

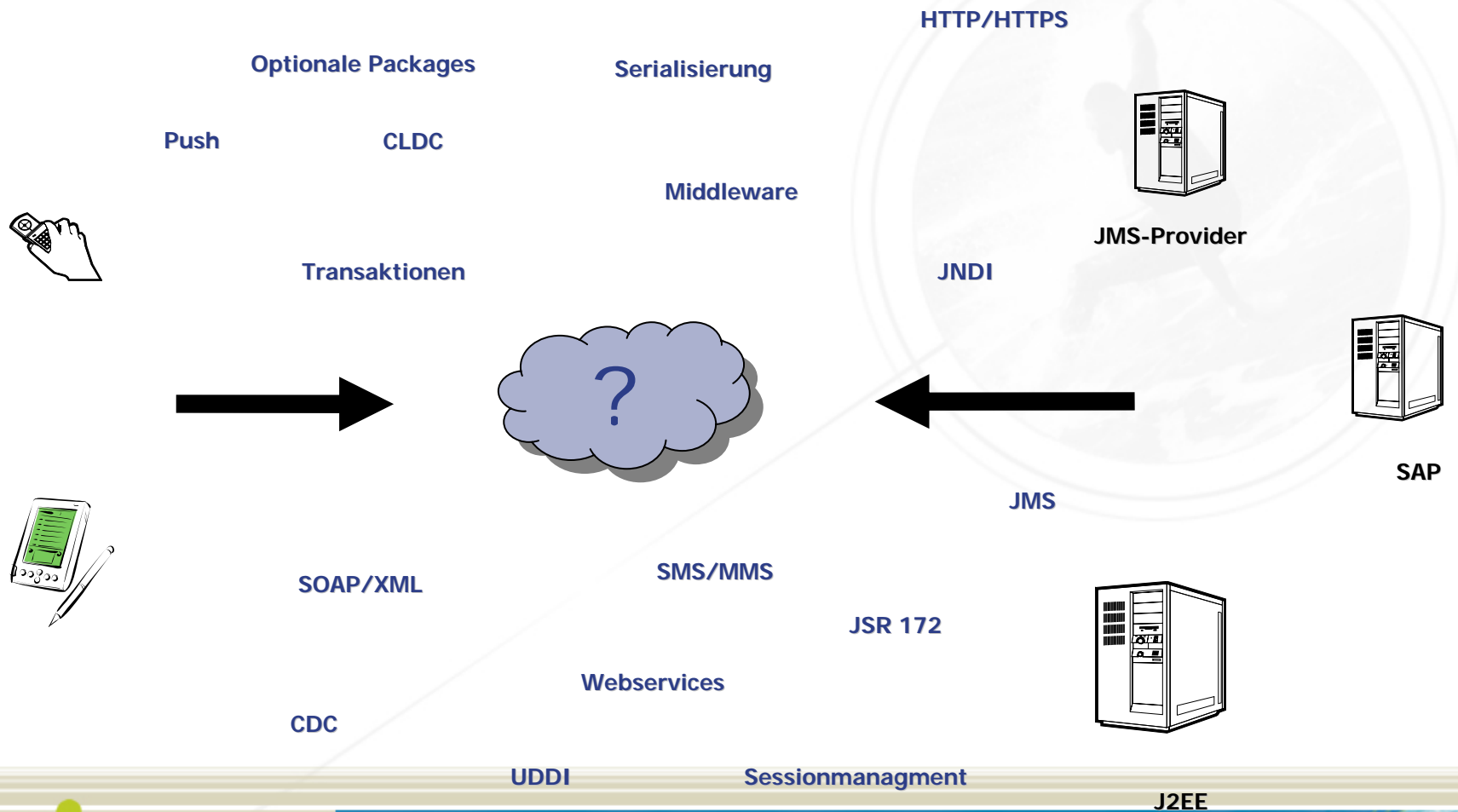
J2ME und Anbindung an Serversysteme



Wer ist der Speaker?

- Frank Schlinkheider
 - 34 Jahre, verheiratet
 - Geschäftsführer der ITSD Consulting GmbH
 - Seit 10 Jahren OO (Smalltalk -> Java -> ??)
 - Schwerpunkte: J2ME und Modellgetriebene SE
- ITSD Consulting GmbH
 - Consulting und Produktentwicklung
 - Objektorientierung, Java (J2ME, J2EE)
 - Banken, Versicherungen, Metallindustrie (ERA)
 - Sitz in OWL

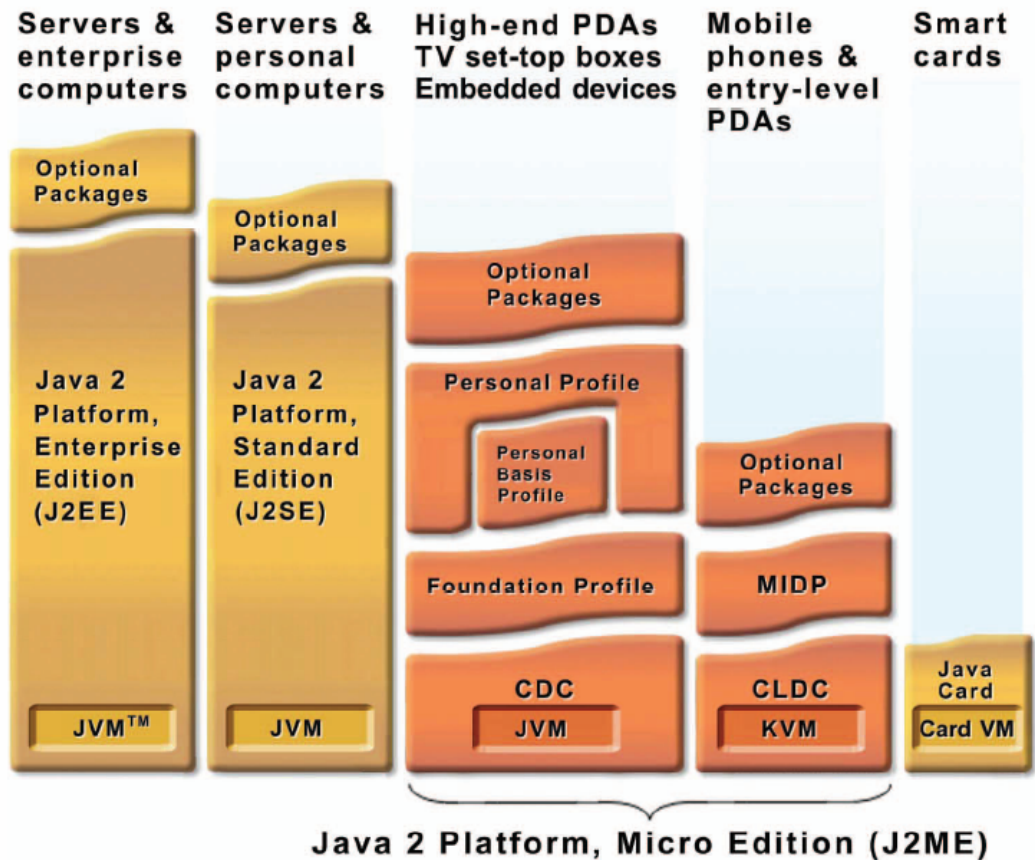
Worum geht es?



