Kapitel 3

WAP-Anwendungen unter WML schreiben

Es gibt keinen besseren Weg, eine Sprache zu lernen, als damit abzufangen, sie zu sprechen. In diesem Kapitel werden wir genau dies tun. Nachdem wir das Konzept der Cards und die Sprache WML vorgestellt haben, werden wir jetzt mit der Gestaltung solcher Cards herumexperimentieren. Dieses Kapitel wird Ihnen nichts beibringen, was Sie über WML wissen müssten, aber es wird Sie in kürzester Zeit auf den Weg bringen.

3.1 Gestalten Sie Ihre erste Card

Wie Sie im vorangegangenen Kapitel gelernt haben, werden WAP-Seiten *Cards* genannt, und diese Cards werden mit WML erstellt. WML ist eine einfache Sprache, sogar einfacher als HTML, aber anders als HTML hat es sehr strenge Regeln.

Um zu beginnen, betrachten wir ein vollständiges WML-Beispiel – eine grundlegende Card mit einfachem Text. Diese Karte wird die Ausgabe erzeugen, die Sie in Abbildung 3.1 sehen.

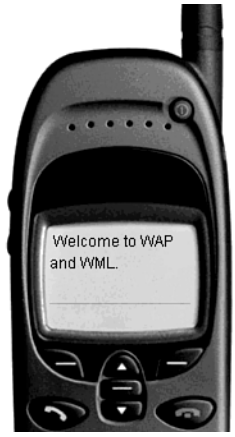

Bild 3.1: Geräteemulatoren sind ideal zum Testen von WML-Programmen.

Hinweis

Ich empfehle Ihnen sehr, diese Beispiele selbst auszuprobieren, während Sie sie durcharbeiten. Der einfachste Weg dazu besteht darin, einen der vielen Geräteemulatoren oder eine Entwicklungsumgebung zu verwenden. Anhang C gibt Ihnen mehr Informationen darüber.

Wenn Sie keinen Emulator besitzen, aber ein WAP-Gerät (ein Handy), können Sie diese Seiten an einen WAP- oder Web-Server übertragen. (Letzterer erfordert es, dass Sie die entsprechenden MIME-Typen angeben wie in Kapitel 2 beschrieben.)

Unter HTML können Sie diese Seite einfach erstellen, indem Sie nichts anderes als den Ausgabetext eingeben. Unter WML können Sie das nicht tun. Die Mindestanforderung an Code, die WML für die Erzeugung dieser Ausgabe benötigt, sehen Sie in Listing 3.1

Listing 3.1: Ihre erste WML-Seite

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
➥"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

   <card>
      <p>
         Welcome to WAP and WML.
      </p>
   </card>

</wml>
```

Wir werden den Code jetzt kurz durchgehen. Die ersten beiden Zeilen enthalten die XML-Deklarationen. Sie legen Typ und Version des Dokuments fest und sollten in allen WML-Dateien enthalten seien. (Ich habe WML Version 1.2 benutzt, und wenn Sie in 1.0 oder 1.1 schreiben, wird `<!DOCTYPE>` ein wenig anders aussehen.

Das gesamte Dokument wird von den Tags `<wml>` und `</wml>` eingeschlossen. Dies ist notwendig, und jede WML-Seite muss einen Satz zusammengehöriger `<wml>`-Tags aufweisen. Cards (jawohl, *Cards* im Plural – in einer einzigen Datei sind mehrere Cards erlaubt) erscheinen zwischen diesen Tags.

Die Card selbst wird durch das Tag `<card>` definiert und endet mit einem Tag `</card>`. Alle Card-Inhalte müssen zwischen diesen beiden Tags stehen. In diesem Beispiel habe ich eine einzige Textzeile eingegeben, die jedoch durch die Tags `<p>` und `</p>` umfasst wird. `<p>` ist ebenso ein Absatz-Kennzeichner wie sein Gegenstück in HTML. Aber anders als HTML verlangt WML dieses Tag. Text kann nicht unmittelbar in eine Card eingegeben werden; es müssen Absätze in ihr platziert und der Text innerhalb dieser Absätze geschrieben werden. Der folgende Ausschnitt für den Code einer Card würde daher einen Fehler hervorrufen:

