Mobile Business Menggunakan Web Service dan J2ME

Robertus Lilik Haryanto
lilik.haryanto@gmail.com
http://secangkirkopipanas.org

Web service merupakan salah satu teknologi yang dikembangkan untuk menjadi tulang punggung arsitektur berbasis layanan (Service Oriented Architecture – SOA). Sebelum teknologi ini dikembangkan, sudah banyak pemrograman berbasis komponen, antara lain COM, CORBA, Java RMI, Microsoft DCOM, dll.

Proses komunikasi yang digunakan pada teknologi web service ini yaitu menggunakan XML seperti WSDL (Web Service Description Language), SOAP (Simple Object Access Protocol), dan UDDI (Universal Description Discovery and Integration).

Pendahuluan

Pada beberapa tahun terakhir ini, banyak sekali aplikasi-aplikasi untuk perangkat mobile yang dirancang untuk melakukan interaksi dengan jaringan internet melalui teknologi GPRS. Penggunaan teknologi GPRS ini semakin marak dengan hadirnya generasi ketiga dari perangkat mobile (baca: 3G), karena dengan menggunakan perangkat mobile yang mendukung generasi ketiga ini, kecepatan untuk interkoneksi melalui GPRS semakin cepat dibandingkan generasi sebelumnya (baca: 2G). Dengan berkembangnya generasi ketiga, diharapkan penggunaan koneksi internet dapat lebih ditingkatkan, sehingga dari sini akan hadir aplikasi-aplikasi untuk perangkat mobile yang terkoneksi dengan jaringan internet.

Selain hadirnya generasi ketiga ini di lingkungan perangkat mobile, teknologi *web service* sudah banyak digunakan dalam pengembangan aplikasi berskala besar (*enterprise*). Dengan penggunaan XML sebagai format data yang ditransmisikan pada teknologi *web service* ini, komunikasi antar *platform* bukan lagi menjadi suatu masalah yang tidak terpecahkan. *Web service* diciptakan untuk mengatasi kelemahan-kelemahan dari beberapa komponen terdahulu yang telah disebutkan di atas. Selain itu, *web service* juga dapat digunakan oleh banyak client,

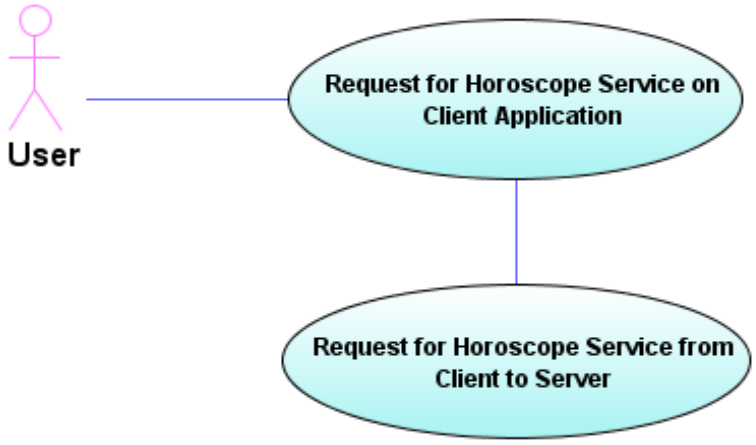
antara lain J2ME.

Dengan menggunakan J2ME MIDP 2.0, perangkat mobile dapat difungsikan sebagai client dari *web service* yang dikembangkan. Sehingga perangkat mobile tidak hanya digunakan sebagai media telekomunikasi, tetapi juga bisa digunakan sebagai media bisnis yang prospeknya sangat terbuka lebar.

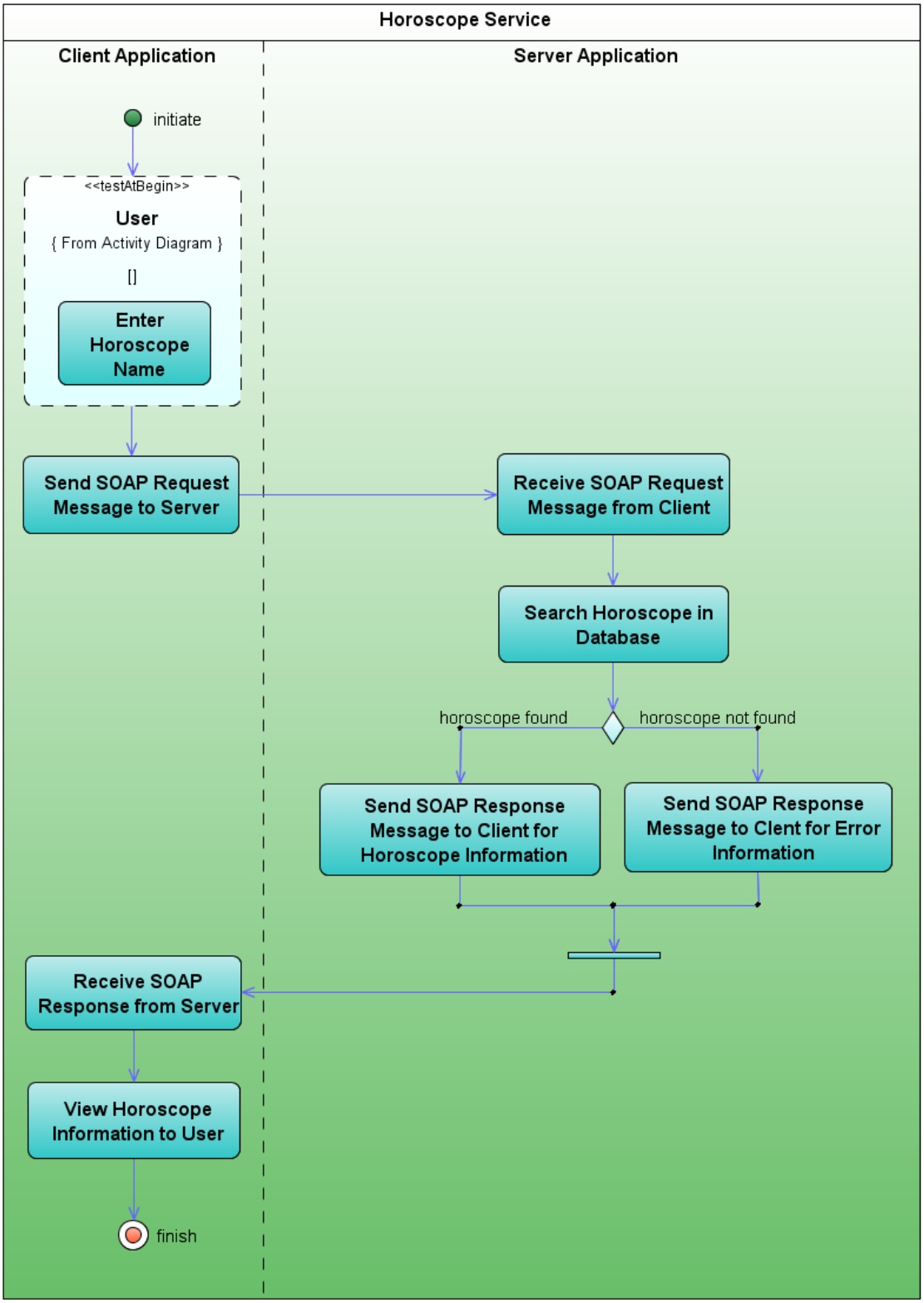
Isi

Aplikasi web service yang akan dikembangkan adalah aplikasi Horoscope yang menyediakan informasi ramalan berdasarkan horoscope yang diminta oleh aplikasi client, dalam hal ini aplikasi J2ME. Berikut pemodelan dari aplikasi yang akan dikembangkan menggunakan UML:

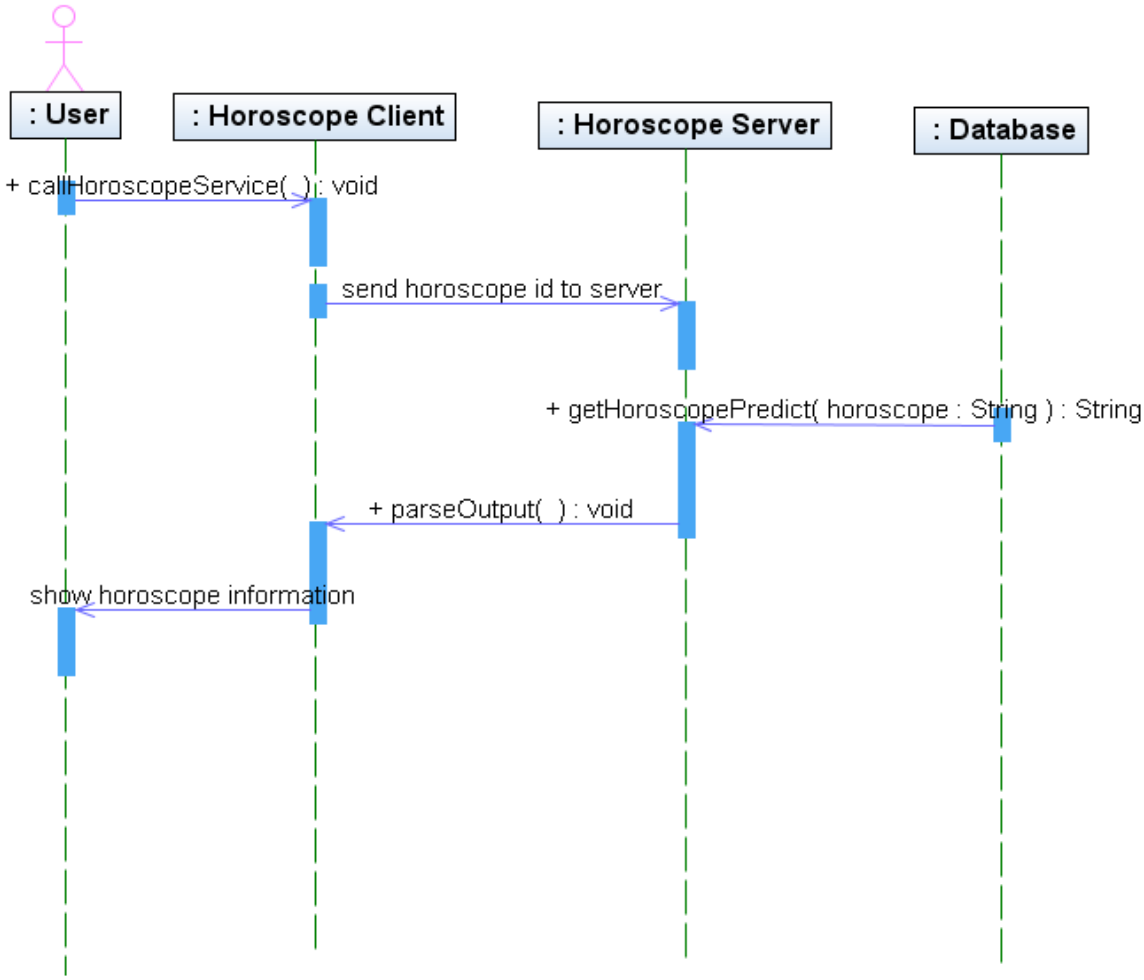
Use Case Diagram



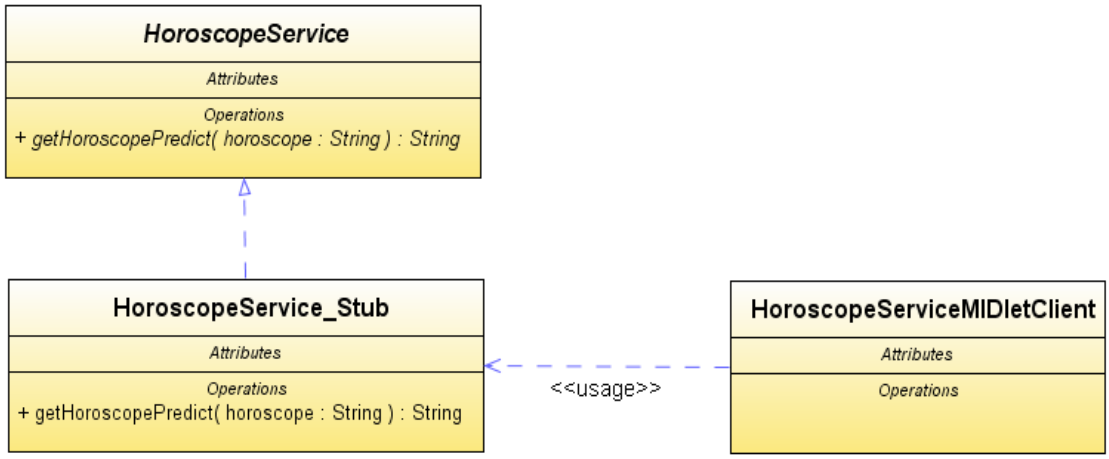
Activity Diagram



Sequence Diagram



Main Class Diagram



Berikut daftar aplikasi-aplikasi pendukung yang digunakan dalam proses pengembangan aplikasi Horoscope ini:

1. Sun JDK 5.0 Update 12
2. Netbeans 5.5.1 + Mobility Pack + Enterprise Pack
3. Jakarta Tomcat 5.5.20
4. JAX-WS 2.0
5. MySQL Server 5.0.33
6. MySQL Connector/J 5.0
7. Sun Java (TM) Wireless Toolkit 2.5.1 for CLDC atau emulator J2ME yang lain
   - Sony Ericsson
   - Nokia
   - Motorola

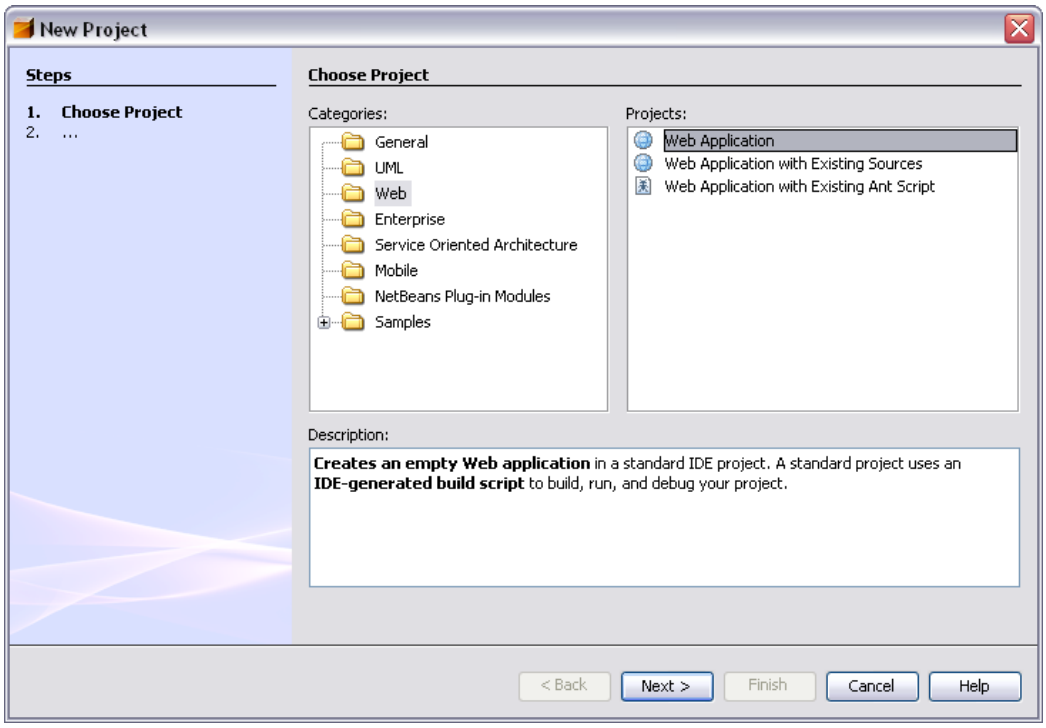
Sebelum Anda mengembangkan aplikasi Horoscope ini, pastikan bahwa aplikasi-aplikasi pendukung di atas telah ter-*install* dengan baik.

Untuk mendukung dalam pengembangan aplikasi ini, saya menyertakan kode program yang ditulis menggunakan **Netbeans 5.5.1**. Berikut langkah-langkah pembuatan aplikasi Horoscope ini:

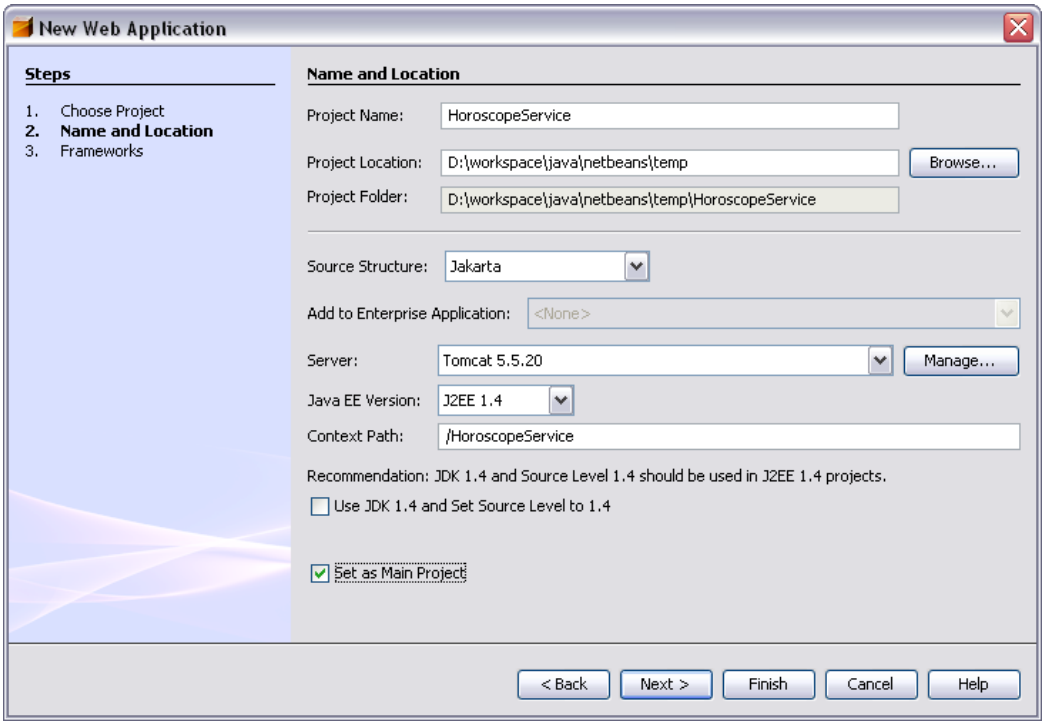
1. HOROSCOPE SERVICE SERVER

a. Sebelum Anda membuat project HoroscopeService berikut, pastikan bahwa library untuk koneksi dengan MySQL sudah ditambahkan pada library dari Tomcat 5.5.20 (`$TOMCAT_HOME\common\lib`).