Agenda

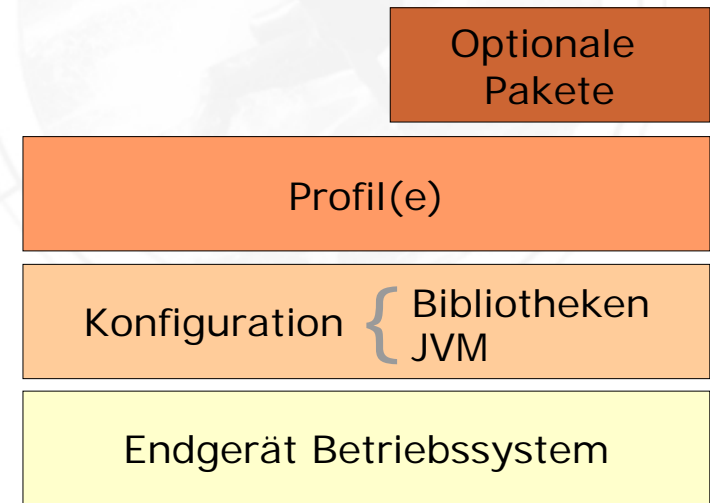
- Kurze Einführung in Java 2 Micro Edition
 - Besonderheiten
 - Probleme
- Anbindungsmöglichkeiten
 - SMS, PushRegistry
 - HTTP, kSOAP, J2ME Web Services (JSR 172)
 - Mobile Middleware
- Beispiele: PushSMS, kSOAP und jtom
- Zusammenfassung und Diskussion

Einführung - Die Java Familie

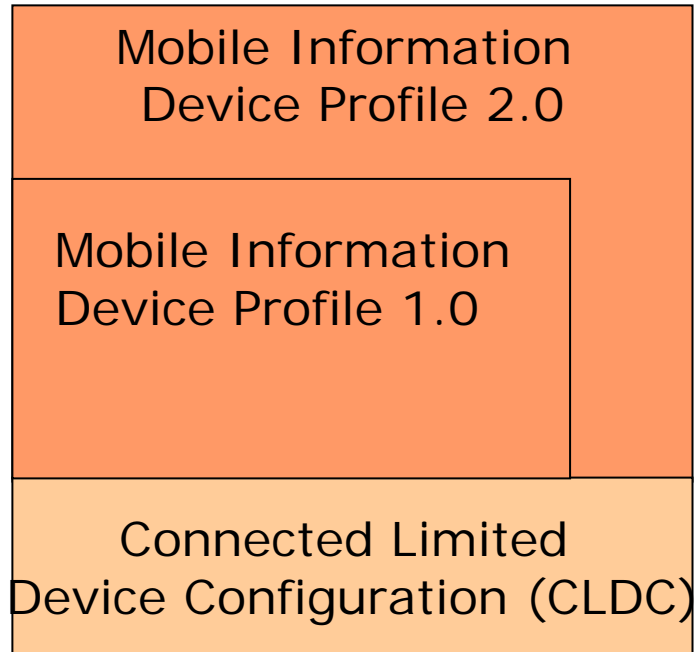


Einführung - Überblick

- mobile und kleinere Geräte
- *Konfiguration*
definieren der Funktionalität
- *Profil(e)*
zusätzliche Funktionalität
für diese Segmente
- *Optionale Pakete*
erweiternde Funktionalität
- In Summe => J2ME API

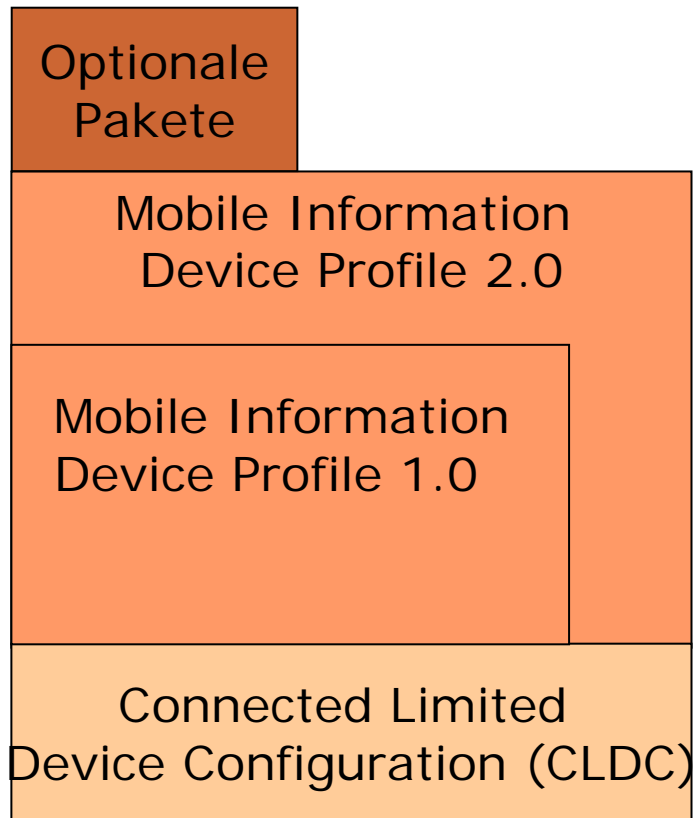


Einführung - CLDC/MIDP



- **MIDP 1.0**
 - HTTP
 - Record Management System
 - Benutzungsschnittstelle:
 - portable High Level API
 - direkte Low Level API
- **MIDP 2.0**
 - MIDP 1.0
 - HTTPS
 - Push Registry