```
<card>
   Welcome to WAP and WML.
</card>
```

Warnung

Obwohl wir es bereits erklärt haben, ist es doch wert, erneut darauf hinzuweisen, dass WML Groß- und Kleinschreibung beachtet (hätten wir anstelle von `<p>` *das Tag* `<P>` *verwendet, hätte dies einen Fehler erzeugt) und dass alle Tags passende End-Tags benötigen (das Weglassen von* `</p>` *hätte ebenfalls einen Fehler hervorgerufen).*

Viele WML-Tags können zusätzliche Attribute aufnehmen, und viele dieser Attribute führen je nach dem verwendeten Gerät zu unterschiedlichen Ergebnissen. Ein Beispiel dafür ist das Attribut `title` des Tags `<card>`, das dazu verwendet wird, einen Titel für die Card bereitzustellen, den das Gerät anzeigen (oder ignorieren) kann. Listing 3.2 zeigt eine überarbeitete Version der Welcome-Seite, die wir gerade gestaltet haben. Der einzige Unterschied besteht darin, dass wir das Attribut `title` hinzugefügt haben.

Listing 3.2: Einzeigen eines Titels für die Card

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
➥"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

   <card title="Welcome!">
      <p>
         Welcome to WAP and WML.
      </p>
   </card>

</wml>
```

Um sich vorzustellen, was ich damit meine, dass verschiedene Geräte das Attribut `title` unterschiedlich behandeln, betrachten Sie bitte Abbildung 3.2 bis Abbildung 3.4. Wie Sie sehen, zeigt der Nokia-Browser den `title`-Text zentriert und als Seitentitel an, der Ericsson-Browser auf Ähnliche Weise, jedoch zur Verstärkung noch umrandet, während der Browser von Phone.com überhaupt keinen Text wiedergibt.

Bild 3.2: Der WAP-Browser von Nokia wird nur von Nokia-Geräten verwendet.

Bild 3.3: Der WAP-Browser von Ericsson wird nur von Ericsson-Geräten verwendet.

Bild 3.4: Der WAP-Browser von Phone.com wird von vielen Geräten vieler Hersteller verwendet und ist der derzeit am weitesten verbreitete Browser.

Hinweis

Anders als HTML verlangt WML, dass alle Tag-Attributwerte in Anführungszeichen gesetzt werden. Im vorigen Beispiel hätte die Angabe `title=Welcome!` *(ohne Anführungszeichen) einen Fehler hervorgerufen.*

Außerdem (und wiederum anders als HTML) erlaubt WML keine Leerzeichen vor oder hinter den Gleichheitszeichen zwischen einem Attribut und seinem Wert.

Tipp

Überprüfen Sie Ihren Code auf so vielen Browsern und Geräten wie möglich. Wie Sie sehen, erzeugt sogar der einfachste Code auf verschiedenen Geräten eine unterschiedliche Ausgabe.

3.1.1 Grundlegende Textformatierung

WAP-Browser unterstützen nur die grundlegendsten Textformatierungen. Es gibt keine verschiedenen Schriften und nur wenige Möglichkeiten zur Veränderung von Schriftgrößen und Hervorhebungen. Unglücklicherweise neigen sogar diese elementarsten Funktionen dazu, von den verschiedenen Geräten nicht einheitlich implementiert zu werden.

Die Textformatierung wird in einem späteren Kapitel genauer beschrieben, aber lassen Sie uns bereits hier ein einfaches Beispiel betrachten. Listing 3.3 enthält wiederum den bekannten Welcome-Bildschirm, aber dieses Mal werden die Wörter WAP und WML durch Fettstellung hervorgehoben (wie in Abbildung 3.5 gezeigt).

