



Anleitung
„Einbindung der Schnittstelle“

für den Business Partner



Version: 4.0 vom 22.08.2007

Zelfi AG
Erthalstraße 1
55118 Mainz
Telefon: 06131 / 9064850
Fax: 06131 / 9064853
eMail: business-partner@zelfi.com

Inhaltsverzeichnis

Inhaltsverzeichnis	2
1. Einleitung	3
2. Kurzanleitung	3
3. Einbinden der Bibliothek	4
3.1 Eclipse	4
3.2 J2ME Polish	5
3.3 Antenna	6
3.4 Wireless Toolkit	6
4. Anpassungen am Quellcode	7
4.1 MIDlet-Klasse	7
4.2 Canvas-Klasse	8
5. Die Konfigurationsdatei key.txt	9
6. Projekt an Zelfi übertragen	9

1. Einleitung

Diese Anleitung ist für Business Partner der Zelfi AG bestimmt. Sie zeigt, mit welchen Verfahren die Zelfi Schnittstelle in ein J2ME Projekt eingebunden wird.

Ab der Version 3.0 gibt es folgende vier Varianten der Bibliothek:

zelfi-lib-generic-midp1.jar	Für alle Geräte die nur MIDP 1.0 unterstützn.
zelfi-lib-generic-midp2.jar	Für alle Geräte die MIDP 2.0 unterstützen.
zelfi-lib-nokia-midp1.jar	MIDP 1.0, benutzt die Klasse FullCanvas von Nokia
zelfi-lib-nokia-midp2.jar	MIDP 2.0, benutzt die Klasse FullCanvas von Nokia

Durch die Verwendung der MIDP 2.0 Variante, können qualitativ bessere Werbungen angezeigt werden. Es ist möglich, Werbeeinblendungen für WAP-Seiten direkt mit einem Link zu verknüpfen. Daraus können entsprechend mehr Einnahmen für den Partner entstehen.

2. Kurzanleitung

Folgende Schritte sind notwendig:

- Neueste Version der Schnittstellen-Bibliothek unter <http://www.zelfi.com/partner/howto> runterladen ([zelfi-lib-v3.2.zip](#)).
- In das Projekt die Bibliotheken von Zelfi einbinden.
- In der MIDlet-Klasse:
 - o Die Klassen `com.zelfi.client.Advertiser` importieren.
 - o Änderung des Konstruktoraufufes der Canvas-Klasse berücksichtigen (siehe unten).
 - o `Advertiser.init(this);` in `startApp()` aufrufen.
 - o `Advertiser.display(this, midlet);` in `startApp()` aufrufen falls beim Starten bereits Werbung kommen soll.
 - o (Ab MIDP 2.0) `Advertiser.showWML(this);` oder `Advertiser.shutdown(this);` in `destroyApp(boolean arg0)` aufrufen.
- In der Canvas-Klasse:
 - o Die Klasse `com.zelfi.client.Advertiser` importieren.
 - o Konstruktor sollte etwa so aussehen: `CanvasClass(MIDletClass midlet)`
 - o An geeigneter Stelle (z.B. nach Beendigung eines Levels) die Funktion `Advertiser.display(this, midlet)` aufrufen.
- Projekt kompilieren und an die Zelfi AG senden.

Hinweis: Ein einfaches Beispiel-Projekt ist unter http://www.zelfi.com/fileadmin/downloads/partner/example_workspace.zip verfügbar.

3. Einbinden der Bibliothek

Eclipse

Um die Bibliotheken in die Entwicklungsumgebung Eclipse einzubinden, muss nur der **Java Build Path** angepasst werden.