Einführung - CLDC/MIDP



- **Optionale Pakete**
 - Bluetooth
 - Webservices
 - SMS (WMA)
 - MMAPI
 - Location API
 - etc.
- **JSR 185**
- **Schlanke VM (ca. 80 KByte)**

Einführung - CLDC/MIDP

JAD-Datei

- Textdatei
- Beschreibung der Anwendung
- Informationen für den Download
- Security
- PushRegistry
- Anwendungsattribute

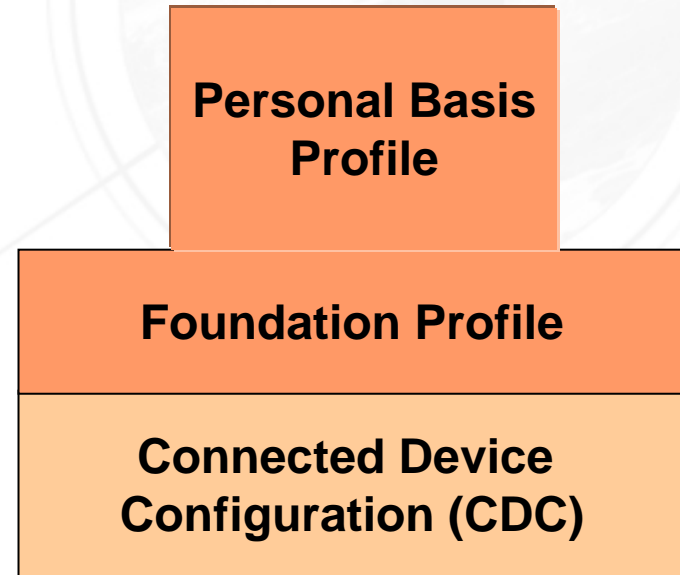
Einführung - CLDC/MIDP

JAD-Datei

- MIDlet-Name: Name der MIDlet-Suite
- MIDlet-Version Versionsnummer der MIDlet-Suite
- MIDlet-Vendor Hersteller
- MIDlet-Description Allgemeine Beschreibung
- MIDlet-Info-URL URL für weitere Informationen
- MIDlet-Data-Size Mindestspeicher für die Speicherung des MIDlets
- MIDlet-1 Name, Icon und Klasse des MIDlets
- MicroEdition-Profile Name und Version des Profile
- MicroEdition-Configuration Name und Version der Configuration
- MIDlet-JAR-URL URL zum Download
- MIDlet-Icon Pfadangabe für Icon
- MIDlet-JAR-Size Grösse der MIDlet-Suite JAR-Datei

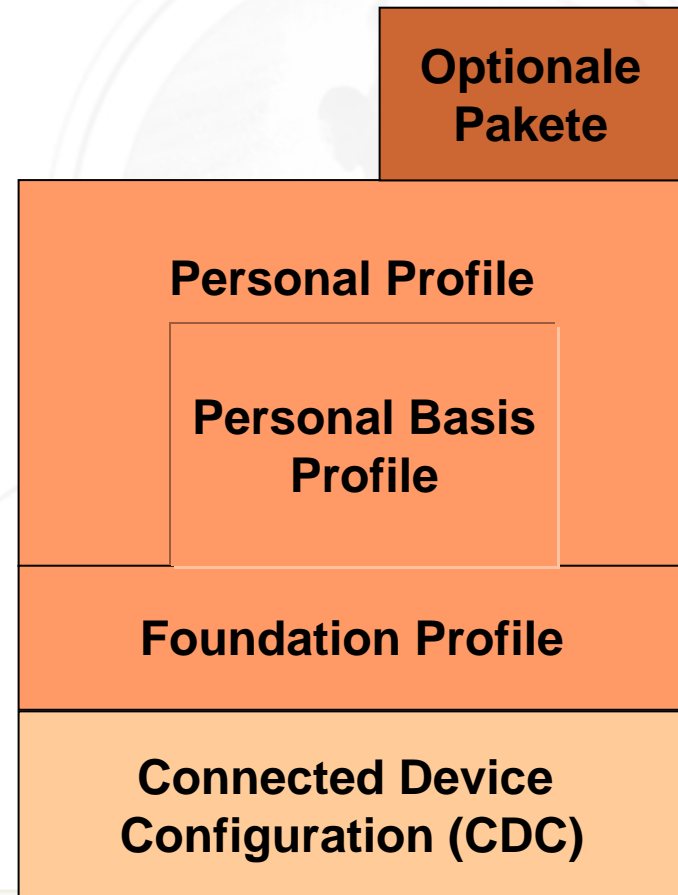
Einführung - CDC + Profile

- **Foundation Profile**
 - ohne grafische Schnittstelle
 - Optimierte Basisklassen
- **Personal Basis Profile**
 - ohne volle AWT Unterstützung
 - unterstützt xlet Programmiermodell



Einführung - CDC + Profile

- Personal Profile
 - AWT und
 - Applet Programmiermodell
- Optionale Pakete
 - J2ME RMI
 - JDBC



Einführung - Anbindungsprobleme

- Nur HTTP/HTTPS
- Keine einheitliche Zugriffsmöglichkeit
- Kein JNDI
- Kein RMI
- Keine asynchrone Kommunikation (JMS)
- Messaging nur über SMS/MMS

Einführung - Anbindungsprobleme

Zudem

- Keine standardisierte mobile Middleware
- Unzuverlässige Netzverbindungen

Lösung?

Agenda

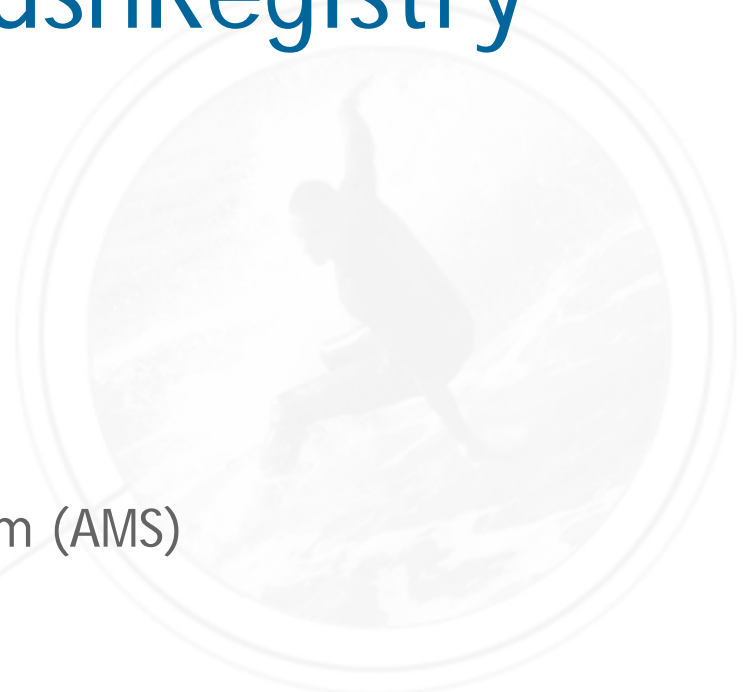
- Kurze Einführung in Java 2 Micro Edition
 - Besonderheiten
 - Probleme
- Anbindungsmöglichkeiten
 - SMS, PushRegistry
 - HTTP, kSOAP, J2ME Web Services (JSR 172)
 - Mobile Middleware
- Beispiele: PushSMS, kSOAP und jtom
- Zusammenfassung und Diskussion