Listing 3.3: Textformatierung

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
➥"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

   <card title="Welcome!">
      <p>
         Welcome to <b>WAP</b> and <b>WML</b>.
      </p>
   </card>

</wml>
```

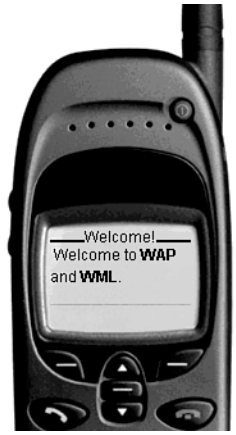

Bild 3.5: Textformatierung – z. B. eine Anzeige in Fettschrift – ist nicht auf allen Geräten möglich.

Wie Sie sehen, markiert das Tag `<b>` den Text für die Anzeige in Fettschrift (genau wie das HTML-Tag `<b>`).

3.1.2 Grundlegende Absatzformatierung

WAP-Browser unterstützen alle grundlegenden Arten der Absatzformatierung. Sie können Text ausrichten, Zeilenumbrüche einfügen und sogar Tabellen verwenden. Aber wiederum neigen einige der ausgefeilteren Funktionen (wie die Ausrichtung und die Benutzung von Tabellen) leider dazu, nur sporadisch implementiert zu sein. Zum Glück scheinen fast alle Browser immerhin grundlegende Absatz- und Zeilenumbrüche zu unterstützen.

Die Absatzformatierung wird ebenfalls in einem späteren Kapitel im Einzelnen abgehandelt, aber lassen Sie uns ein einfaches Beispiel betrachten. Listing 3.4 enthält einen leicht veränderten Welcome-Bildschirm (wie in Abbildung 3.6 gezeigt).

Listing 3.4: Formatieren eines Absatzes

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
➥"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

   <card title="Welcome!">
      <p>
         Welcome to WAP and WML.
         <br />
```

```
            This is easy, there is nothing to it.
        </p>
    </card>

</wml>
```

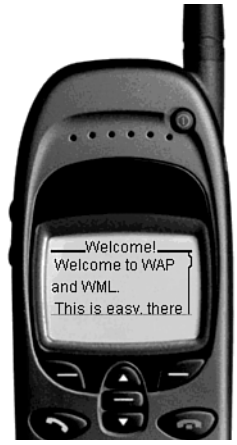

Bild 3.6: WAP-Browser ermöglichen automatisch das Rollen, wenn der Text nicht in die Anzeige passt.

Das Tag `<br>` fügt einen Zeilenumbruch ein. Anders als das HTML-Tag `<br>` erfordert dieses ein zugehöriges End-Tag (wie alle WML-Tags). Aus diesem Grund wird `<br />` benutzt. Dies ist eine Abkürzung für `<br></br>`, und diese Syntax wird bei allen Tags benötigt, die über kein passendes End-Tag verfügen.

3.2 Der Aufbau von Decks aus mehreren Cards

Soweit bestand jede WML-Datei, die wir erzeugt haben, aus einer einzigen Card. Wie ich zuvor erwähnt habe (und wie in Kapitel 2 erklärt wurde), können WML-Dateien mehrere Cards enthalten – diese Dateien werden als Decks bezeichnet.

Jede Card in einem Deck muss einen eindeutigen Namen tragen (eindeutig innerhalb des Decks). Cards werden mithilfe des Attributs `id` des Tags `<card>` benannt, und die `id`-Werte können Text und Zahlen enthalten (aber keine Sonderzeichen).

Wenn ein Deck geladen wird, verarbeitet und speichert der Browser alle enthaltenen Card und bestimmt dann, welche er anzeigen soll:

- Wenn eine bestimmte Card-`id` verlangt wurde und auch vorhanden ist, wird die zugehörige Card angezeigt.

- Wenn keine Card-`id` angegeben wurde oder die verlangte `id` nicht existiert, wird die erste Card des Decks angezeigt.

Listing 3.5 enthält den Code für ein einfaches Deck aus drei Cards. Ein einziges Paar von `<wml>`-Tags ist vorhanden, und alle drei `<card>`-Tags sind darin eingeschlossen.