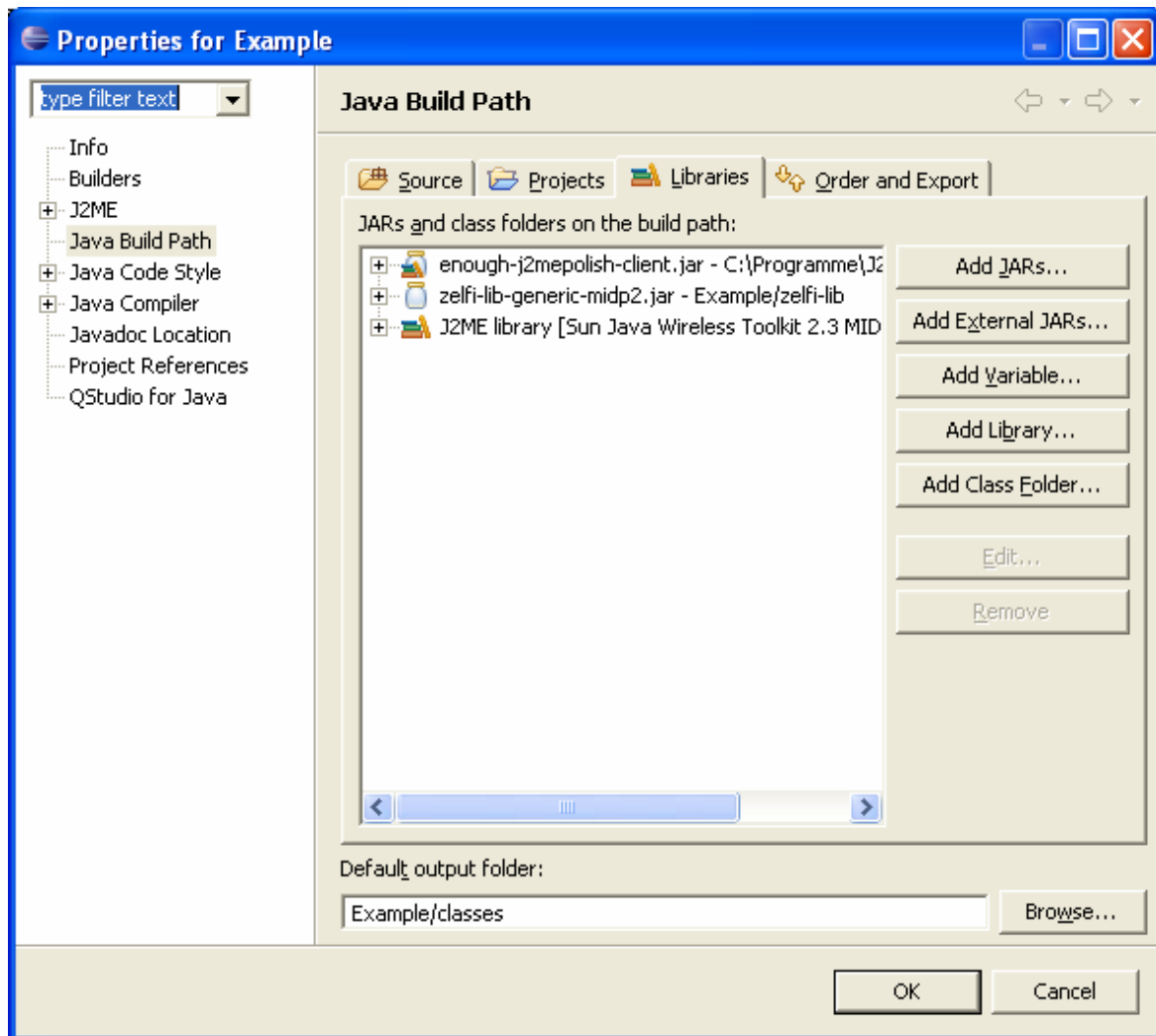


Abbildung 1: Java Build Path anpassen, Schritt 1.