Anbindungen - SMS/MMS

- Messaging
 - SMS: Text, 160 bis 1024 Zeichen
 - MMS: Bilder, Sound, Video, beliebige Länge
 - Keine Serialisierung
 - Keine Transaktionen
 - Kosten
 - Keine zeitlich garantierte Zustellung
 - Kein Empfang von „normalen“ Nachrichten
 - Keine „direkte“ Anbindung an Serversysteme
- > *für Anbindung geeignet, Versandkosten*

Anbindungen - PushRegistry

- MIDP 2.0
- Starten von Midlets
 - per Connection
 - per Timer
- Application Management System (AMS)
- Filter
- Connection
 - SMS
 - Datagram
 - Socket



Anbindungen - PushRegistry

- Statisch
 - Midlet-Attribut im Jad-File
 - Fester Port
 - Deinstallation entfernt Registrierung
- Dynamisch
 - J2ME-API
 - „dynamischer Port“
- AMS „cached“ die Daten

Anbindungen - PushRegistry

Statische Registrierung

MIDlet-Push-<n>: <ConnectionURL>, <MIDlet-ClassName>,
<AllowedSender>

MIDlet-Push-<n>	Name
<ConnectionURL>	URL für die eingehende Verbindung
<MIDletClassName>	vollständiger Klassenname
<Allowed-Sender>	Filter für Server

MIDlet-Push-1: socket://:5000,itsd.basicpush.PushMIDlet, *

Anbindungen - PushRegistry

Dynamische Registrierung

```
PushRegistry.registerConnection(<ConnectionURL>,  
                                < MIDletClassName>, <Allowed-Sender>);
```

<ConnectionURL>	URL für die eingehende Verbindung
<MIDletClassName>	vollständiger Klassenname
<Allowed-Sender>	Filter für Server

```
PushRegistry.registerConnection(socket://:5000,  
                                itsd.basicpush.PushMIDlet, *);
```

Agenda

- Kurze Einführung in Java 2 Micro Edition
 - Besonderheiten
 - Probleme
- Anbindungsmöglichkeiten
 - SMS, PushRegistry
 - HTTP, kSOAP, J2ME Web Services (JSR 172)
 - Mobile Middleware
- Beispiele: PushSMS, kSOAP und jtom
- Zusammenfassung und Diskussion

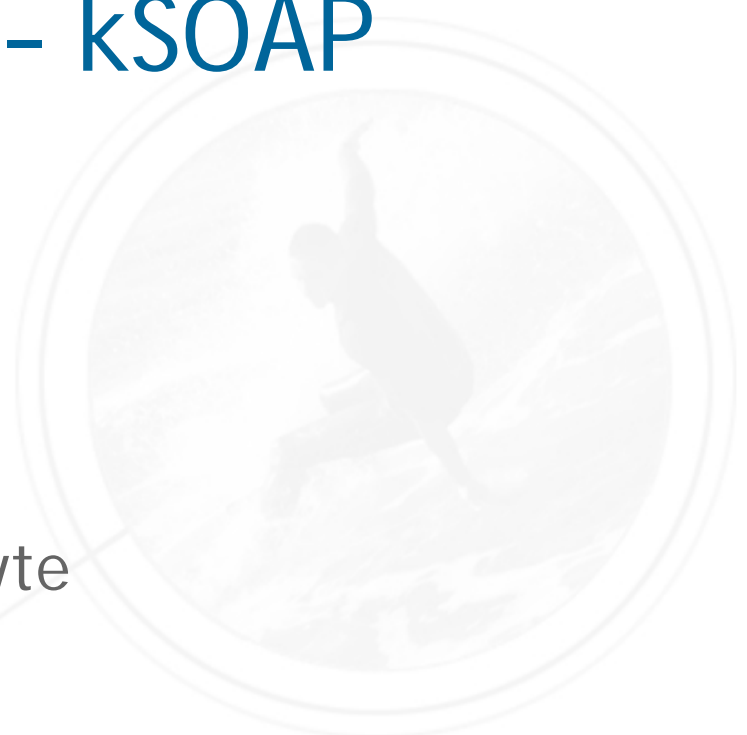
Anbindungen - HTTP/HTTPS

- Request/Response
- Keine Serialisierung
- Keine Transaktionen
- Session-Management
- Kein Messaging
- Keine „direkte“ Anbindung an Serversysteme

-> *für einfachste Anbindung geeignet.*

Anbindungen - kSOAP

- Webservice für J2ME
- Open Source
- CDC(J2SE)/CLDC
- Client-Bibliothek < 40 KByte
- Serialisierung