Listing 3.5: Anzeigen eines einfachen Decks

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
➥"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

   <card id="card1">
      <p>
      This is card 1.
      </p>
   </card>

   <card id="card2">
      <p>
      This is card 2.
      </p>
   </card>

   <card id="card3">
      <p>
      This is card 3.
      </p>
   </card>

</wml>
```

Wie Sie in Listing 3.5 sehen können, wird jede Card durch das Attribut `id` eindeutig bezeichnet. Aber wie geben Sie nun die `id` der Card an, die angezeigt werden soll? Hier borgt sich WML abermals die Syntax von HTML und verwendet das Bookmark-Zeichen (#). In dem folgenden URL der WML-Datei:

```
localhost/wap/deck1.wml
```

zeigt der URL auf die Card mit der id card2 innerhalb dieses Decks:

```
localhost/wap/deck1.wml#card2
```

In Abbildung 3.7 sehen Sie die Card, die angezeigt würde, wenn keine Card-`id` angegeben wäre, und in Abbildung 3.8 die Anzeige bei der ausdrücklichen Angabe von `card2`.

Hinweis

Innerhalb eines Decks kann man einfach durch `#id` *auf eine Card verweisen. Von* `card1` *oder* `card2` *aus beziehen Sie sich in unserem Beispiel durch* `#card2` *auf* `card2`*.*

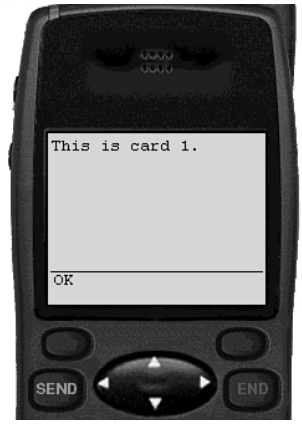

Bild 3.7: Standardmäßig wird die erste Card des Decks angezeigt.

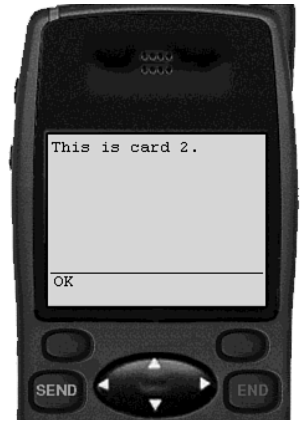

Bild 3.8: Bestimmte Cards des Decks können angezeigt werden, indem man die `id` dieser Card angibt.

Viele Geräteemulatoren speichern Anfragen zwischen, was dazu führen kann, dass ältere Versionen von Seiten angezeigt werden. Wenn Sie dieses Verhalten beobachten, müssen Sie den Zwischenspeicher nach den Angaben des Herstellers löschen.

3.2.1 Wozu Decks?

Decks aus mehreren Cards sind beim Entwerfen von WAP-Anwendungen äußerst wichtig. Bedenken Sie das folgende Beispiel:

- Sie stellen einen Suchbildschirm zur Verfügung, der den Benutzern erlaubt, nach Angestellten zu suchen.
- Wenn die Suche abgeschlossen ist, werden die Ergebnisse mit nur den notwendigsten Informationen dargestellt (vielleicht nur mit dem Namen).
- Jedes Ergebnis kann ausgewählt werden, um genauere Informationen über den Angestellten anzuzeigen (einschließlich der Telefonnummer und E-Mail-Adresse).

Dies ist ein Beispiel für eine klassische Drill-down-Benutzerschnittstelle. Drei Arten von Cards werden wir verwenden: die anfängliche Such-Card und zwei Typen von Ergebnis-Cards, eine für die Liste und eine für Einzelheiten.

Offensichtlich muss die Such-Card an das Gerät gesendet werden, bevor die Suche durchgeführt werden kann. Aber nachdem sie Ergebnisse gezeitigt hat, wird der Benutzer wahrscheinlich verschiedene Ergebnisse genauer betrachten, um die gewünschte Information zu erhalten.

Eine einfache Implementierung (eine, die wie eine HTML-Implementierung funktioniert) könnte wie folgt ablaufen:

1. Eine Ergebnis-Card mit der Liste der Treffer wird erzeugt.
2. Der Benutzer wählt einen Eintrag dieser Liste, woraufhin eine Anfrage an den Server zurückgesendet und die gewünschte Information angezeigt wird.
3. Um einen anderen Eintrag zu betrachten, muss der Benutzer zu der Liste zurückgehen (die auf dem Gerät zwischengespeichert wird) und eine weitere Auswahl treffen, woraufhin wiederum eine Anfrage an den Server zurückgesendet und die gewünschte Information angezeigt wird
4. Schritt 3 wird so oft wiederholt wie nötig, und jedes Mal wird eine neue Anfrage an den Server gesendet.

Obwohl dies eine funktionierende Lösung sein mag, ist es keinesfalls eine wirtschaftliche. Daten werden in Paketen an die Geräte gesendet, und das Senden von teilweise gefüllten Paketen dauert ebenso lange wie das von vollständig gefüllten. Mit anderen Worten, es wirkt sich nicht auf die Leistungsfähigkeit aus, zu viele Informationen zu versenden (oder Information, die noch nicht benötigt wird, aber voraussichtlich benötigt werden könnte).

Eine bessere Möglichkeit, die Drill-down-Schnittstelle zu handhaben, wäre die folgende:

1. Ein Ergebnis-Deck wird erzeugt, dessen erste Card eine Liste der Treffer enthält. Weitere Cards, die jeweils die Einzelheiten eines Eintrags enthalten, werden aufgebaut.