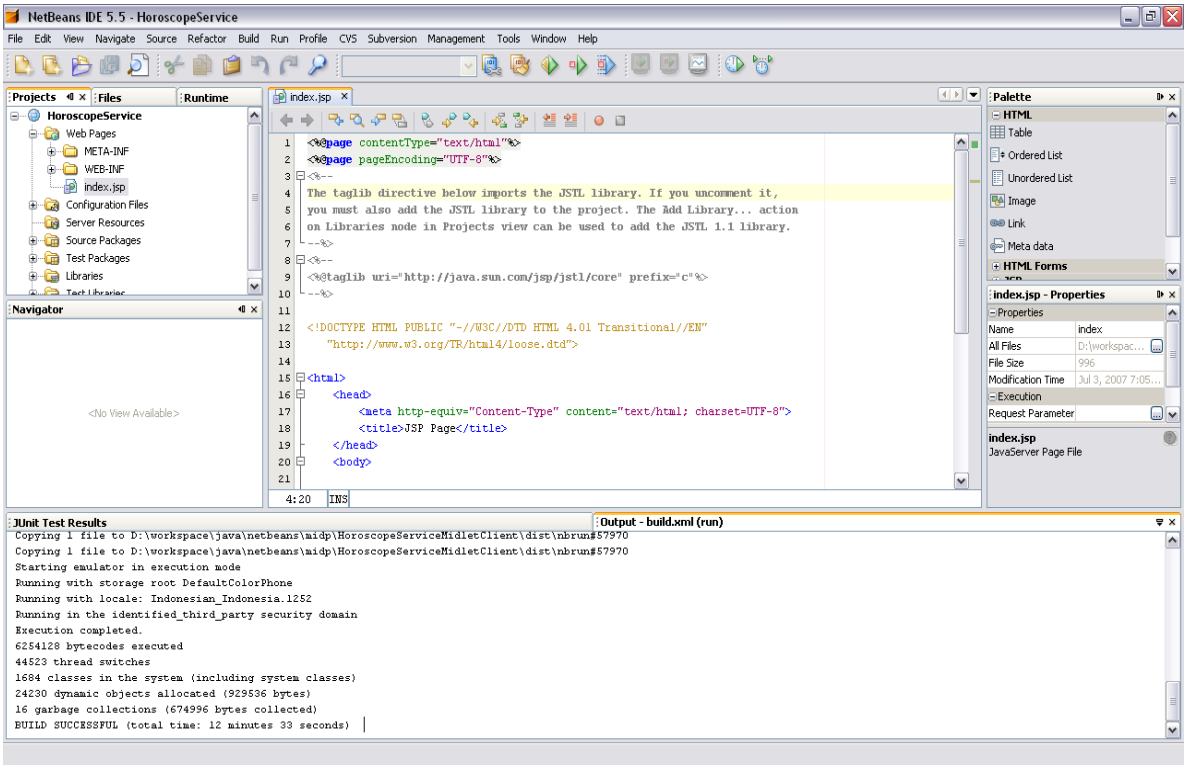
b. Jalankan **Netbeans 5.5.1**, dan buat project baru dengan kategori **Web** -> **Web Application**, kemudian klik tombol **Next**.



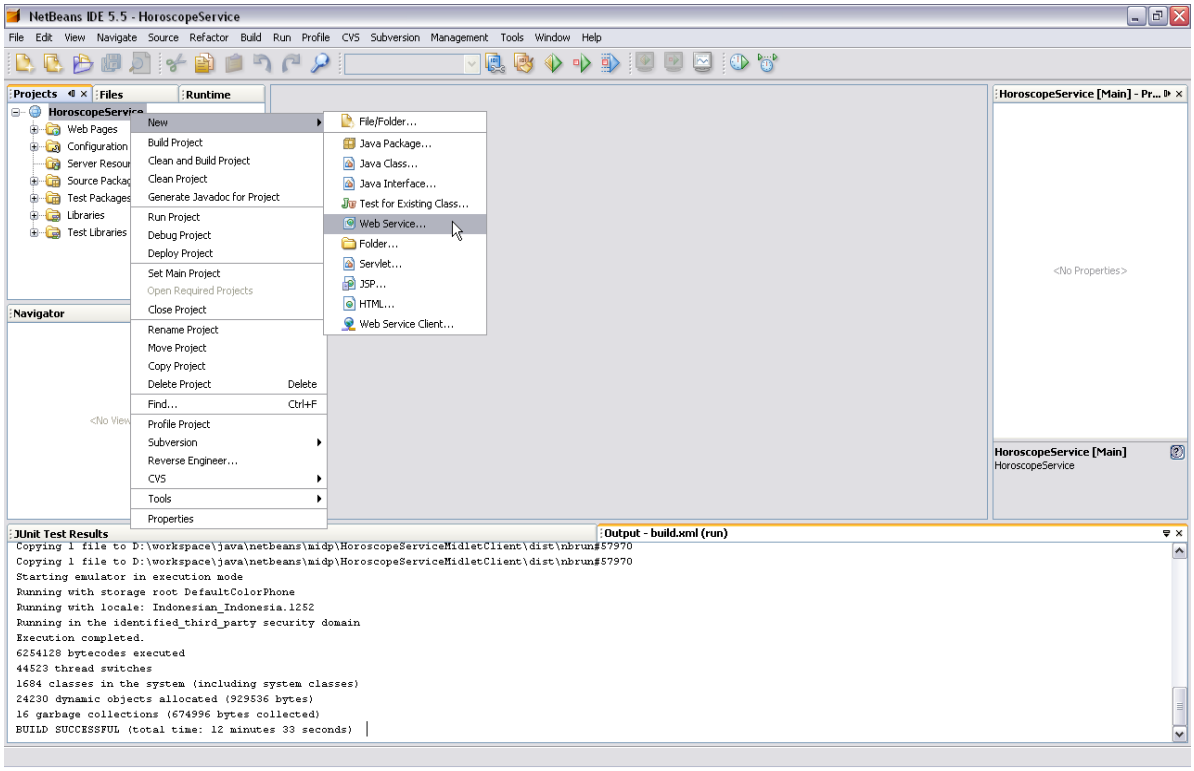
c. Isi beberapa field seperti gambar berikut, kemudian klik tombol **Finish**.



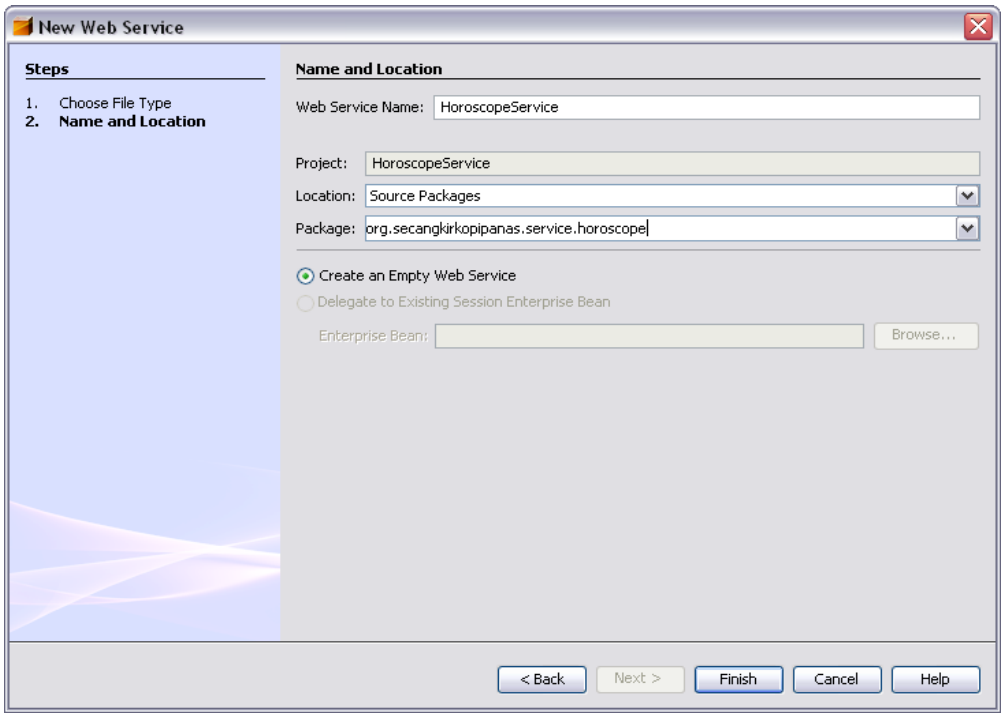
d. Project baru akan terbentuk, seperti tampak pada gambar berikut:



e. Buat Web Service dengan cara klik kanan pada project HoroscopeService, pilih **New -> Web Service**.



f. Isi field-field pada form "New Web Service", sesuai dengan kriteria berikut:
- **Web Service Name**: HoroscopeService
- **Location**: Source Packages
- **Package**: (*disesuaikan kebutuhan*)