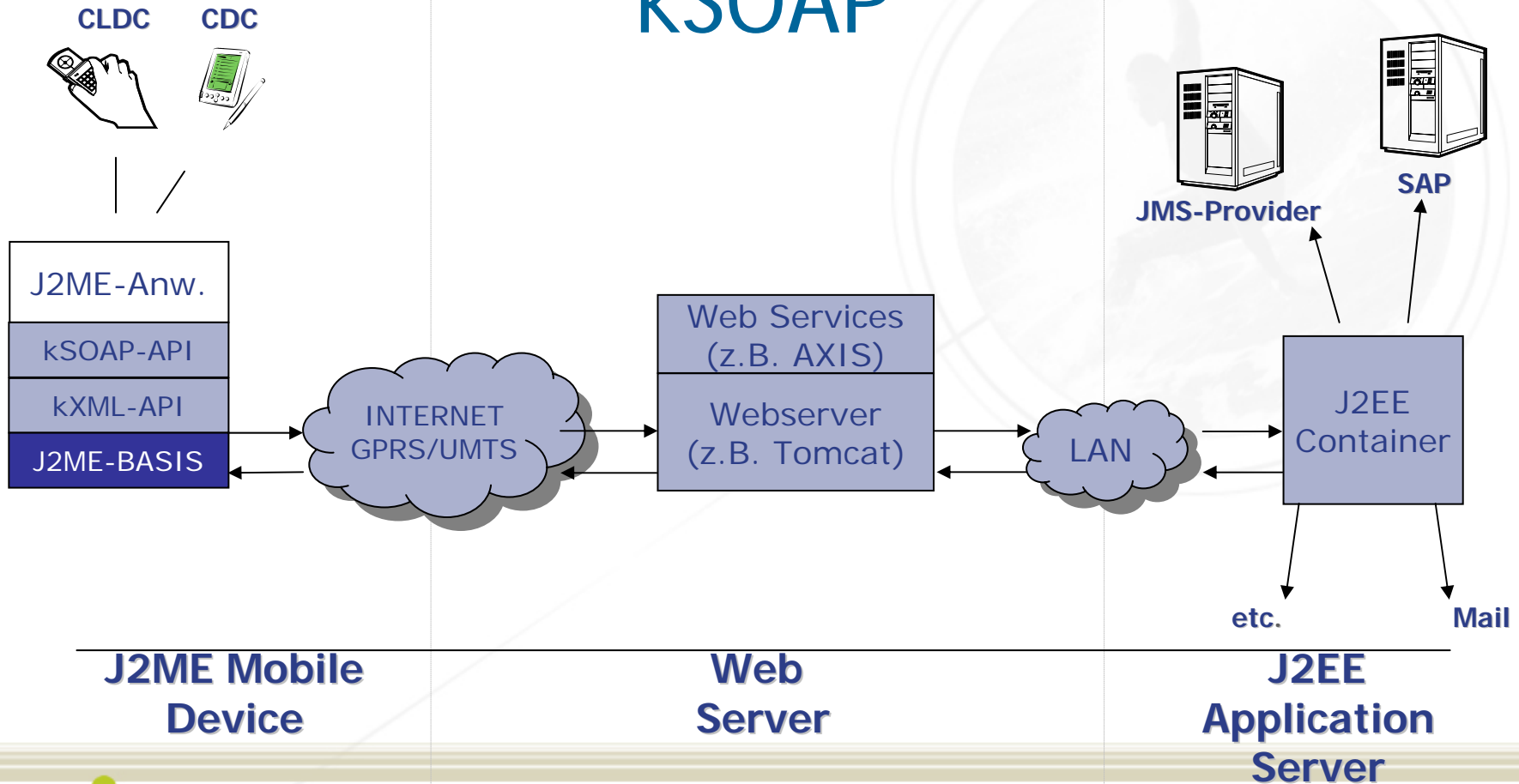


Anbindungen - kSOAP

- Kein UDDI
- Keine Transaktionen
- Kein Sessionmanagement
- Keine SOAP Messages nach SOAP 1.1 Encoding
- Keine „direkte“ Anbindung

-> *einfache kostengünstige Lösung*

Anbindungen - Architekturbeispiel kSOAP



Anbindungen - J2ME Webservice API (JSR172)

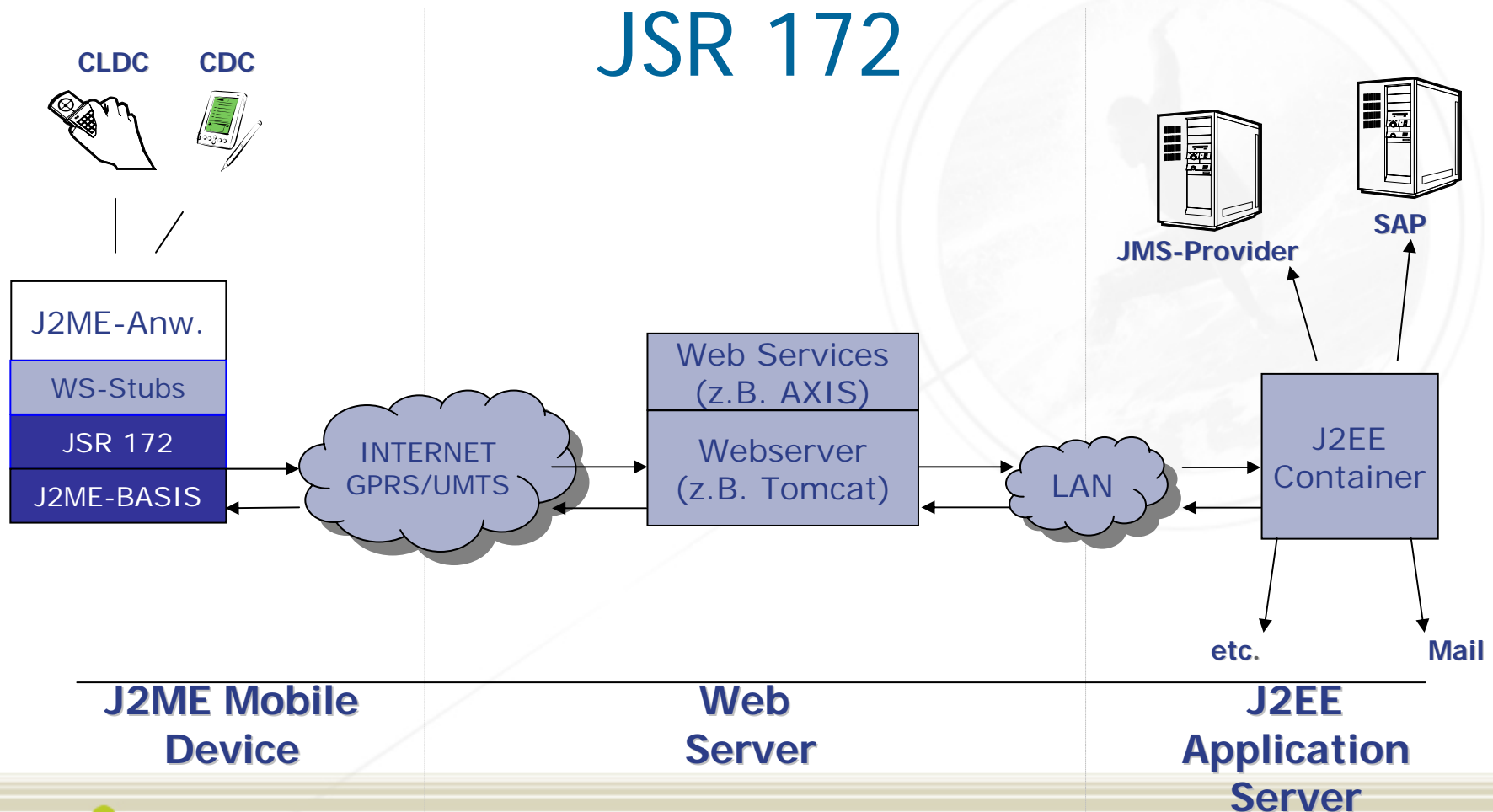
- Webservices für J2ME (JSR 172)
- Optionales Package!
- CDC/CLDC
- SOAP 1.1
- Stub-Generator
- Serialisierung