2. Der Benutzer wählt einen Eintrag aus der Liste, und die zugehörige Card wird angezeigt (ohne dass eine Anfrage an den Server benötigt wird, da sich die Card schon in dem Gerät befindet).
3. Um einen anderen Eintrag zu betrachten, muss der Benutzer zu der Liste zurückgehen (die auf dem Gerät zwischengespeichert wird) und eine weitere Auswahl treffen, woraufhin eine andere Card des Decks angezeigt wird (abermals ohne die Notwendigkeit, eine Anfrage an den Server zu richten).
4. Schritt 3 wird so oft wiederholt wie nötig, ohne dass jemals eine weitere Anfrage an den Server erforderlich würde.

Der Vorteil dieser Schnittstelle liegt darin, dass sie Ihnen erlaubt, die nächsten Anforderungen des Benutzers vorauszusehen und darauf zu antworten. Dies führt zu einer weit schnelleren und rückmeldungsfreudigeren Anwendung und verhindert die unnötige Verschwendung von Brandbreite.

Tipp

Wie später in diesem Buch erklärt wird, gibt es für die Größe von Decks eine Obergrenze. Aber so lange die Daten innerhalb dieser Größenbeschränkung passen, erfordert es ebenso viel Zeit, ein Deck von Cards zu senden wie eine einzelne Card. Daher gibt es keine Nachteile dadurch, zusätzliche Cards »für alle Fälle« zu senden.

3.2.2 Digests

Der UP.Link Server und -Browser (entworfen von Phone.com) unterstützt einen Mechanismus zur Gruppierung von mehreren Decks (oder eines einzelnen Decks und anderer Entitäten) in eine einzige Einheit, die *Digest* genannt wird. Digests wurden entworfen, um Ihnen zu erlauben, die Bandbreite und Paketgröße des mobilen Netzwerks vollständig auszunutzen und Ihre Anwendung kommunikativer zu gestalten. Der Grund, aus dem dies funktioniert, liegt darin, dass jedes Datenpaket, das zum Client gesendet wird, unabhängig vom Umfang der gesendeten Daten einen Overhead aufweist. Da Digests die Übertragung mehrerer Entitäten auf einmal erlauben, wird ein Gutteil des Overheads entfernt, der bei jeder einzelnen Einheit aufgetreten wäre.

So nützlich Digests auch sind, sie werden derzeit von keiner Plattform außer der von Phone.com unterstützt. Für mehr Informationen über Digests schlagen Sie in der Dokumentation nach, die von Phone.com zur Verfügung gestellt wird.

3.3 Grundlegende Navigationsfunktionen

Ebenso wie die Seiten einer Web-Site erfordern die Cards einer WAP-Anwendung die Möglichkeit einer Navigation, um von einer Card zur anderen zu gelangen. Da WAP-Geräte derzeit über keine Zeigegeräte verfügen (wie eine Maus oder einen Touchscreen), wird die gesamte Navigation über die Tastatur durchgeführt.

Zwei Arten der Navigation werden unterstützt:

- Anker können um Text oder Bilder herum platziert werden, sodass sie sich auswählen lassen (ähnlich wie mit dem HTML-Tag `<a>`).
- Alle WAP-Geräte verfügen über Aktionstasten (zumindest über zwei davon), die wie erforderlich programmiert werden können.

Wir werden uns diese beiden Möglichkeiten nun kurz anschauen.

→ Die Navigation wird ausführlich in Kapitel 4 dargestellt.

3.3.1 Erzeugen von Links

WML-Links sind ihren HTML-Entsprechungen sehr ähnlich. Der Text (oder das Bild), der (das) verlinkt werden soll, wird in die Tags `<a>` und `</a>` eingeschlossen und der Browser verwendet eine besondere Darstellungsform (gewöhnlich eine Unterstreichung), um anzuzeigen, dass er angewählt werden kann. Der tatsächliche Auswahlmechanismus ist von Gerät zu Gerät unterschiedlich, obwohl meistens die Accept-Taste für die Bestätigung verwendet wird.

Um die Benutzung von Links zu veranschaulichen, enthält Listing 3.6 eine erweiterte Version unseres Decks.

Listing 3.6: Verwenden von Links in einem einfachen Deck

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
➥"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

   <card id="index">
      <p>
      <a href="#card1">Card 1</a>
      <br />
      <a href="#card2">Card 2</a>
      <br />
      <a href="#card3">Card 3</a>
      <br />
      </p>
```

```
   </card>

   <card id="card1">
      <p>
      This is card 1.
      </p>
   </card>

   <card id="card2">
      <p>
      This is card 2.
      </p>
   </card>

   <card id="card3">
      <p>
      This is card 3.
      </p>
   </card>

</wml>
```

