MD Consulting & Informationsdienste GmbH

Leitfaden für die Evaluierung von TD Mobile

So einfach geht es, mobile Web-Anwendung zu entwickeln

Martin Diestelmann

MD CONSULTING & INFORMATIONSDIENSTE GMBH

Inhaltsverzeichnis

Leitfaden für die Evaluierung von TD Mobile

Vorab

Vielen Dank, dass Sie die Testversion von TD Mobile aus dem Internet heruntergeladen haben. Um Ihnen einen möglichst einfachen Einstieg in die Welt der Webprogrammierung für mobile Endgeräte zu geben, haben wir eine kleine Beispielanwendung erstellt, die in diesem Papier Schritt für Schritt erläutert werden soll.

Vorarbeiten

- Führen Sie die Setup-Routine aus, wobei sichergestellt sein muss, dass Sie auf dem Arbeitsplatz Administratorenrechte haben.
- Testen Sie, nach erfolgreicher Installation, ob Sie die Entwicklungsumgebung TD Mobile starten können.

Wenn alles erfolgreich installiert ist, haben Sie TD Mobile und eine Expressversion des Internet Information Servers auf Ihrem Arbeitsplatzrechner installiert.

Um dieses Beispiel nachvollziehen zu können, müssen Sie zudem auch das Datenbanksystem SQLBase mit der Beispieldatenbank ISLAND installiert haben, wobei es egal ist, ob sich die Datenbank auf ihrem Arbeitsplatzrechner oder auf einem entfernten Rechner befindet.

Anlegen der Quellcodedatei der Beispielanwendung

Speichern Sie als erstes die leere Quellcode-Datei unter dem Namen Start in einem Verzeichnis mit dem Namen Start ab.

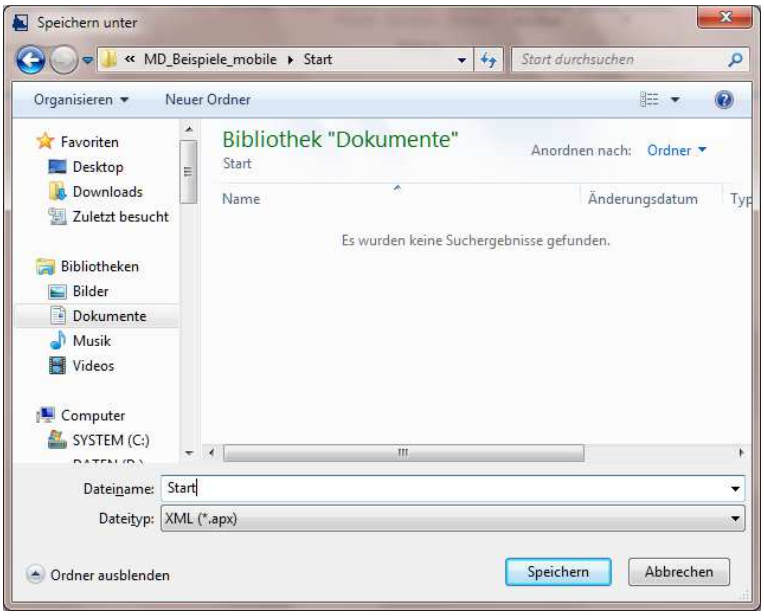


Abbildung 1: Speichern der Quellcode-Datei

Die Datei Start wird im Verzeichnis Start (eines Unterverzeichnisses von Dokumente) abgespeichert.

Testen der leeren Beispielanwendung

Wenn Sie möchten, können Sie den noch nicht editierten Quellcode der Anwendung auch testen. Klicken Sie auf den Eintrag Testen in der Multifunktionleiste (auf dem Reiter Datei in der Sektion Anwendung).

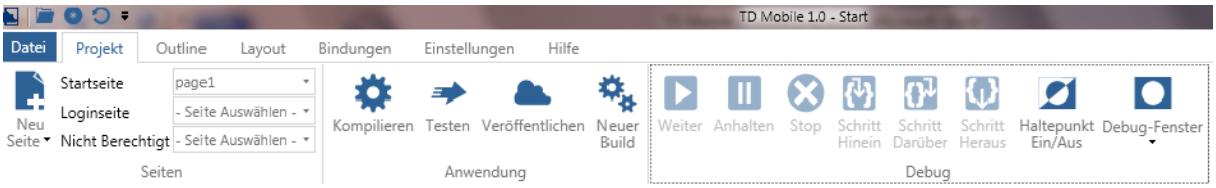


Abbildung 2: Testen der leeren Beispielanwendung

Die leere Beispielanwendung wird kompiliert und über die Express-Version des Internet Information Servers dem Entwickler für Testzwecke im Browser zur Verfügung gestellt.