Anbindungen - J2ME Webservice API (JSR172)

- Keine Transaktionen
- Sessionmanagement
- Keine SOAP Messages mit SOAP 1.1 Encoding
- Kein UDDI
- Keine „direkte“ Anbindung

-> gute Lösung, aber nicht für alle Endgeräte vorhanden

Anbindungen - Architekturbeispiel JSR 172



Agenda

- Kurze Einführung in Java 2 Micro Edition
 - Besonderheiten
 - Probleme
- Anbindungsmöglichkeiten
 - SMS, PushRegistry
 - HTTP, kSOAP, J2ME Web Services (JSR 172)
 - Mobile Middleware
- Beispiele: PushSMS, kSOAP und jtom
- Zusammenfassung und Diskussion

Anbindungen - mobile Middleware jtom

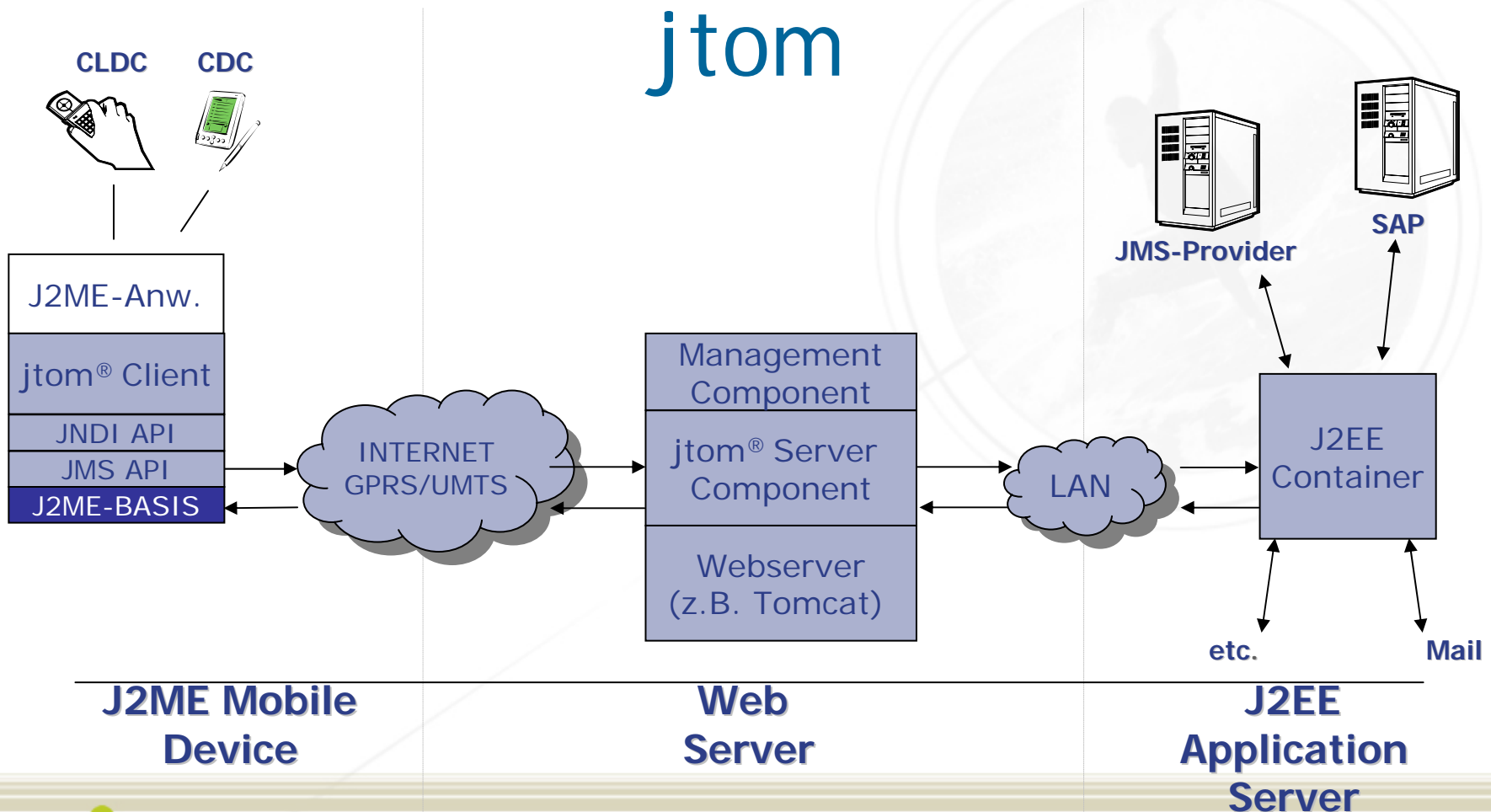
- JNDI und JMS
- Client-Package < 50 KByte (für JNDI < 40 KByte)
- Effiziente Serialisierung
- Transaktionsmanagement
- Sessionmanagement
- Messaging

Anbindungen - mobile Middleware jtom

- „direkte“ Anbindung an Serversysteme
- Unterstützung von CDC/CLDC
- Sockets/HTTP/HTTPS
- JBoss, IBM WebSphere, BEA WebLogic
- Unterstützung jeglicher VM

-> *gute Lösung, kommerzieller Einsatz: Lizenzkosten*

Anbindungen - mobile Middleware jtom



Agenda

- Kurze Einführung in Java 2 Micro Edition
 - Besonderheiten
 - Probleme
- Anbindungsmöglichkeiten
 - SMS, PushRegistry
 - HTTP, kSOAP, J2ME Web Services (JSR 172)
 - Mobile Middleware
- Beispiele: PushSMS, kSOAP und jtom
- Zusammenfassung und Diskussion