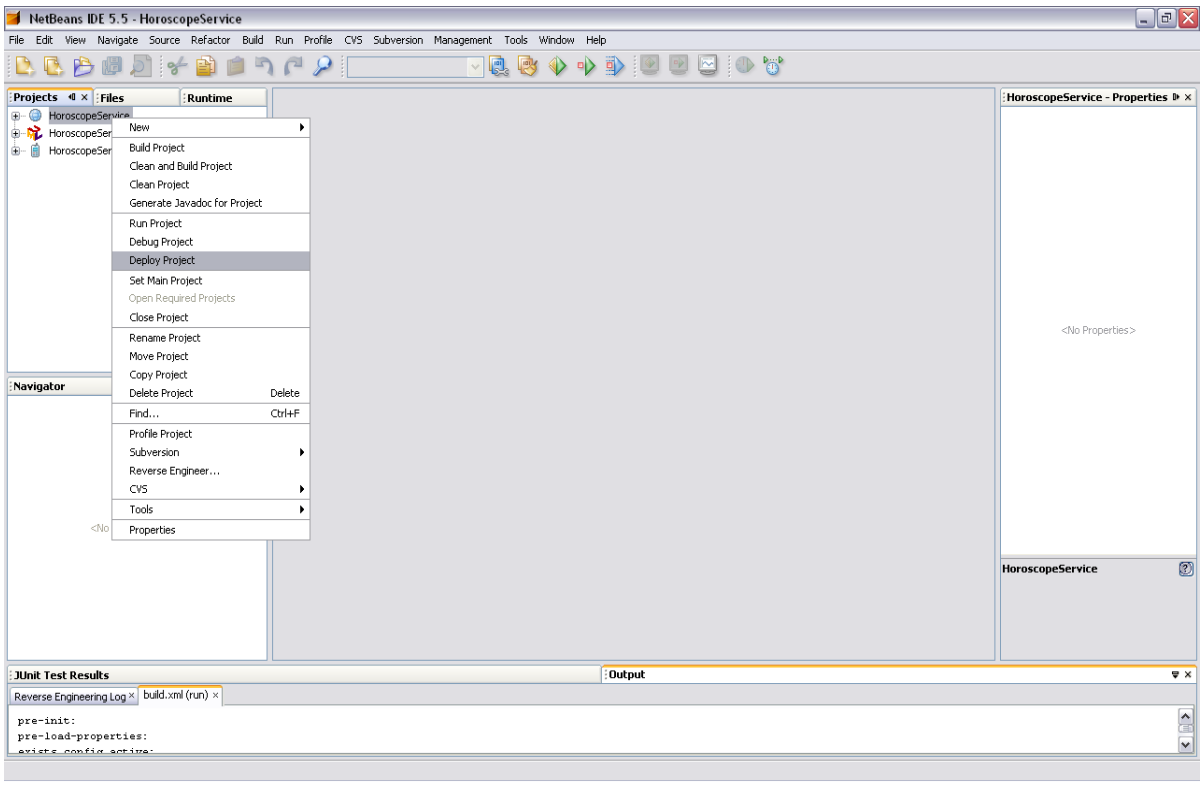


g. Ketikkan kode program berikut pada file **HoroscopeService.java**

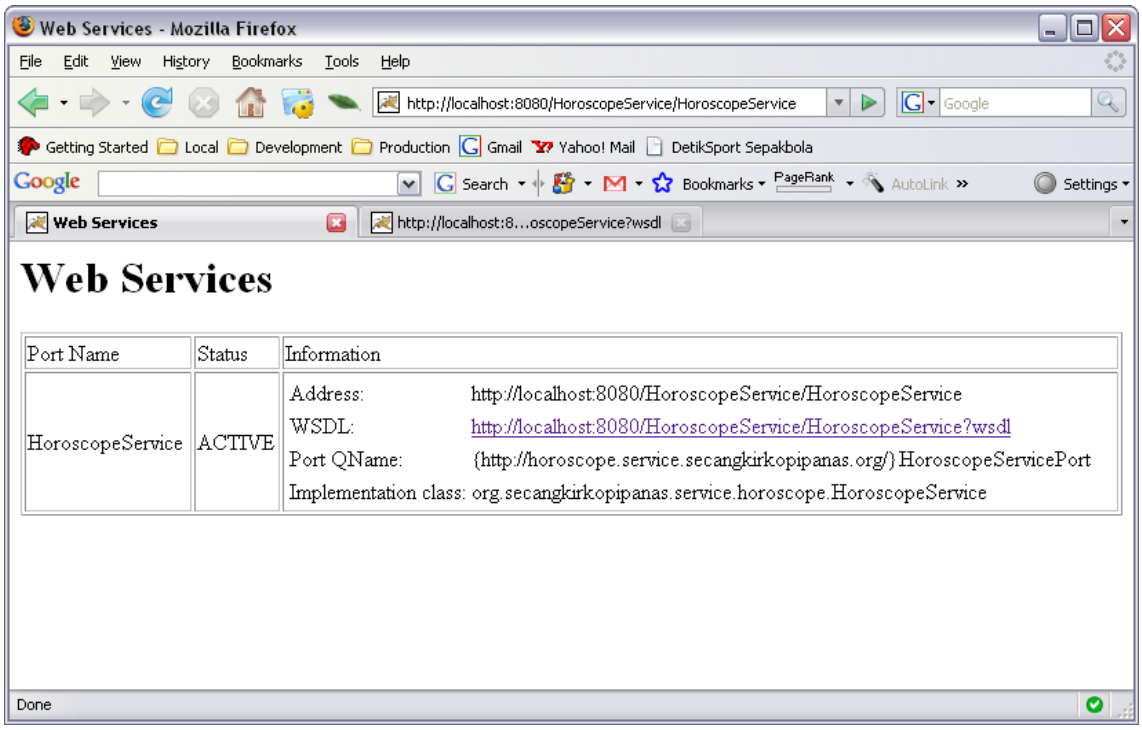
```
/*
 * HoroscopeService.java
 *
 * Created on June 29, 2007, 4:51 PM
 */

package org.secangkirkopipanas.service.horoscope;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLData;
import java.sql.SQLException;
import java.sql.Statement;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.WebParam;

/**
 *
 * @author lilik.haryanto
 */
@WebService()
public class HoroscopeService {

    @WebMethod()
     public String getHoroscopePredict(@WebParam(name = "horoscope")
String horoscope) {
        Connection conn = null;
        ResultSet rs = null;
        Statement stmt = null;
        String id = null;
        String deskripsi = null;
        String tanggal = null;
        String ramalan = null;
        String asmara = null;
        String keuangan = null;
        String kesehatan= null;
        String karir = null;
        String warna = null;

        String output = "";

        try {

            Class.forName("com.mysql.jdbc.Driver");
               String sql = "select id, deskripsi, tanggal, ramalan,
asmara, keuangan, kesehatan, karir, warna from tblhoroscope where id
= '" + horoscope + "'";
                                                                conn     =
DriverManager.getConnection("jdbc:mysql://localhost/horoscope",
"root", "");
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            rs.beforeFirst();

            while (rs.next()) {
                id = rs.getString(1);
                deskripsi = rs.getString(2);
                tanggal = rs.getString(3);
                ramalan = rs.getString(4);
                asmara = rs.getString(5);
                keuangan = rs.getString(6);
                kesehatan = rs.getString(7);
```

```
                karir = rs.getString(8);
                warna = rs.getString(9);

                output = id + "~" + deskripsi + "~" + tanggal + "~" +
ramalan + "~" +
                    asmara + "~" + keuangan + "~" + kesehatan + "~" +
karir + "~" +
                    warna;
            }

            return output;

        } catch (Exception e) {

            return "Error: " + e.getMessage();

        } finally {
            try {
                stmt.close();
                rs.close();
                conn.close();
            } catch (SQLException sqle) {
            }

        }
    }
}
```

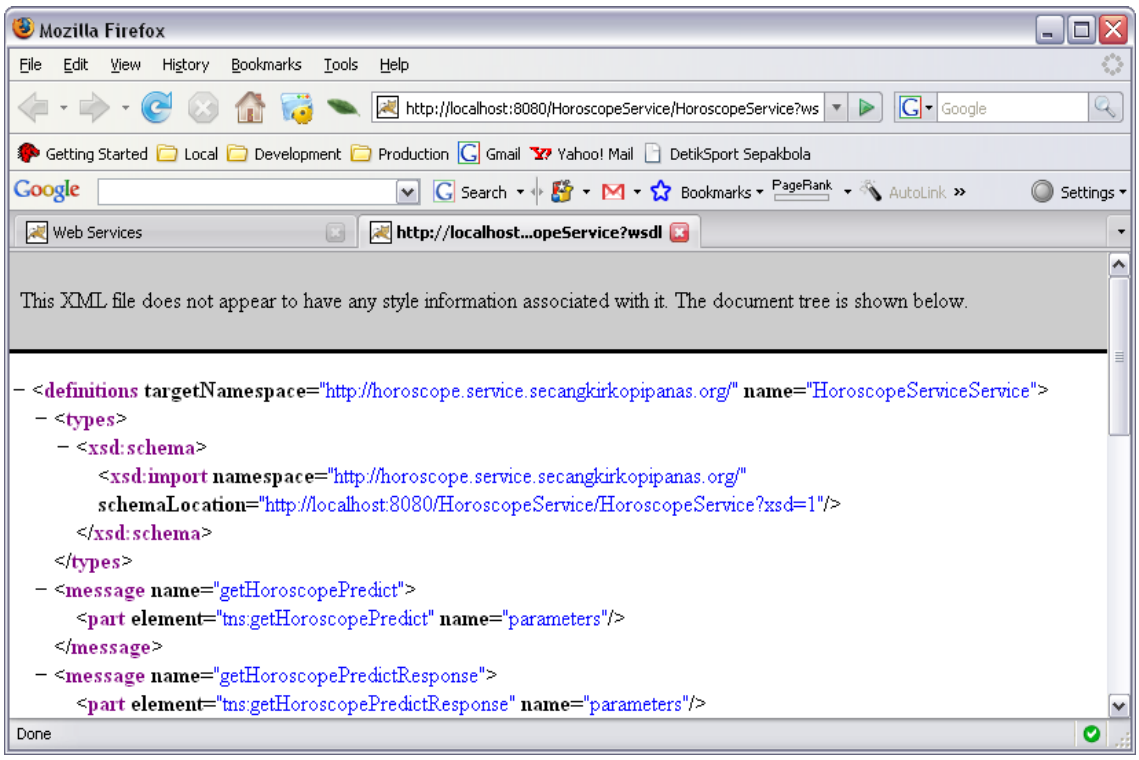
h. Jalankan Tomcat 5.5.20 yang telah terinstall pada komputer Anda.
i. Jika **Netbeans 5.5.1** yang Anda miliki telah terintegrasi dengan Tomcat 5.5.20, lakukan deployment project HoroscopeService dengan cara klik kanan pada project ini di bagian **Project Panel**, pilih menu **Deploy Project**. Atau lakukan deploy secara manual, *copy*-kan file **HoroscopeService.war** yang terbentuk dari proses build project ke direktori webapps Tomcat 5.5.20 (`$TOMCAT_HOME\webapps`).



j. Setelah proses deployment, buka internet browser favorit Anda, untuk menjalankan beberapa URL berikut:
   - http://localhost:8080/HoroscopeService/HoroscopeService



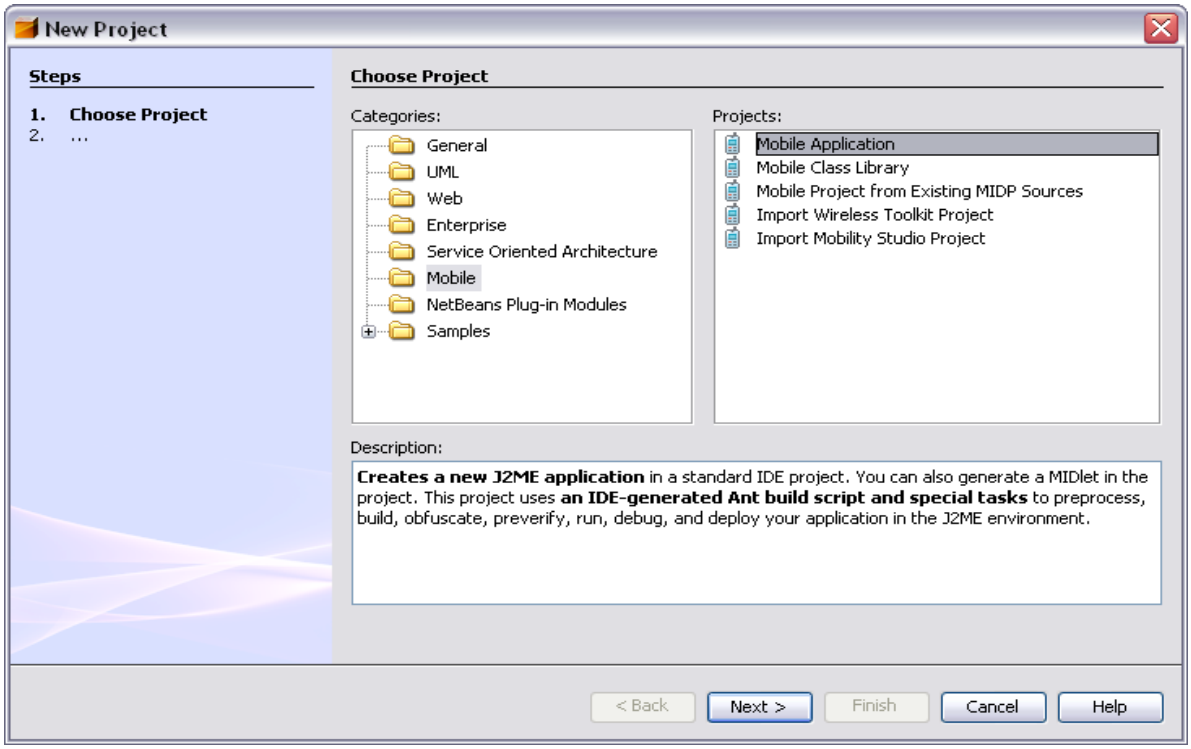
   - http://localhost:8080/HoroscopeService/HoroscopeService?wsdl



