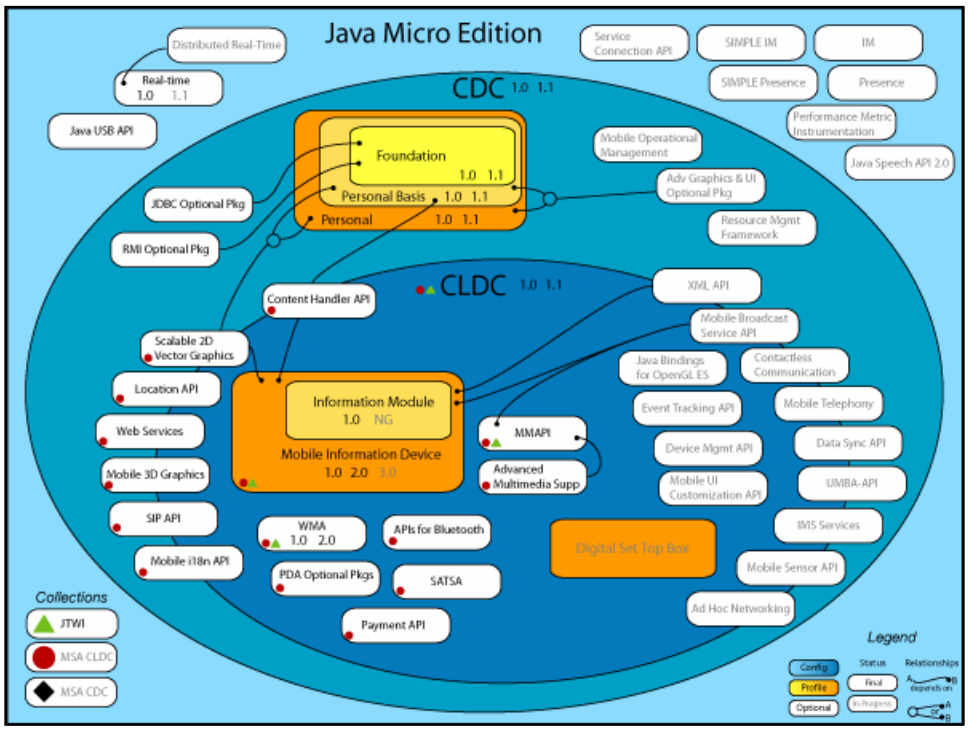
Standard dla pewnych grup urządzeń

- Connected Limited Devices Configuration (CLDC)
  - procesor 16-bitowy, 192kB RAM
  - telefony komórkowe, notesy elektroniczne, itp.
- Connected Device Configuration (CDC)
  - procesor 32-bitowy, 2MB RAM
  - palmtopy, smartfony

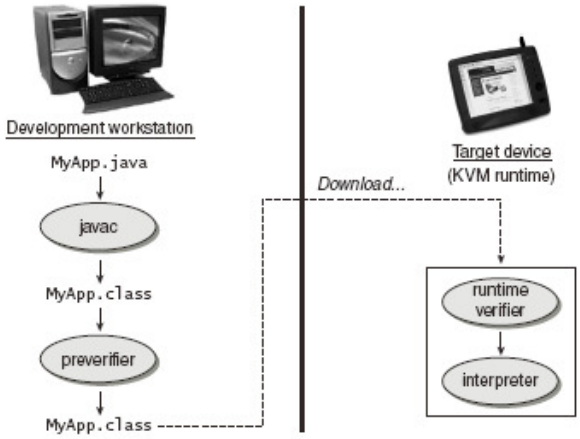
Profile

Standard API dla pewnej grupy urządzeń w ramach danej konfiguracji:

- **MIDP (Mobile Information Device Profile) – CLDC**
  - **uproszczone GUI (dla LCD), midlety**
- **Foundation Profile – CDC**
  - **bez GUI, standardowe klasy Javy**
- **Personal Basis Profile – CDC**
  - **xlet**
- **Personal Profile – CDC**
  - **aplety i AWT**