Anbindungen – SMS Example

- Es soll eine J2ME-SMS verschickt werden
- Registrierung für den SMS-Empfang
- Empfangen der SMS

Anbindungen - SMS senden

```
try {  
    String number = "+4917112345678";  
    String url = "sms://" + number + ":" + 4711;  
    MessageConnection messageCon = (MessageConnection)  
        Connector.open(url);  
    TextMessage msg = (TextMessage)  
        messageCon.newMessage(MessageConnection.TEXT_MESSAGE);  
    msg.setPayloadText("Hello World");  
    messageCon.send(msg);  
}  
catch (Exception e) {  
    ...  
}
```

Anbindungen - Registrierung SMS

```
try {  
    String number = "+4917112345678";  
    String url = "sms:///4711";  
  
    MessageConnection messageCon =(MessageConnection)  
        Connector.open(url);  
  
    // "this" implements MessageListener  
    messageCon.setMessageListener(this);  
  
}  
catch (Exception e) {  
    ...  
}
```

Anbindungen -SMS Empfangen

```
public void notifyIncomingMessage(MessageConnection conn) {  
  
    try {  
        TextMessage lastMsg = (TextMessage ) conn.receive();  
        String str =lastMsg.getPayloadText();  
        ...  
    }  
    catch (Exception e) {  
        ...  
    }  
}
```

Anbindungen - kSOAP Example

- Es soll eine einfache Preisinformation eingeholt werden.
- Serverklasse „sample.ksoap.PriceService“
- WSDD (Deployment für AXIS)
- Clientklasse „sample.ksoap.PriceJ2MEService“

Anbindungen - kSOAP Webservice

```
package sample.ksoap;

public class PriceService {

    public String getPrice(String number) {

        //Return the price
        if (number.equals("1"))
            return new String(1500.50f);
        else
            return new String(2000.00f);
    }
}
```

Anbindungen - kSOAP Deployment

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="PriceService" provider="java:RPC">
    <parameter value="sample.ksoap.PriceService" name="className"/>
    <parameter value="getPrice" name="allowedMethods"/>
  </service>
</deployment>
```

Anbindungen - kSOAP Client

```
try {
    String symbol = "1";
    SoapObject rpc = new SoapObject("urn:PriceService", "getPrice");
    rpc.addProperty ("price", symbol);

    SoapSerializationEnvelope envelope =
        new SoapSerializationEnvelope
    envelope.bodyOut = rpc;

    HttpTransport ht = new HttpTransport
        ("http://localhost:80/axis/services/PriceService");

    ht.call("urn:PriceService#getPrice", envelope);
    return envelope.getResult();
}
catch (Exception e) {
```

Anbindungen - jtom Example

- Client registriert sich für bestimmte JMS-Nachrichten
- Client verschickt JMS-Nachrichten
- Client empfängt bestimmte JMS-Nachricht

Anbindungen - jtom - Registrierung

```
/* register a MessageListener with a message provider */
public void registerReceiver(String url) {
    try {
        // open connection to JMS-server
        MessageConnection mConn = (MessageConnection)Connector.open(url);
        // "this" is registered as MessageListener for JMS-messages
        // When receiving a message the method
        // "this.notifyIncomingMessage()" is called
        mConn.setMessageListener(this, "userid = '4711'", true);

        mConn.setExceptionListener(this);
    } catch (java.io.IOException ioEx) {
        // handle Exceptions as required
    }
}
```

Anbindungen - jtom - URL

```
// req: jtom protocol identification
String url = "wma2jms://"

// req: connection url for service
+"http://localhost:8080/jtomServer/MIDPService;"

// req: key for accessing WMA-to-JMS mapping bean
+"jtom/Connection;"

// req: JMS connection factory +"factory=ConnectionFactory;"
// req: message destination name
+"name=testTopic;"

// req: used messaging model (topic|queue)
+"type=TOPIC;"

// opt: user and password
+"username=john;password=needle";
```

Anbindungen - jtom - JMS-Versand

```
/* Send JMS-messages. */
public void sendMessage(String message, String url) {
    try {
        // open connection to JMS-server
        MessageConnection mConn = (MessageConnection)Connector.open(url);

        // create new TextMessage
        TextMessage msg = (TextMessage)mConn.
            newMessage(MessageConnection.TEXT_MESSAGE);

        // set Property
        msg.setMessageProperty("userid", "4711");
        // set text to send
        msg.setPayloadText(message);
        // send message
        mConn.send(msg);

    } catch (java.io.IOException ioEx) {
        // handle Exceptions as required
    }
}
```

Anbindungen - jtom - JMS-Empfang

```
/* This method is called when a message arrives. */
public void notifyIncomingMessage(MessageConnection conn) {
    try {
        // fetches message from connection
        Message msg = (Message) conn.receive();