Die ursprünglichen Cards in diesem Deck haben sich nicht im geringsten geändert. Die einzige Veränderung besteht in der Hinzufügung der folgenden neuen Card (die jetzt die erste Card des Decks ist):

```
<card id="index">
      <p>
      <a href="#card1">Card 1</a>
      <br />
      <a href="#card2">Card 2</a>
      <br />
      <a href="#card3">Card 3</a>
      <br />
      </p>
   </card>
```

Diese führt die Namen der drei Cards auf, einen pro Zeile (durch einen Umbruch getrennt). Jeder Name ist ein durch die Tags `<a>` und `</a>` erzeugter Link. Der URL, zu dem dieser Link führen soll, wird durch das Attribut `href` angegeben, und in diesem Beispiel erfolgt die Verknüpfung zu Cards innerhalb desselben Decks (wobei die Syntax verwendet wird, die weiter oben in diesem Kapitel erklärt wurde).

Der Code aus Listing 3.6 erzeugt eine Anzeige wie die in Abbildung 3.9.

Nun kann jede Card angezeigt werden, indem man sie in der Liste auswählt und die Accept-Taste des Gerätes drückt.

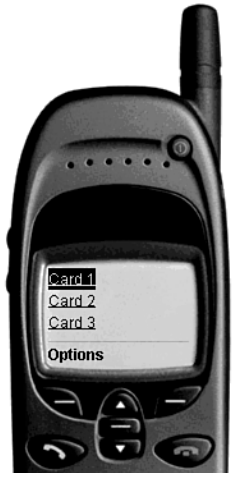

Bild 3.9: Links können verwendet werden, um Blöcke von Texten oder Bildern auswählen zu können.

3.3.2 Verwenden von Aktionen

Aktionen gibt es nur bei WML; sie haben kein Gegenstück in HTML. Jedes WAP-Gerät hat Tasten für besondere Aufgaben, gewöhnlich mindestens zwei und gewöhnlich unmittelbar unterhalb des Bildschirms. Meistens ist die linke die Accept- und die rechte die Back-Taste, aber nicht in jedem Fall. Unabhängig von der Anzahl der Tasten und ihrer Position ist eines sicher: Diese Tasten existieren und Sie können festlegen, was sie bewirken.

Eine Aktion zu programmieren, erfordert es, einen URL mit einem Tastentyp zu verbinden, und dies wird mithilfe des Tags `<do>` durchgeführt. Lassen Sie uns zur Veranschaulichung ein einfaches Beispiel betrachten:

```
<do type="accept">
 <go href="/index.wml" />
</do>
```

`<do>` erfordert es, dass ein Tastentyp angegeben wird; in diesem Beispiel haben wir `accept` benutzt. Zwischen den Tags `<do>` und `</do>` steht die Aktion, die ausgeführt werden soll, und hier haben wir das Tag `<go>` verwendet, um den URL festzulegen, zu dem übergegangen werden soll. (Es gibt weitere wichtige Einzelheiten bei der Verwendung von Aktionen, die in Kapitel 4 erklärt werden.)