Tworzenie aplikacji w CLDC



Biblioteki CLDC

- java.lang, java.lang.ref
  - typy, stringi, math, wątki, system
- java.io
  - podstawowe strumienie (brak ObjectStream, Gzip, ZIP)
- java.util
  - calendar, date, proste kontenery
- javax.microedition.io
  - connector

Profil MIDP

CDLC + GUI

Dodatkowe biblioteki:

- javax.microedition.lcdui, javax.microedition.lcdui.game
  - GUI
- javax.microedition.media, javax.microedition.media.control
  - media player
- javax.microedition.midlet
  - środowisko aplikacji

Profil MIDP

midlety rozszerzają klasę MIDlet.

Stany midletu:

- paused: stan początkowy, po wywołaniu stopApp()
- active: po wywołaniu startApp()
- destroyed: po wywołaniu destroyApp()

Przykład 1

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class FirstMIDlet extends MIDlet
      implements CommandListener {

private Command exitCommand;
 private Command infoCommand;
 private Command buyCommand;

private Display display;

 public FirstMIDlet() {
  display = Display.getDisplay(this);
  exitCommand = new Command("Exit", Command.SCREEN, 1);
  infoCommand = new Command("Info",Command.SCREEN, 2);
  buyCommand = new Command("Buy", Command.SCREEN, 2);
 }
```

```
public void startApp() {
 TextBox t = new TextBox ("FirstMIDlet",
  "Welcome to MIDP Programming", 256,
    0);
    t.addCommand(exitCommand);
    t.addCommand(infoCommand);
    t.addCommand(buyCommand);
    t.setCommandListener(this);
    display.setCurrent(t);
  }

public void pauseApp() { }
public void destroyApp(boolean
unconditional) { }

public void commandAction(Command c,
Displayable s) {
   if (c == exitCommand) {
     destroyApp(false);
     notifyDestroyed();
   }
  }
}
```


Przykład 2

```
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class FirstExample extends MIDlet {
   private Display display;
   String url = "http://www.javacourses.com/hello.txt";
   public FirstExample() {
     display = Display.getDisplay(this);
   }

public void startApp() {
   try {
      getViaStreamConnection(url);
   } catch (IOException e) {
        System.out.println("IOException " + e);
        e.printStackTrace();
    }
 }
```

```
public void pauseApp() { }
public void destroyApp(boolean unconditional) { }
void getViaStreamConnection(String url) throws
   IOException {
 StreamConnection c = null;
 InputStream s = null;
 StringBuffer b = new StringBuffer();
 TextBox t = null;
  try {
   c = (StreamConnection)Connector.open(url);
   s = c.openInputStream();
   int ch;
   while((ch = s.read()) != -1) {
       b.append((char) ch);
   }
   System.out.println(b.toString());
   t=new TextBox("hello....", b.toString(),1024,0);
   } finally {
       if(s != null) { s.close(); }
       if(c != null) { c.close();
    }
  }
 display.setCurrent(t);
 }
}
```

Pakiet opcjonalny WMA

```
...
import javax.microedition.io.*;
import javax.wireless.messaging.*;
...
MessageConnection conn = null;
String url = "sms://+417034967891";
try {
   conn = (MessageConnection) Connector.open( url );
   TextMessage msg = conn.newMessage( conn.TEXT_MESSAGE );
   msg.setPayloadText( "Please call me!" );
   conn.send( msg );
}
catch( Exception e ){
   // obsługa błędów
}
finally {
   if( conn != null ){
      try { conn.close(); } catch( Exception e ){}
   }
}
...
```


Pakiet MMAPI