        // type based handling of messages
        if(msg instanceof BinaryMessage) {
            // BinaryMessage
            byte[] bytes = ((BinaryMessage )msg).getPayloadData();
        } else {
            // TextMessage
            String string = ((TextMessage )msg).getPayloadText();
            // do other stuff with the payload here
        }
    } catch (java.io.IOException ioEx) {
        // handle Exceptions as required
    }
}
```

Agenda

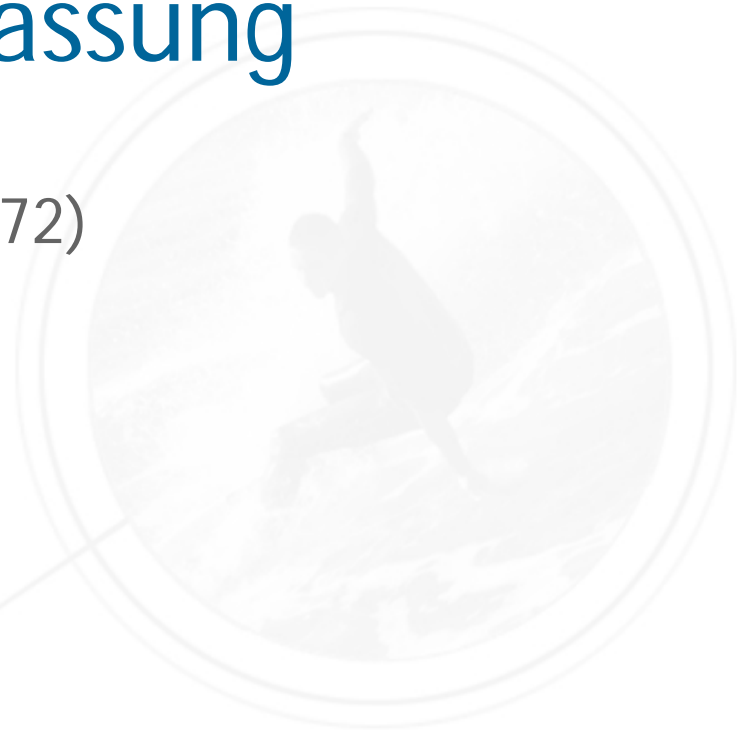
- Kurze Einführung in Java 2 Micro Edition
 - Besonderheiten
 - Probleme
- Anbindungsmöglichkeiten
 - SMS, PushRegistry
 - HTTP, kSOAP, J2ME Web Services (JSR 172)
 - Mobile Middleware
- Beispiele: PushSMS, kSOAP und jtom
- Zusammenfassung und Diskussion

Zusammenfassung

- SMS/MMS/PushRegistry
 - immer vorhanden
 - Kosten entstehen beim Versender
 - Aufwecken von Midlets
 - Kein Zugriff auf den „normalen“ SMS-Ordner
- HTTP/HTTPS
 - immer vorhanden
 - aufwendige Umsetzung
 - keine direkte Anbindung an das Backendsystem
- kSOAP
 - klein und flexibel
 - Open Source
 - erhöht die Client-JAR-Größe
 - CDC und CLDC

Zusammenfassung

- J2ME Web Services (JSR 172)
 - optionales Package
 - CDC und CLDC
 - Stub-Generator
- jtom
 - „direkte“ Verbindung
 - JNDI/JMS
 - Messaging zum J2ME-Client
 - Sessionmanagement
 - effiziente Serialisierung



Zusammenfassung

- Muss das Midlet „aufgeweckt“ werden
- Verwenden Sie bereits Webservices im Unternehmen?
- Benötigen Sie ein Sessionmanagement für die Anbindung Ihrer mobilen Endgeräte?
- Können Sie die zu benutzenden Endgeräte vorgeben?

Zusammenfassung

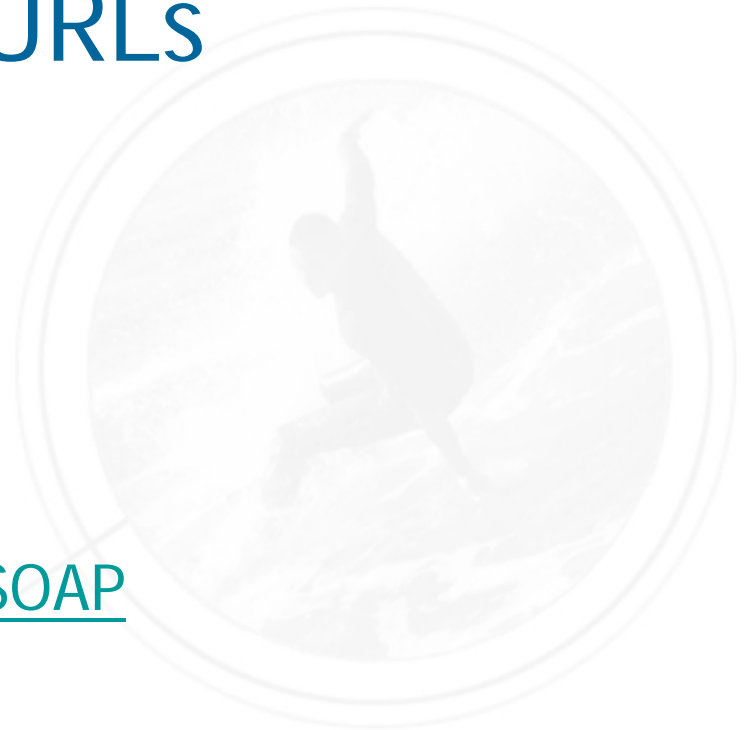
- Kennen Sie alle zu benutzenden Endgeräte?
- Benötigen Sie eine JMS-Funktionalität für Ihre Applikation?
- Ist die Netzbandbreite gering?
(GPRS oder GSM)

Wichtige URLs

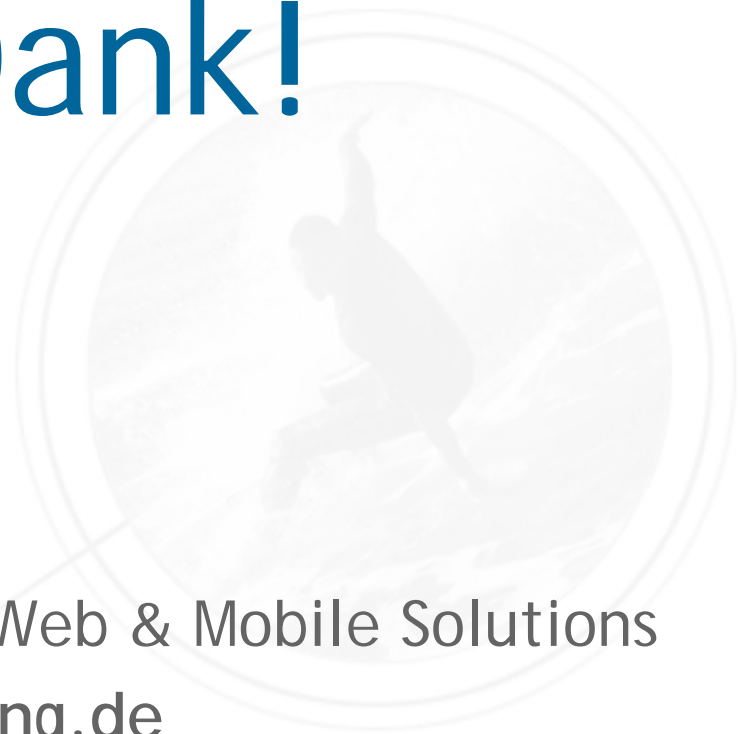
- J2ME Web Services API (JSR 172)
<http://www.jcp.org/en/jsr/detail?id=172>
- kSOAP
<http://www.ksoap.org>
- J2ME
<http://developers.sun.com/techttopics/mobility>
- Java Web Services
<http://java.sun.com/webservices>

Wichtige URLs

- jtom
<http://www.jtom.de>
- SOAP v1.1
<http://www.w3.org/TR/SOAP>
- WebServices AXIS
<http://ws.apache.org/axis>



Vielen Dank!



Frank Schlinkheider • Enterprise Web & Mobile Solutions
fs@itsd-consulting.de
www.itsd-consulting.de und www.jtom.de