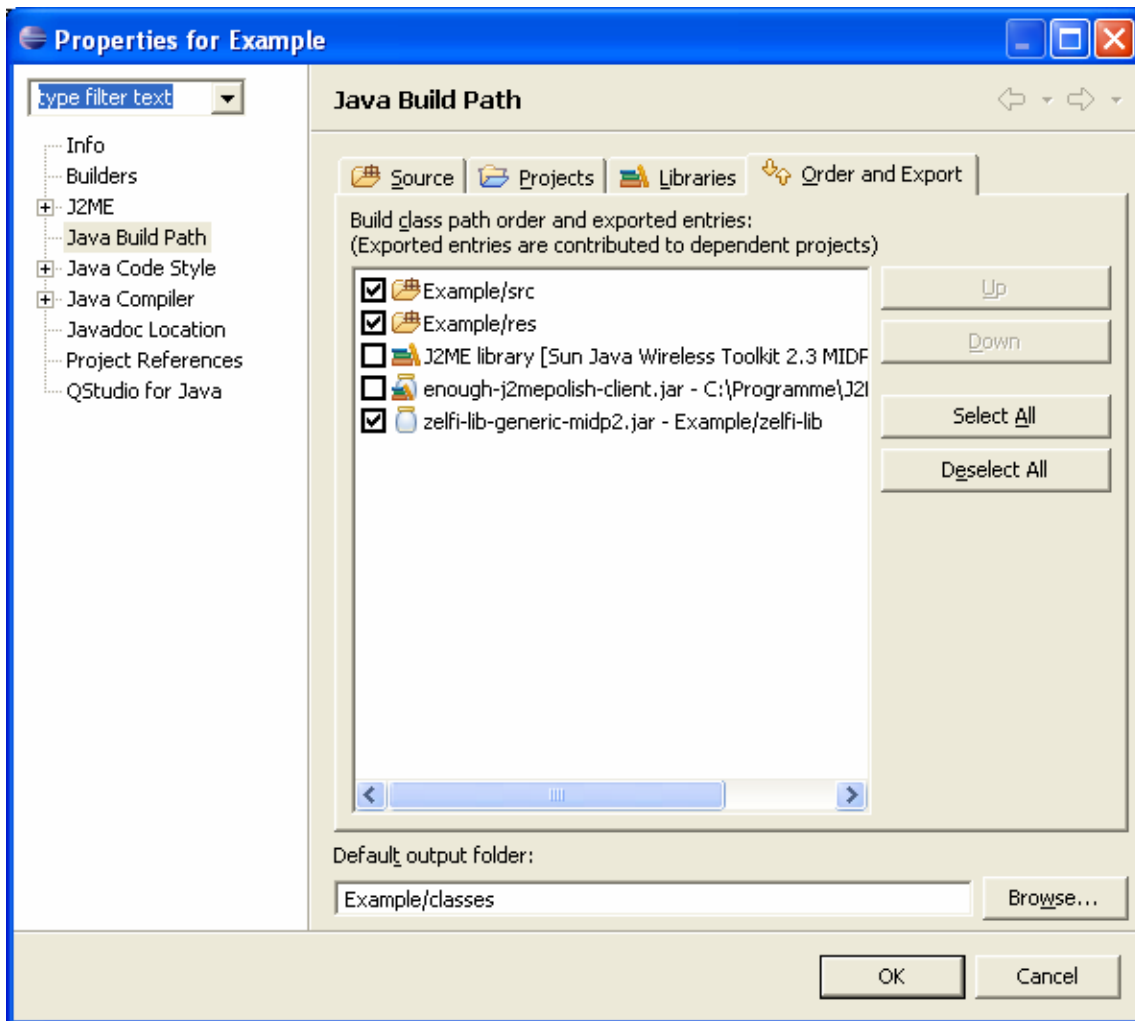


Abbildung 2: Java Build Path anpassen, Schritt 2.