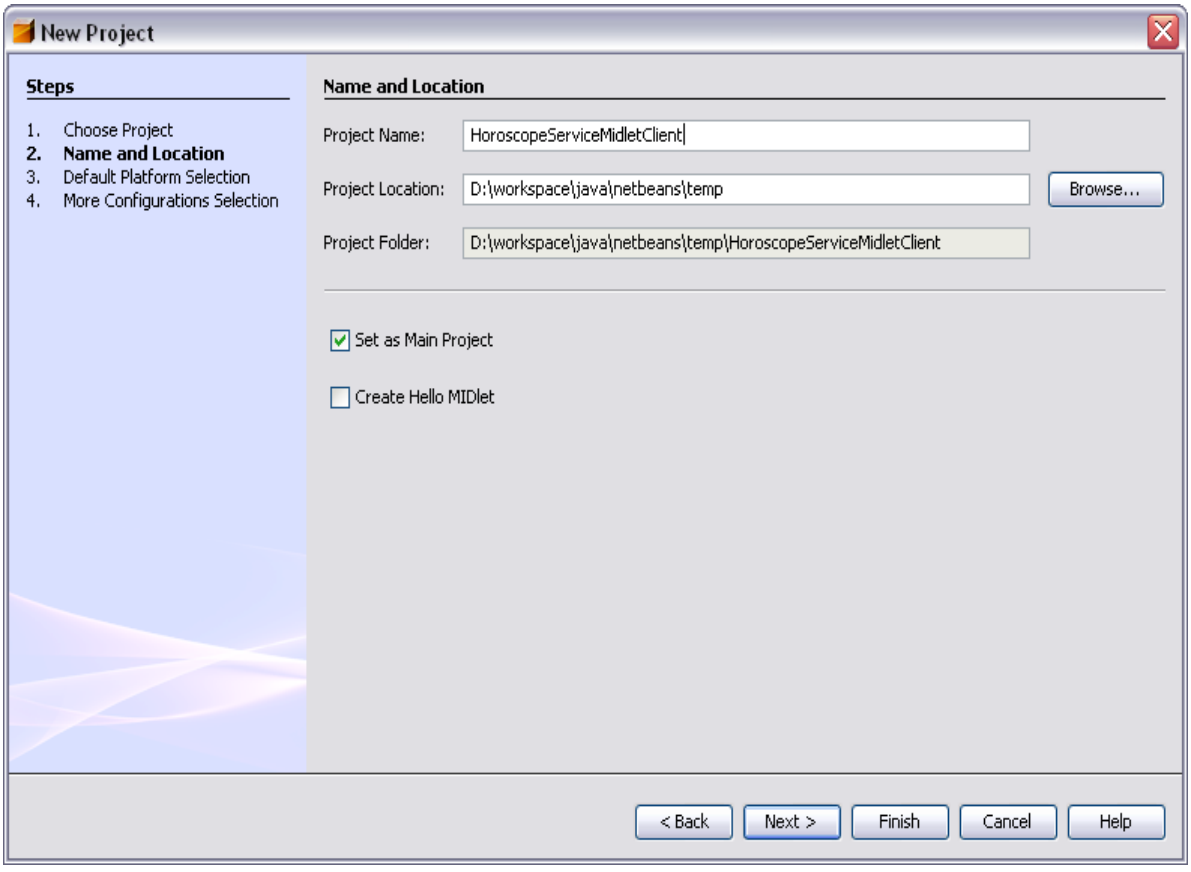
k. Jika Anda melihat internet browser Anda menampilkan informasi seperti pada gambar di atas, berarti proses deployment project HoroscopeService Anda telah berhasil. **Selamat!**

2. HOROSCOPE SERVICE CLIENT

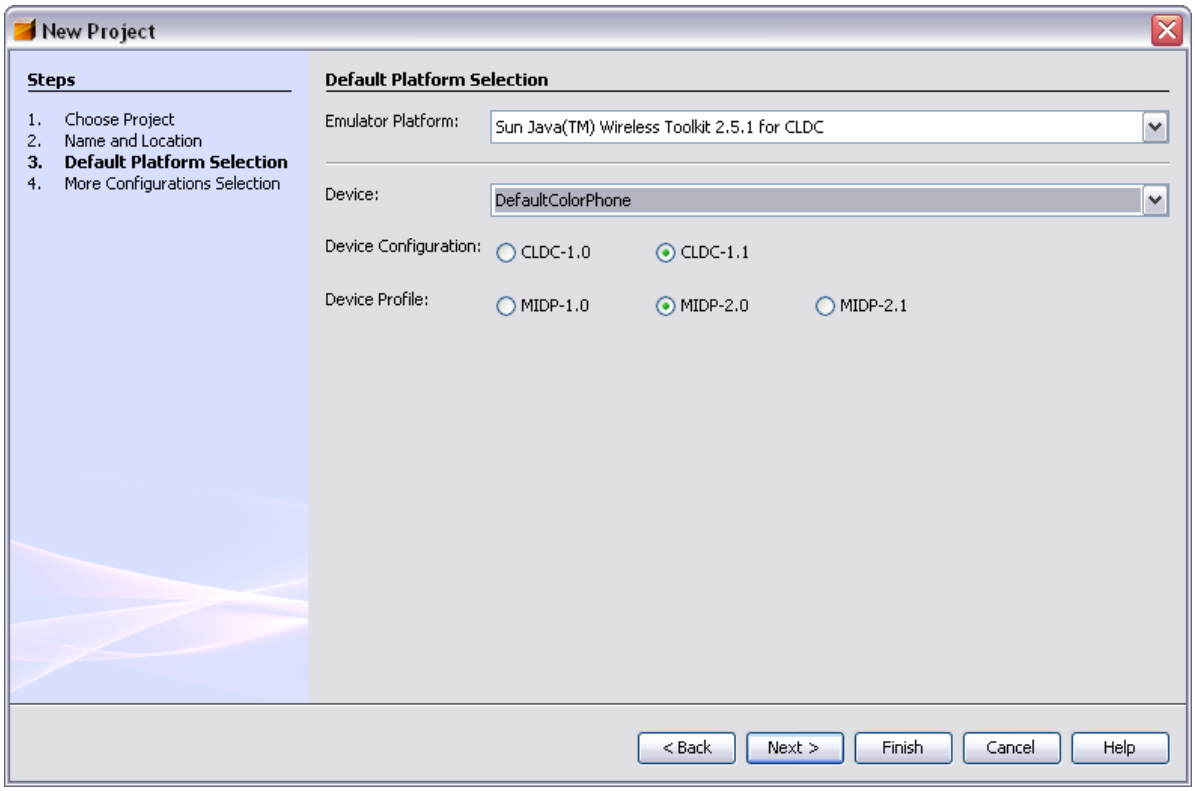
a. Jalankan **Netbeans 5.5.1**, dan buat project baru dengan kategori **Mobile** -> **Mobile Application**, kemudian klik tombol Next.



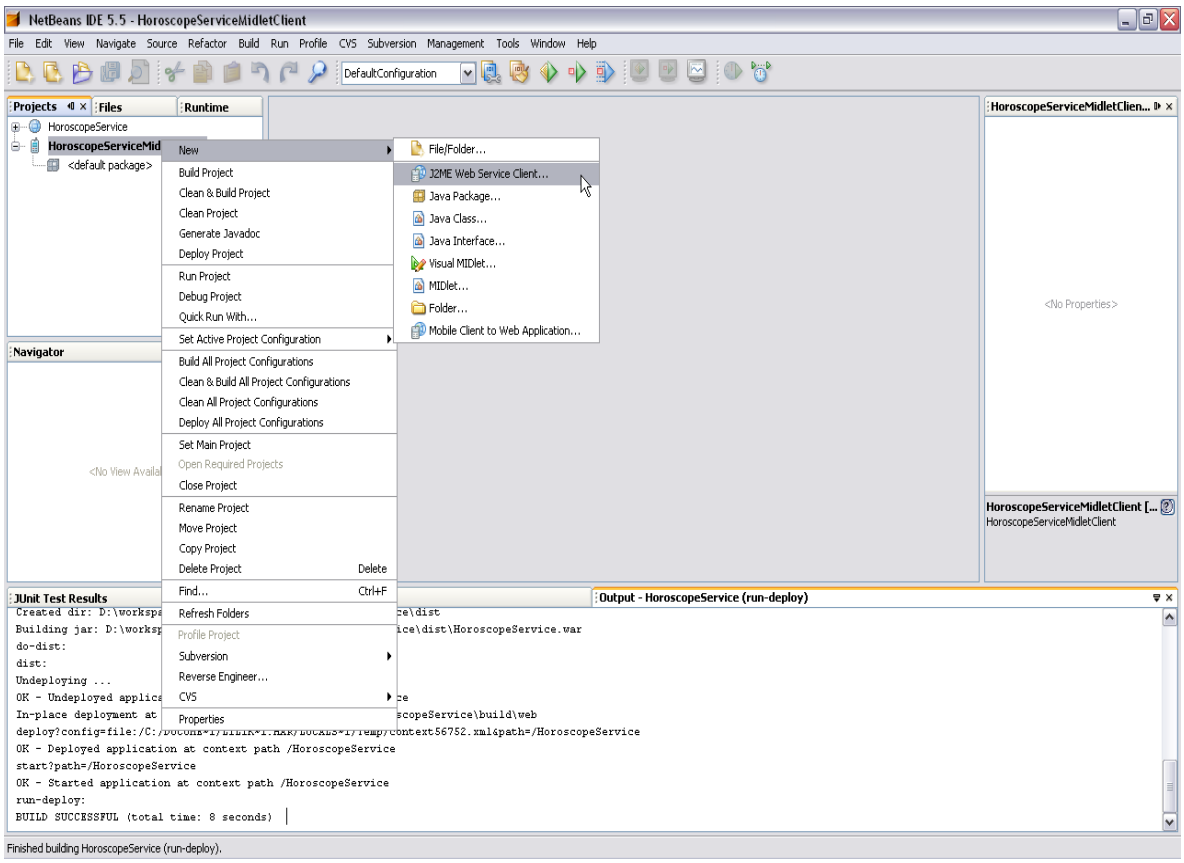
b. Isi beberapa field seperti gambar berikut, kemudian klik tombol **Finish**.