```
...
import java.io.*;
import javax.microedition.media.*;
...
try {
   Player p = Manager.createPlayer(
            "http://somesite.com/music.mp3" );
   p.start();
}
catch( IOException ioe ){
}
catch( MediaException me ){
}
...
```

Konfiguracja CDC

- pełna implementacja Javy i JVM
- pełna zgodność z CLDC
- pakiety opcjonalne:
  - RMI
  - JDBC
  - Web Services
  - AGUI (zaawansowana grafika)
  - Security

Foundation Profile

- dla urządzeń o ograniczonych zasobach, np. drukarki sieciowe, routery,
- bez grafiki, GUI
- wszystkie pakiety Javy (bez AWT) + CDLC

Personal Basis Profile

Foundation Profile +

- ograniczone GUI (AWT bez Button, Panel itp.), JavaBeans, RMI
- xlety: java.microedition.xlet, java.microedition.xlet.ixc,
- przykładowe zastosowania: interaktywna TV, kamery itp.

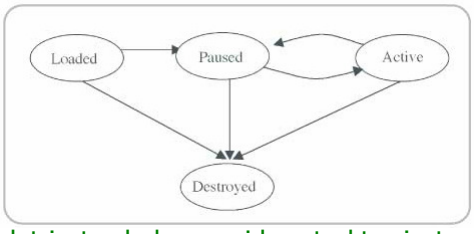
Personal Profile

- pełne GUI (AWT)
- aplety
- zastosowania: zaawansowane PDA, wbudowane przegladarki internetowe

Xlety

```
import javax.microedition.xlet.*;
public class BasicXlet implements Xlet {
 private XletContext context;
 public BasicXlet() { ... }
 public void initXlet( XletContext context )  throws XletStateChangeException {
   this.context = context;
  ...
  }
 public void destroyXlet( boolean unconditional ) throws XletStateChangeException {
  }
  public void pauseXlet(){ ... }
  public void startXlet() throws XletStateChangeException { ... }
}
```


Stany xletu



- loaded: xlet jest załadowany i konstruktor jest wykonany
- paused: xlet zainicjowany (metoda initXlet() jest wykonana) i zatrzymany (metoda pauseXlet()),
- active: xlet działa (aktywowany metodą startXlet())
- destroyed: xlet jest zniszczony (metoda destroyXlet())

Przykład xletu (zegar)

Przykład xletu (zegar), cd.

```
import javax.microedition.xlet.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;

public class ClockXlet implements Xlet,
                ActionListener {
   TextField display;
   MyClock clock;
   Button pauseButton = new Button("Pause");
   Button stopButton = new Button("Stop");
   Button resumeButton = new Button("Resume");
   XletContext context;
   public void initXlet(XletContext ctx)
         throws XletStateChangeException {
      Container c;
      context = ctx;
      try {
         c = ctx.getContainer();
      } catch (UnavailableContainerException e) {
         throw new XletStateChangeException(
                              e.getMessage());
      }
```

```
      display = new TextField(30);
      clock = new MyClock(display);
      pauseButton.addActionListener(this);
      resumeButton.addActionListener(this);
      resumeButton.setEnabled(false);
      stopButton.addActionListener(this);
      c.setSize(200, 200);
      c.setVisible(true);
      c.add(display);
      c.add(pauseButton);
      c.add(resumeButton);
      c.add(stopButton);
      clock.start();
   }

   public void startXlet() {
      clock.setPaused(false);
      resumeButton.setEnabled(false);
      pauseButton.setEnabled(true);
   }
```

Przykład xletu (zegar), cd.