Weite Infos über Eclipse: <http://www.eclipse.org/>

J2ME Polish

Falls J2ME Polish als Build Werkzeug verwendet wird, sind folgende Anpassungen am Skript vorzunehmen.

```
<property name="ZELFI-GENERIC-MIDP1.lib" value="/zelfi-lib/zelfi-lib-generic-midp1.jar"/>
```

```
<build
  usePolishGui="false"
  workDir="${dir.work}"
  binaryLibrary = "${ ZELFI-GENERIC-MIDP1.lib} "
  Symbols="polish"
>
```

Weiter Infos über J2ME Polish: <http://www.j2mepolish.org/>

Antenna

Das Build Werkzeug Antenna benötigt die unten dargelegten Anpassungen.

```
<property name="ZELFI-GENERIC-MIDP1.lib" value=""/zelfi-  
lib/zelfi-lib-generic-midp1.jar"/>  
  
<path id="bootclasspath">  
  <pathelement path="{MIDP.lib}"/>  
  <pathelement path="{CLDC.lib}"/>  
  <pathelement path="{ZELFI-GENERIC-MIDP1.lib}"/>  
</path>  
  
<!-- Package everything. -->  
  <wtkpackage jarfile="{midlet.name}.jar" profile="MIDP-  
1.0" config="CLDC-1.0" jadfile= "{midlet.name}.jad"  
obfusate="true" >  
  <fileset dir="classes"/>  
  <fileset dir="res"/>  
  <bootclasspath path="{MIDP.lib}"/>  
  <bootclasspath path="{CLDC.lib}"/>  
  <libclasspath path="{ZELFI-GENERIC-MIDP1.lib}"/>  
</wtkpackage>
```

Weiter Infos über Antenna: <http://antenna.sourceforge.net/>

Wireless Toolkit

Das J2ME Wireless Toolkit ist eine Plattform mit Emulatoren Umgebung für Entwickler.

Einfach die Zelfi-Bibliothek in den „lib“-Ordner des Projektes kopieren und die Bibliothek wird automatisch vom WTK eingebunden.

Weiter Infos über WTK: http://java.sun.com/products/sjwtoolkit/download-2_3.html

4. Anpassungen am Quellcode

MIDlet-Klasse

Folgende Klasse muss in der MIDlet-Klasse importiert werden.

Import in MIDlet-Klasse:

```
import com.zelfi.client.Advertiser;
```

In der startApp()-Funktion der MIDlet-Klasse wird die statische Funktion `Advertiser.init` aufgerufen, die die Zelfi-Schnittstelle initialisiert.

Aufruf der `init()`-Funktion:

```
Advertiser.init(this);
```

Neu ab Version 3.0: Ist eine Werbeeinblendung direkt beim Starten der Anwendung vorgesehen, so kann in der startApp()-Funktion folgender Aufruf stehen:

Anzeigen der Werbung beim Start definieren:

```
Advertiser.display(this, midlet, true, true);
```

Ab MIDP 2.0 kann der Konsument auf WAP-Seiten verwiesen werden. Dabei hat er die Möglichkeit die WAP-Seite später anzuschauen, um das Spiel nicht zu unterbrechen. Beim Beenden der Anwendung wird er noch mal gefragt. In der destroyApp() muss folgender Aufruf stehen.

Anzeigen der im RMS gespeicherten WAP-Seiten beim Beenden der Anwendung:

```
Advertiser.showWML(this);
```

Neu ab Version 4.0: Beim Beenden der Anwendung kann eine Übersichtseite der noch zu besuchenden Wap-Seiten angezeigt werden, oder falls keine Wap-Seiten abgespeichert wurden, wird eine weitere Werbung angezeigt. Diese Funktion darf aber nur in der destroyApp() des MIDlets stehen.

Anzeigen der im RMS gespeicherten WAP-Seiten oder, falls keine Wap-Seiten gespeichert sind, eine weitere Werbung der Anwendung:

```
Advertiser.shutdown(this);
```

Canvas-Klasse

Um den Zugriff auf die Funktionen von Advertiser zu ermöglichen, muss die Advertiser-Klasse importiert werden.