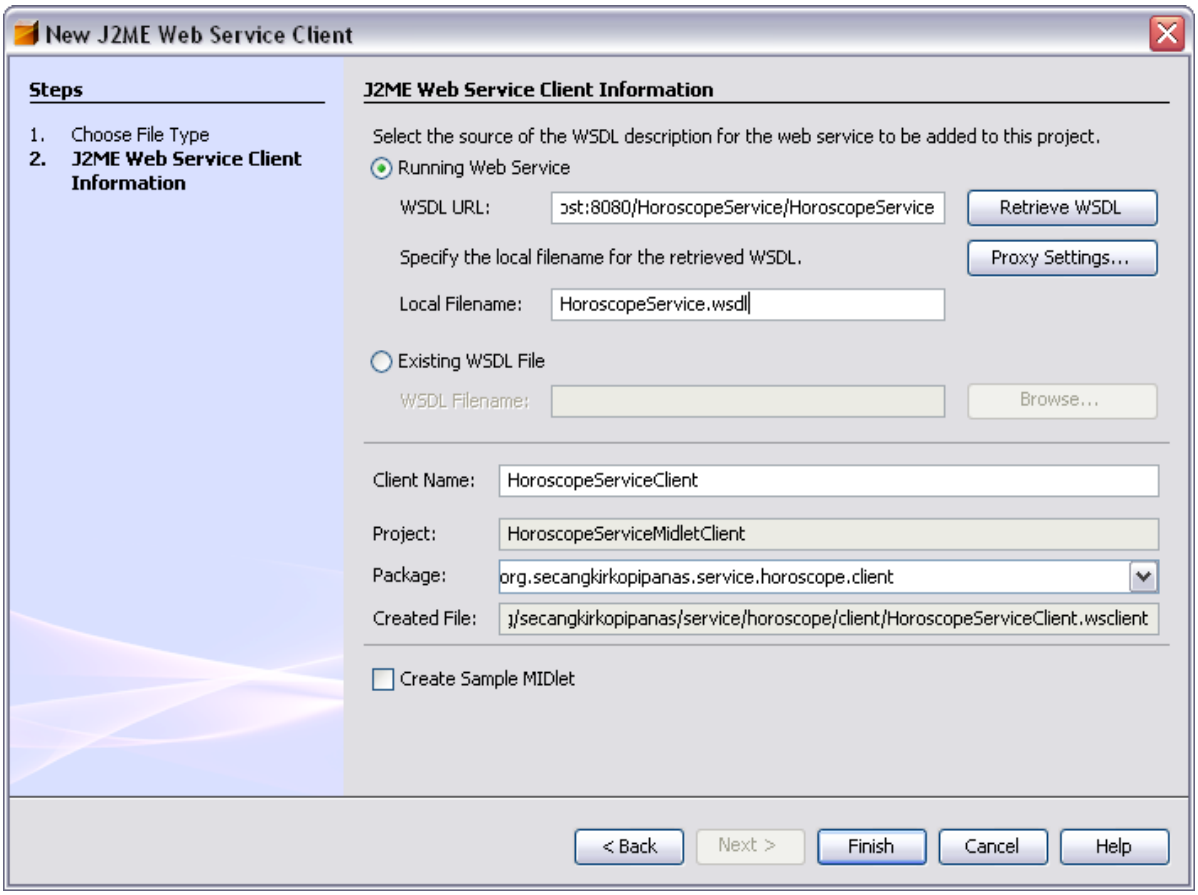
c. Pilih emulator yang ingin digunakan untuk pengetesan project yang akan dibuat, kemudian klik tombol **Finish**.



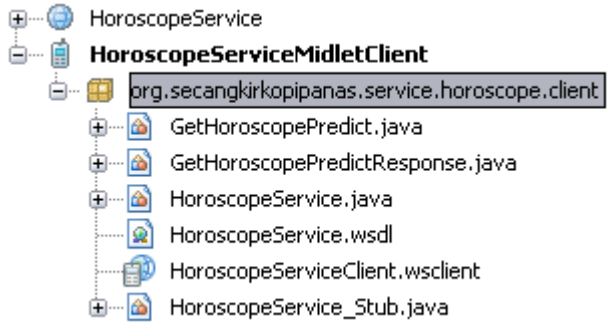
d. Buat J2ME Web Service Client dengan cara klik kanan pada project, pilih **New** ->



J2ME Web Service Client. Kemudian isikan beberapa field yang dibutuhkan seperti tampak pada gambar berikut:

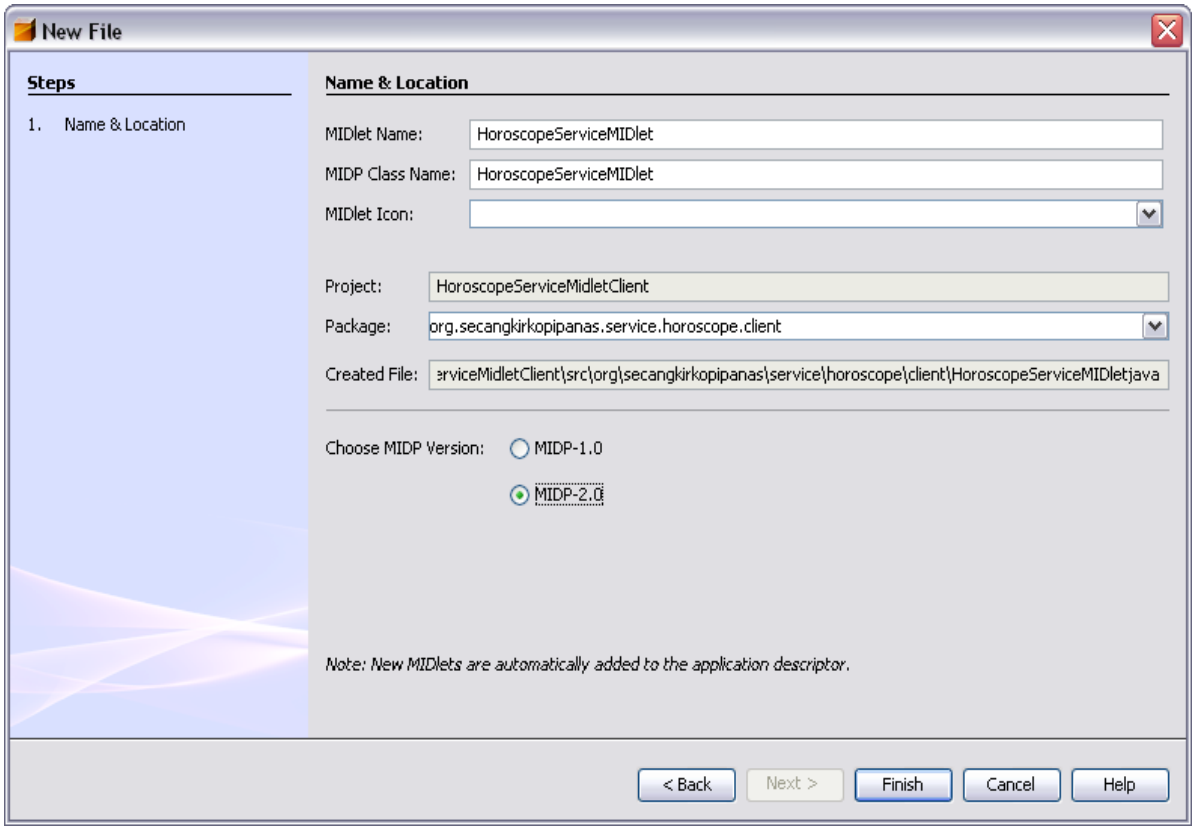


e. Setelah itu, klik tombol **Finish**. J2ME Web Service Client akan di-*generate* oleh NetBeans, sehingga muncul file-file seperti gambar berikut:

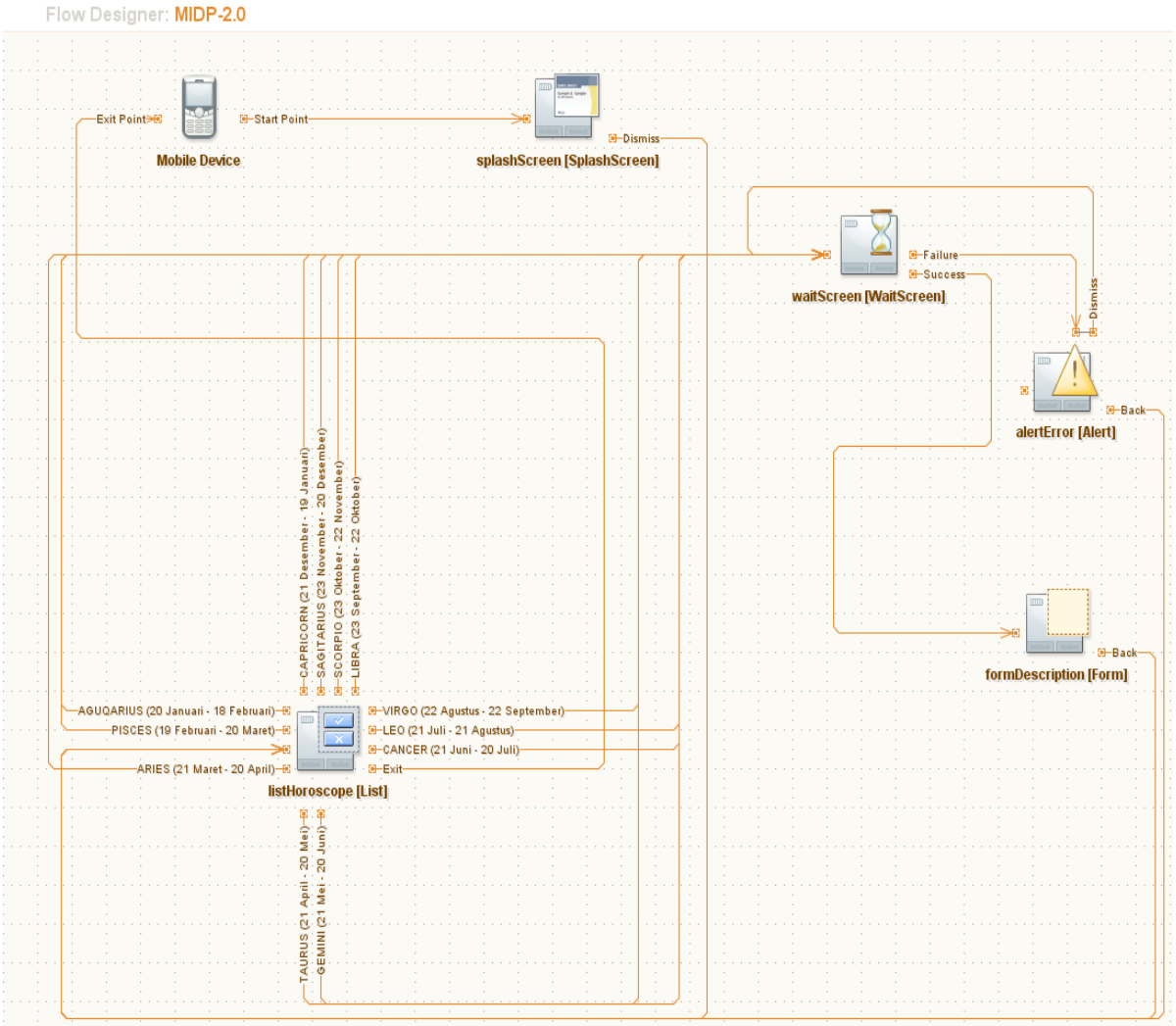


f. File-file yang tampak pada gambar di atas, merupakan file yang akan digunakan oleh file MIDlet yang akan dibuat pada proses setelah ini.

g. Buat Visual MIDlet class dengan cara klik kanan pada package yang ingin ditambahkan Visual MIDlet class, pilih **New** -> **Visual MIDlet**, kemudian isi field-field yang dibutuhkan seperti tampak pada gambar berikut:



h. Berikut flow design dari Visual MIDlet **HoroscopeServiceMIDlet**:



i. Berikut kode program dari file **HoroscopeServiceMIDlet.java**:

```
/*
 * HoroscopeServiceMIDletClient.java
 *
 * Created on June 30, 2007, 7:08 PM
 */

package org.secangkirkopipanas.midlet.service.horoscope;

import java.rmi.RemoteException;
import
org.secangkirkopipanas.midlet.service.horoscope.StringTokenizer;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 * @author lilik.haryanto
 */
public class HoroscopeServiceMIDletClient extends MIDlet implements
CommandListener {

    private String[] horoscope = {
```

```
        "cancer", "leo", "virgo", "libra", "scorpio", "sagitarius",
             "capricorn",  "aquarius",  "pisces",  "aries",  "taurus",
"gemini"
    };

    private HoroscopeService_Stub stub = new HoroscopeService_Stub();

    private String output = "";

    private String input = "";

    private String id = "";

    private String deskripsi = "";

    private String tanggal = "";

    private String ramalan = "";

    private String asmara = "";

    private String keuangan = "";

    private String kesehatan = "";

    private String karir = "";

    private String warna = "";

    /** Creates a new instance of HoroscopeServiceMIDletClient */
    public HoroscopeServiceMIDletClient() {
        initialize();
    }

    private List listHoroscope;
    private Command okCommand;
    private org.netbeans.microedition.lcdui.WaitScreen waitScreen;
    private Form formDescription;
    private Alert alertError;
    private Command backCommandFromAlertError;
              private    org.netbeans.microedition.lcdui.SplashScreen
splashScreen;
    private Image image;
    private Font font;
    private Command backCommandFromDescription;
    private Command exitCommand;
        private  org.netbeans.microedition.util.SimpleCancellableTask
simpleCancellableTask;



     /** Called by the system to indicate that a command has been
invoked on a particular displayable.
     * @param command the Command that ws invoked
      * @param displayable the Displayable on which the command was
invoked
     */
         public  void  commandAction(Command  command,  Displayable
displayable) {
    // Insert global pre-action code here
        if (displayable == listHoroscope) {
            if (command == listHoroscope.SELECT_COMMAND) {
                switch (get_listHoroscope().getSelectedIndex()) {
                    case 0:
                        // Insert pre-action code here
```

```
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 1:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 2:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 3:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 4:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 5:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 6:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 7:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 8:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 9:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
```

```
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 10:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                    case 11:
                        // Insert pre-action code here
                        this.setInput(horoscope[get_listHoroscope().g
etSelectedIndex()]);
                        getDisplay().setCurrent(get_waitScreen());
                        // Insert post-action code here
                        break;
                }
            } else if (command == exitCommand) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        } else if (displayable == waitScreen) {
            if (command == waitScreen.SUCCESS_COMMAND) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_formDescription());
                // Insert post-action code here
                this.parseOutput();
            } else if (command == waitScreen.FAILURE_COMMAND) {
                // Insert pre-action code here
                            getDisplay().setCurrent(get_alertError(),
get_waitScreen());
                // Insert post-action code here
            }
        } else if (displayable == alertError) {
            if (command == backCommandFromAlertError) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_listHoroscope());
                // Insert post-action code here
            }
        } else if (displayable == splashScreen) {
            if (command == splashScreen.DISMISS_COMMAND) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_listHoroscope());
                // Insert post-action code here
            }
        } else if (displayable == formDescription) {
            if (command == backCommandFromDescription) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_listHoroscope());
                // Insert post-action code here
            }
        }
    // Insert global post-action code here
}

    /** This method initializes UI of the application.
     */
    private void initialize() {
        // Insert pre-init code here
        getDisplay().setCurrent(get_splashScreen());
        // Insert post-init code here
    }
```

```
    /**
     * This method should return an instance of the display.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * This method should exit the midlet.
     */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** This method returns instance for listHoroscope component and
should be called instead of accessing listHoroscope field directly.
     * @return Instance for listHoroscope component
     */
    public List get_listHoroscope() {
        if (listHoroscope == null) {
            // Insert pre-init code here
                    listHoroscope = new List("Horoscope Service",
Choice.IMPLICIT, new String[] {
                "CANCER (21 Juni - 20 Juli)",
                "LEO (21 Juli - 21 Agustus)",
                "VIRGO (22 Agustus - 22 September)",
                "LIBRA (23 September - 22 Oktober)",
                "SCORPIO (23 Oktober - 22 November)",
                "SAGITARIUS (23 November - 20 Desember)",
                "CAPRICORN (21 Desember - 19 Januari)",
                "AGUQARIUS (20 Januari - 18 Februari)",
                "PISCES (19 Februari - 20 Maret)",
                "ARIES (21 Maret - 20 April)",
                "TAURUS (21 April - 20 Mei)",
                "GEMINI (21 Mei - 20 Juni)"
            }, new Image[] {
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null
            });
            listHoroscope.addCommand(get_exitCommand());
            listHoroscope.setCommandListener(this);
            listHoroscope.setSelectedFlags(new boolean[] {
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
                false,
```

```
                false
            });
            // Insert post-init code here
        }
        return listHoroscope;
    }

     /** This method returns instance for okCommand component and
should be called instead of accessing okCommand field directly.
     * @return Instance for okCommand component
     */
    public Command get_okCommand() {
        if (okCommand == null) {
            // Insert pre-init code here
            okCommand = new Command("Submit", Command.OK, 1);
            // Insert post-init code here
        }
        return okCommand;
    }

     /** This method returns instance for waitScreen component and
should be called instead of accessing waitScreen field directly.
     * @return Instance for waitScreen component
     */
                public    org.netbeans.microedition.lcdui.WaitScreen
get_waitScreen() {
        if (waitScreen == null) {
            // Insert pre-init code here
                                                waitScreen    =    new
org.netbeans.microedition.lcdui.WaitScreen(getDisplay());
            waitScreen.setCommandListener(this);
            waitScreen.setTitle("Please wait!");
            waitScreen.setFullScreenMode(true);
            waitScreen.setText("Getting data from server...");
            waitScreen.setTask(get_simpleCancellableTask());
            // Insert post-init code here
        }
        return waitScreen;
    }

     /** This method returns instance for formDescription component
and should be called instead of accessing formDescription field
directly.
     * @return Instance for formDescription component
     */
    public Form get_formDescription() {
        if (formDescription == null) {
            // Insert pre-init code here
            formDescription = new Form(null, new Item[0]);
             formDescription.addCommand(get_backCommandFromDescription
());
            formDescription.setCommandListener(this);
            // Insert post-init code here
        }
        return formDescription;
    }

     /** This method returns instance for alertError component and
should be called instead of accessing alertError field directly.
     * @return Instance for alertError component
     */
    public Alert get_alertError() {
        if (alertError == null) {
            // Insert pre-init code here
            alertError = new Alert(null, "<Enter Text>", null, null);
```

```
            alertError.addCommand(get_backCommandFromAlertError());
            alertError.setCommandListener(this);
            alertError.setTimeout(-2);
            // Insert post-init code here
        }
        return alertError;
    }

     /** This method returns instance for backCommandFromAlertError
component   and   should   be   called   instead   of   accessing
backCommandFromAlertError field directly.
     * @return Instance for backCommandFromAlertError component
     */
    public Command get_backCommandFromAlertError() {
        if (backCommandFromAlertError == null) {
            // Insert pre-init code here
                    backCommandFromAlertError  =  new  Command("Back",
Command.BACK, 1);
            // Insert post-init code here
        }
        return backCommandFromAlertError;
    }


     /** This method returns instance for splashScreen component and
should be called instead of accessing splashScreen field directly.
     * @return Instance for splashScreen component
     */
                public    org.netbeans.microedition.lcdui.SplashScreen
get_splashScreen() {
        if (splashScreen == null) {
            // Insert pre-init code here
                                                splashScreen    =    new
org.netbeans.microedition.lcdui.SplashScreen(getDisplay());
            splashScreen.setCommandListener(this);
            splashScreen.setTitle("Horoscope Service Client");
            splashScreen.setFullScreenMode(true);
            splashScreen.setText("");
            splashScreen.setImage(get_image());
            splashScreen.setTextFont(get_font());
            splashScreen.setTimeout(1000);
            // Insert post-init code here
        }
        return splashScreen;
    }

     /** This method returns instance for image component and should
be called instead of accessing image field directly.
     * @return Instance for image component
     */
    public Image get_image() {
        if (image == null) {
            // Insert pre-init code here
            try {
                                                                image    =
Image.createImage("/org/secangkirkopipanas/midlet/service/horoscope/s
ecangkirkopipanas.png");
            } catch (java.io.IOException exception) {
                exception.printStackTrace();
            }
            // Insert post-init code here
        }
        return image;
    }
```

```
    /** This method returns instance for font component and should be
called instead of accessing font field directly.
     * @return Instance for font component
     */
    public Font get_font() {
        if (font == null) {
            // Insert pre-init code here
                        font  =  Font.getFont(Font.FACE_SYSTEM,  0x1,
Font.SIZE_LARGE);
            // Insert post-init code here
        }
        return font;
    }

     /** This method returns instance for backCommandFromDescription
component   and   should   be   called   instead   of   accessing
backCommandFromDescription field directly.
     * @return Instance for backCommandFromDescription component
     */
    public Command get_backCommandFromDescription() {
        if (backCommandFromDescription == null) {
            // Insert pre-init code here
                    backCommandFromDescription  =  new  Command("Back",
Command.BACK, 1);
            // Insert post-init code here
        }
        return backCommandFromDescription;
    }

     /** This method returns instance for exitCommand component and
should be called instead of accessing exitCommand field directly.
     * @return Instance for exitCommand component
     */
    public Command get_exitCommand() {
        if (exitCommand == null) {
            // Insert pre-init code here
            exitCommand = new Command("Exit", Command.EXIT, 1);
            // Insert post-init code here
        }
        return exitCommand;
    }

       /** This method returns instance for simpleCancellableTask
component   and   should   be   called   instead   of   accessing
simpleCancellableTask field directly.
     * @return Instance for simpleCancellableTask component
     */
         public  org.netbeans.microedition.util.SimpleCancellableTask
get_simpleCancellableTask() {
        if (simpleCancellableTask == null) {
            // Insert pre-init code here
            this.output = "";
                                        simpleCancellableTask   =   new
org.netbeans.microedition.util.SimpleCancellableTask();
                                simpleCancellableTask.setExecutable(new
org.netbeans.microedition.util.Executable() {
                public void execute() throws Exception {
                    callHoroscopeService();
                }
            });
            // Insert post-init code here
        }
        return simpleCancellableTask;
    }
```