Jetzt werden wir unser Beispiel-Deck erweitern, um eine Back-Taste auf jeder einzelnen Card-Seite zur Verfügung zu stellen, sodass Sie mit Leichtigkeit zur ersten Card zurückgehen können (wie in Abbildung 3.11 gezeigt). Listing 3.7 enthält den erweiterten Code.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
➥"http://www.wapforum.org/DTD/wml12.dtd">
```

Listing 3.7: Programmieren von Aktionstasten

```
<wml>

   <card id="index">
      <p>
      <a href="#card1">Card 1</a>
      <br />
      <a href="#card2">Card 2</a>
      <br />
      <a href="#card3">Card 3</a>
      <br />
      </p>
   </card>

   <card id="card1">
      <do type="options" label="Back">
       <prev />
      </do>
      <p>
      This is card 1.
      </p>
   </card>

   <card id="card2">
      <do type="options" label="Back">
       <prev />
      </do>
      <p>
      This is card 2.
      </p>
   </card>

   <card id="card3">
      <do type="options" label="Back">
       <prev />
      </do>
      <p>
      This is card 3.
      </p>
   </card>

</wml>
```

In diesem Listing wurde jede der drei Cards erweitert, den folgenden Code aufzunehmen:

```
<do type="options" label="Back">
   <prev />
</do>
```

Dies programmiert die Options-Taste (gewöhnlich die rechte) und erzeugt zugleich eine Bezeichnung, die darüber angezeigt wird (wie in Abbildung 3.10 gezeigt). Die mit dieser Taste verknüpfte Aktion ist `<prev />`, ein WML-Tag, das wie eine Zurück-Taste funktioniert und zur vorhergehenden Card zurückkehrt. Durch die Verwendung dieser Taste ist es nun möglich, von jeder Card aus zur Card-Liste zurückzugehen.

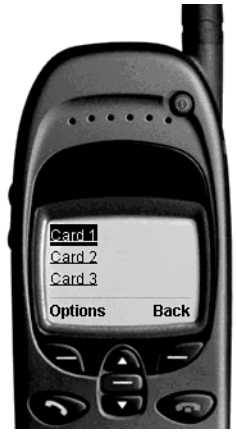

Bild 3.10: Der Template-Code wird zu jeder Card des Decks hinzugefügt.

3.3.3 Verwenden von Templates

In Listing 3.7 wird Ihnen aufgefallen sein, dass der Aktionscode für jede einzelne Card des Decks wiederholt wurde. Dies ist ausgesprochen ineffizient:

- Denselben Code immer und immer wieder eingeben zu müssen, provoziert Fehler, da es sehr wahrscheinlich ist, dass man eine Karte vergisst oder sich einmal vertippt.
- Derselbe Code an verschiedenen Stellen macht Aktualisierungen und Erweiterungen sehr kompliziert, da stets die Gefahr besteht, die eine oder andere zu übersehen.
- Der ganze zusätzliche Code vergrößert die Seite – nicht gerade ein wirtschaftlicher Umgang mit der kostbaren Bandbreite.

Um dieses Problem zu lösen, unterstützt WML die Verwendung von Template-Code, also Blöcken von Code, die auf der Deck-Ebene angegeben werden und für alle Cards dieses Decks gültig sind. Listing 3.8 enthält ein erweitertes Deck; diesmal ist der Code für die Back-Taste nicht in jeder Card angegeben, sondern in einem `<template>`-Block zu Anfang des Decks. Der gesamte Code zwischen `<template>` und `</template>` wird auf alle Cards des Decks angewendet, als wäre er dort unmittelbar eingegeben.

Listing 3.8: Verwenden von Templates

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
➥"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

   <template>
      <do type="options" label="Back">
         <prev />
      </do>
   </template>

   <card id="index">
      <p>
      <a href="#card1">Card 1</a>
      <br />
      <a href="#card2">Card 2</a>
      <br />
      <a href="#card3">Card 3</a>
      <br />
      </p>
   </card>

   <card id="card1">
      <p>
      This is card 1.
      </p>
   </card>

   <card id="card2">
      <p>
      This is card 2.
      </p>
   </card>

   <card id="card3">
      <p>
```

```
      This is card 3.
      </p>
   </card>

</wml>
```