```
public void pauseXlet() {
    clock.setPaused(true);
    resumeButton.setEnabled(true);
    pauseButton.setEnabled(false);
 }
 public void destroyXlet(boolean unconditional) {
    clock.setStopped(true);
  }
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == stopButton) {
       clock.setStopped(true);
       context.notifyDestroyed();
    } else if (e.getSource() == pauseButton) {
       clock.setPaused(true);
       resumeButton.setEnabled(true);
       pauseButton.setEnabled(false);
       context.notifyPaused();
    } else if (e.getSource() == resumeButton) {
       context.resumeRequest();
    }
  }
}
```

```
class MyClock extends Thread {
  boolean paused, stopped;
  TextField display;
 public MyClock(TextField t) {
    display = t;
  }

String getTime() {
    Calendar rightNow = Calendar.getInstance();
    String hour = String.valueOf(rightNow.get(
                          Calendar.HOUR_OF_DAY));
    String min = String.valueOf(rightNow.get(
                          Calendar.MINUTE));
    if (min.length() == 1) {
       min = "0" + min;
    }
    String sec = String.valueOf(rightNow.get(
                          Calendar.SECOND));
    if (sec.length() == 1) {
       sec = "0" + sec;
    }
    return hour + ":" + min + ":" + sec;
  }
```


Przykład xletu (zegar), cd.

```
public synchronized boolean isStopped() {
    return stopped;
  }

  public synchronized void setStopped(boolean
              value) {
    stopped = value;
    notifyAll();
  }

  public synchronized boolean isPaused() {
    return paused;
  }

  public synchronized void setPaused(boolean
              value) {
    paused = value;
    notifyAll();
  }
```

```
  public void run() {
    while (!isStopped()) {
      try {
        if (!isPaused()) {
          Thread.sleep(1);
          display.setText(getTime());
        } else {
          synchronized (this) {
            wait();
          }
        }
      } catch (InterruptedException e) {
      }
    }
  }
}
```

Przykład xletu (komunikacja)

```
import java.rmi.*;
import javax.microedition.xlet.*;
import javax.microedition.xlet.ixc.*;

public class RunOnceXlet extends BasicXlet {
   private static final String NAME =
                "RunOnceXlet.activator";
   private boolean removeBinding = false;
   public RunOnceXlet(){    }

   public void initXlet( XletContext context )
                  throws XletStateChangeException {
      try {
       IxcRegistry registry =
              IxcRegistry.getRegistry( context );
       RemoteInterface remote = new
              RemoteInterfaceImpl();
        while( true ){
           try {
              registry.bind( NAME, remote );
              removeBinding = true;
              break;
           }
```

```
 catch( AlreadyBoundException abe ){
    try {
       remote =(RemoteInterface)
                    registry.lookup( NAME );
       String[] args = (String[])
       context.getXletProperty(
                     XletContext.ARGS );
       remote.activateAgain( args );
       throw new XletStateChangeException(
                     "Already running" );
     }
 catch( NotBoundException nbe ){  }
  }
 }
 }
 catch( RemoteException e ){
        System.out.println( "Registry error:" );
        e.printStackTrace();
     }
     super.initXlet( context );
   }
```

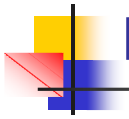

Przykład xletu, cd.

```
public void exit(){
 if( removeBinding ){
    try {
     IxcRegistry registry =
         IxcRegistry.getRegistry( getContext() );
     registry.unbind( NAME );
    }
    catch( NotBoundException e ){    }
        catch( RemoteException e ){   }
 }
 super.exit();
}

 public interface RemoteInterface extends Remote
{
 void activateAgain( String[] args )
                  throws RemoteException;
}
```

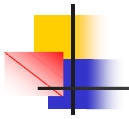
```
private class RemoteInterfaceImpl
             implements RemoteInterface {
 public RemoteInterfaceImpl() throws
                  RemoteException {
}

public void activateAgain( String[] args )
                  throws RemoteException {
System.out.println(
  "[RemoteInterfaceImpl] Activation request" );
   }
 }
}
```



Narzędzia

- Sun Java Wireless Toolkit, NetBeans Mobility Pack
- Nokia Developer's Suite for J2ME , Siemens Wireless Java SDK , Sony Ericsson SDK for the Java(TM) ME Platform
- J2ME Polish



Koniec