Import in Canvas-Klasse:

```
import com.zelfi.client.Advertiser;
```

Konstruktor:

```
MIDletClass midlet;  
  
public CanvasClass(MIDletClass midlet) {  
    this.midlet=midlet;  
    ...  
}
```

Das Objekt der MIDlet-Klasse muss als Instanzvariable gespeichert werden, um es später an die Zelfi-Schnittstelle übermitteln zu können.

Als Letztes benötigt die Klasse die Funktion `display()` vom Advertiser, die einen Thread anstößt, der auf das Signal zum Anzeigen der Werbung wartet. Nach dem Aufruf wird geprüft, ob die vorgegebene Zeit bis zur nächsten Werbeeinblendung bereits verstrichen ist.

Anzeigemöglichkeit der Werbung definieren:

```
Advertiser.display(this,midlet);
```

Wichtig: Nicht jedes Mal wird nach Aufruf von `Advertiser.display()` eine Werbebotschaft angezeigt, sondern nur dann, wenn die vorgegebene Zeitspanne bis zum Aufruf einer Werbebotschaft erreicht ist (z.B. Die erste Werbebotschaft erfolgt nach 1 Minute, jede weitere nach 5 Minuten). Der Programmierer kann Zelfi ein geeignetes Intervall der Spielunterbrechungen durch die Werbebotschaften vorschlagen.

Um den Spielfluß nicht zu beeinträchtigen, kann der Programmier die exakte Stelle des Aufrufs vorgeben.

Threads, die wichtig für das Spiel sind, sollten pausiert werden, bevor `Advertiser.display()` aufgerufen wird, um den Spielablauf nicht durcheinander zu bringen.

Neu: Die ab der Version 3.0 verfügbare **optionale** Funktion, besitzt zwei weitere Parameter (*wait* und *internet*).

```
Advertiser.display(this,midlet,false,true);
```

wait

- Vom Typ Boolean (true oder false)
- Gibt an, ob die Funktion erst beendet werden soll, wenn die Werbung angezeigt wurde.
- **Vorsicht! Niemals als Parameter „true“ in Funktionen paint(), keyPressed(), keyReleased und commandAction() verwenden.** Das blockiert die Anwendung!

Internet

- Vom Typ Boolean (true oder false)
- **„false“ nur zum Testen oder Debuggen geeignet!**
- Gibt an, ob die Werbung aus dem Internet oder statische geladen werden soll.
- **Statische Werbung erzeugt keine Werbung, die abgerechnet werden kann.**

5. Die Konfigurationsdatei key.txt

Die Datei key.txt aus dem JAR der Zelfi Bibliothek kann verändert werden, um die Wartezeiten zwischen den Werbeeinblendungen festzulegen.

- Die erste Zeile ist die Adresse zum Server, wo die Werbung runtergeladen wird. Diese Zeile sollte nicht geändert werden.
- Die zweite Zeile gibt die Wartezeit bis zur ersten Werbeeinblendung an. Hier kann eine Wartezeit in ms zwischen 0 und 60000 gewählt werden.
- Die dritte Zeile, gibt die Wartezeit für weitere Werbeeinblendungen und kann zwischen 0 ms und 300000 ms eingeplant werden.

```
http://gtdcjzt.zelfi.com/04/pa?id=v2hy07n5&game=lib_4 //Adresse des Servers (nicht ändern)
0 //Wartezeit, bis zur ersten Werbeeinblendung in ms. (60000 = 1 Min)
0 //Wartezeit, für weitere Werbeeinblendungen in ms. (300000 = 5 Min)
```

Inhalt der Datei key.txt: Nur fett Markierte Werte dürfen geändert werden.

6. Projekt an Zelfi übertragen

Sende an Zelfi:

- JAR-Datei(en)
- JAD-Datei(en)
- kurze Spielbeschreibung, Screenshots, Liste der getesteten Handys
- Angabe der Punkte, in denen `Advertiser.display` aufgerufen wird
- Empfehlung für Wartezeiten.

Kontakt: business-partner@zelfi.com