Es gibt nur ein Problem mit diesem Code, und Sie werden sehen, was ich meine, wenn Sie sich Abbildung 3.10 anschauen. Wie Sie sehen, verfügt nun jede Card über eine Back-Taste, auch die erste (die Liste der Card-Namen), die keine haben sollte. Wenn Sie Template-Code verwenden, betrifft er jede Card des Decks, und das sind manchmal zu viele.

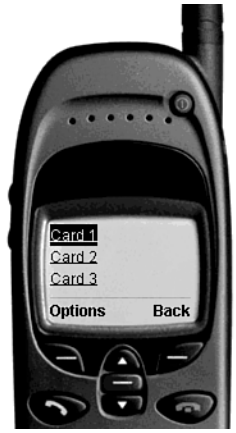

Bild 3.11: Template-Code wird auf jede Card des Decks angewendet.

Die Lösung besteht darin, eine Möglichkeit zur Verfügung zu stellen, den Code auf Card-Ebene zu überschreiben. Wenn die gleichen Aktionen auf Deck- und Card-Ebene vorkommen, genießt der Card-Code Priorität. Das bedeutet, dass wir zur Programmierung der Back-Taste den Template-Code verwenden und dem Browser mitteilen können, diesen Code nicht für die erste Card zu verwenden. Listing 3.9 enthält den verbesserten Code:

Listing 3.9: Verwenden von Templates mit Shadowing

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
➥"http://www.wapforum.org/DTD/wml12.dtd">
<wml>

   <template>
      <do type="options" label="Back">
         <prev />
      </do>
   </template>
```

```
   <card id="index">
      <do type="options">
         <noop />
      </do>
      <p>
      <a href="#card1">Card 1</a>
      <br />
      <a href="#card2">Card 2</a>
      <br />
      <a href="#card3">Card 3</a>
      <br />
      </p>
   </card>

   <card id="card1">
      <p>
      This is card 1.
      </p>
   </card>

   <card id="card2">
      <p>
      This is card 2.
      </p>
   </card>

   <card id="card3">
      <p>
      This is card 3.
      </p>
   </card>

</wml>
```

Der einzige Unterschied besteht jetzt in dem folgenden Code der ersten Card:

```
<do type="options">
   <noop />
</do>
```

Das Tag `<do>` überschreibt die Tags `<template><do>` und legt `no label` fest. Die Aktion, die mit der Taste verknüpft wird, lautet jetzt `<noop />` – oder »no operation«. Dieses merkwürdige Tag weist den Browser an, überhaupt nichts zu machen, und wird zum Shadowing verwendet – es schaltet die Optionen der Deck-Ebene für bestimmte Cards aus.

Hinweis

Nur Aktionen desselben Typs werden überschrieben. Wenn Sie auf der Deck-Ebene eine Options-Taste und auf der Card-Ebene eine Accept-Taste festgelegt haben, werden beide angezeigt – die Taste auf Card-Ebene wird die auf der Deck-Ebene nicht überschreiben, da beide von unterschiedlichem Typ sind.

Nun haben Sie ein vollständiges, funktionierendes (wenn auch ziemlich langweiliges) Deck von Cards. Die erste führt die einzelnen Cards auf, die zur Anzeige ausgewählt werden können, und jede außer der ersten verfügt über eine Back-Taste, die durch einen gemeinsam genutzten Code programmiert wurde.

3.4 Zusammenfassung

Sie haben nun erste Erfahrungen mit WML gesammelt und dabei herausgefunden, dass es wirklich ganz einfach ist, Inhalte für WAP-Browser zu entwerfen. Sie haben gelernt, dass WAP-Seiten Cards heißen und Gruppen von Cards innerhalb einer einzigen Datei Decks. Sie haben außerdem gelernt, grundlegende Formatierungs- und Navigationsfunktionen auszuführen. Diese Themen werden in den nächsten beiden Kapiteln genauer beleuchtet.