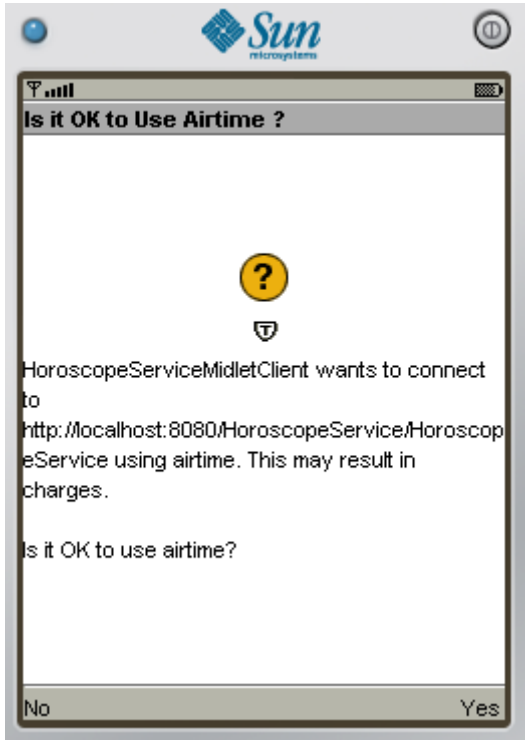
```
    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
        this.notifyDestroyed();
    }

    public void callHoroscopeService() {
        try {
            this.setOutput(stub.getHoroscopePredict(this.getInput()).
trim());
        } catch (RemoteException re) {
            alertError.setString("Error: " + re.getMessage());
        }
    }

    public void resetAll() {
        this.setId("");
        this.setDeskripsi("");
        this.setTanggal("");
        this.setRamalan("");
        this.setAsmara("");
        this.setKeuangan("");
        this.setKesehatan("");
        this.setKarir("");
        this.setWarna("");
    }

    public void parseOutput() {
        formDescription.deleteAll();
        resetAll();
        if (this.output != null) {
            if (!this.getOutput().equals("")) {
                                        StringTokenizer token = new
StringTokenizer(this.getOutput(), "~");
                            if (this.getOutput().indexOf("~") > -1 &&
token.hasMoreTokens() && token.countTokens() == 9) {
                    this.setId(token.nextToken());
                    this.setDeskripsi(token.nextToken());
                    this.setTanggal(token.nextToken());
                    this.setRamalan(token.nextToken());
                    this.setAsmara(token.nextToken());
                    this.setKeuangan(token.nextToken());
                    this.setKesehatan(token.nextToken());
                    this.setKarir(token.nextToken());
                    this.setWarna(token.nextToken());

                                         StringItem siDeskripsi = new
StringItem("Deskripsi: ", this.getDeskripsi());
                     StringItem siTanggal = new StringItem("Tanggal:
", this.getTanggal());
                      StringItem siRamalan = new StringItem("Ramalan:
", this.getRamalan());
                     StringItem siAsmara = new StringItem("Asmara: ",
this.getAsmara());
                    StringItem siKeuangan = new StringItem("Keuangan:
", this.getKeuangan());
                                         StringItem siKesehatan = new
StringItem("Kesehatan: ", this.getKesehatan());
                      StringItem siKarir = new StringItem("Karir: ",
this.getKarir());
                      StringItem siWarna = new StringItem("Warna: ",
```

```
this.getWarna());

                    formDescription.append(siDeskripsi);
                    formDescription.append(siTanggal);
                    formDescription.append(siRamalan);
                    formDescription.append(siAsmara);
                    formDescription.append(siKeuangan);
                    formDescription.append(siKesehatan);
                    formDescription.append(siKarir);
                    formDescription.append(siWarna);

                } else {
                    formDescription.append("Data tidak ditemukan!");
                }
            } else {
                formDescription.append("Data tidak ditemukan!");
            }
        } else {
            formDescription.append("Data tidak ditemukan!");
        }
    }

    public String getOutput() {
        return output;
    }

    public void setOutput(String output) {
        this.output = output;
    }

    public String getInput() {
        return input;
    }

    public void setInput(String input) {
        this.input = input;
    }

    public String getId() {
        return id;
    }

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

    public String getDeskripsi() {
        return deskripsi;
    }

    public void setDeskripsi(String deskripsi) {
        this.deskripsi = deskripsi;
    }

    public String getTanggal() {
        return tanggal;
    }

    public void setTanggal(String tanggal) {
        this.tanggal = tanggal;
    }

    public String getRamalan() {
        return ramalan;
    }
```

```
    public void setRamalan(String ramalan) {
        this.ramalan = ramalan;
    }

    public String getAsmara() {
        return asmara;
    }

    public void setAsmara(String asmara) {
        this.asmara = asmara;
    }

    public String getKeuangan() {
        return keuangan;
    }

    public void setKeuangan(String keuangan) {
        this.keuangan = keuangan;
    }

    public String getKesehatan() {
        return kesehatan;
    }

    public void setKesehatan(String kesehatan) {
        this.kesehatan = kesehatan;
    }

    public String getKarir() {
        return karir;
    }

    public void setKarir(String karir) {
        this.karir = karir;
    }

    public String getWarna() {
        return warna;
    }

    public void setWarna(String warna) {
        this.warna = warna;
    }

}
```

j. Dalam pengembangan project di atas, dibutuhkan class **StringTokenizer**. Anda bisa mendapatkan file source-nya di dalam **src.zip** dari direktori installasi JDK 1.4 atau dalam source file yang disertakan.
k. Compile / build dan jalankan project client ini, dan emulator akan menampilkan output dari aplikasi seperti gambar di bawah.

Output



Penutup

Aplikasi yang telah dikembangkan ini, dapat dimodifikasi sesuai dengan kebutuhan, karena banyak sekali ide-ide yang dapat diterapkan sehingga dapat dijadikan bisnis baru dalam dunia perangkat mobile.

Referensi

1. JAX-WS Documentation
2. Web Services (JAX-WS) in Java EE 5
3. The Daily *Dilbert* Viewer: Creating a Mobile-to-Web Service Application

Biografi Penulis

Robertus Lilik Haryanto. Lahir di Klaten, 2 Oktober 1983. Menyelesaikan pendidikan program S1 pada program studi Teknik Informatika Universitas Sanata Dharma (USD), Yogyakarta pada bulan Agustus 2005. Pernah bekerja di sebuah perusahaan penyedia layanan *web hosting*, PT. Inter Lintas Media Yogyakarta sebagai *web programmer*. Selain itu, pernah mengajar pelatihan Java dan Delphi di Gama Learning Center (GLC), Yogyakarta. Saat ini bekerja sebagai developer di PT. Jati Piranti Solusindo (eCom), Jakarta, sejak tahun 2005, dan juga aktif mengajar Java Programming di Binus Center Kelapa Gading sejak Desember 2006.

Menggeluti dunia pemrograman sejak tahun 1999 dan bahasa pemrograman yang paling digemari adalah Java (J2SE, J2ME, dan J2EE), C++, Microsoft C#.NET dan PHP. Selain melakukan beberapa penelitian, juga sedang mendalami beberapa bahasa pemrograman di atas.