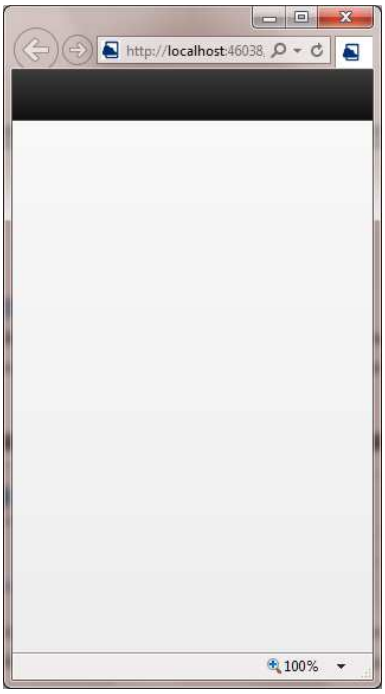


Abbildung 3: die leere Anwendung im Testmodus auf dem Arbeitsplatzrechner

Wir haben zwar noch nicht allzu viel entwickelt, aber wir können sehen, dass die leere Vorlagedatei von TD Mobile kompilierbar und ausführbar ist.

Die leere Anwendung besteht bisher aus einer Seite, wobei diese Seite auch ein derzeit leeres Titelobjekt enthält.

Design der ersten Beispielseite

Ziel

In einem nächsten Schritt soll die grundsätzliche Vorgehensweise beim Design der Bedienoberfläche einer Anwendung erläutert werden. Das Ziel ist, die nachfolgend abgebildete Anwendung (die bisher lediglich aus einer Seite besteht) zu erstellen.

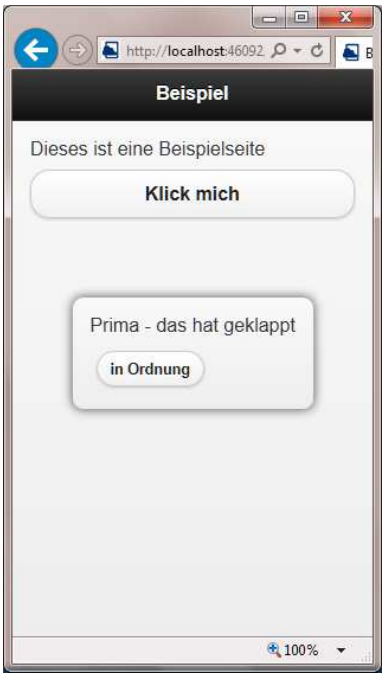


Abbildung 4: Oberfläche der Beispielanwendung

Die Beispielseite enthält einen statischen Text („Dieses ist eine Beispielseite“) und eine Schaltfläche. Wenn die Schaltfläche angeklickt wird, wird ein Dialogfenster geöffnet, in dem der Text „Prima – das hat geklappt“ und eine Schaltfläche „in Ordnung“ angezeigt wird.

Diese erste Beispielanwendung soll nun codiert werden.

Design

Bevor wir mit dem Design der Beispielanwendung beginnen (können), sollen noch ein paar grundsätzliche Bemerkungen zu TD Mobile gemacht werden.

Grundsätzliches

Standardmäßig wird das Design einer Anwendung für ein Smartphone angelegt. Diese Voreinstellung wollen wir zur Entwicklung der Beispielanwendung nutzen.

Displays

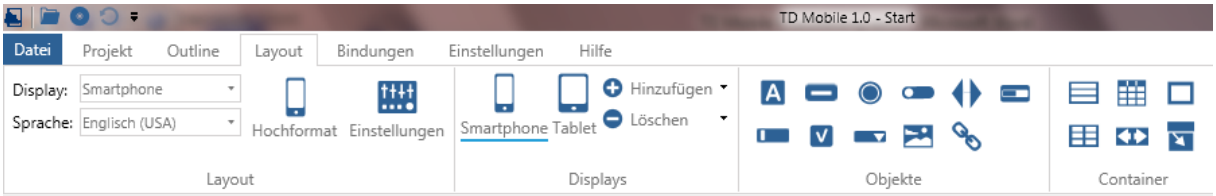


Abbildung 5: der Inhalt des Reiters Layout

In der Sektion Displays wird (durch Unterstrich) angezeigt, dass das aktuelle Layout für den Displaytyp Smartphone angelegt ist. Mit Hinzufügen kann ein weiterer Displaytyp Tablet hinzugefügt werden, sodass es grundsätzlich möglich ist, unterschiedliche Designs für verschiedene Displays einer Anwendung anzulegen.

In dieser ersten Beispielanwendung soll aber lediglich ein Design für ein Smartphone angelegt werden.

Format

Mit der Schaltfläche Hochformat in der Sektion Layout auf dem Reiter Layout kann festgelegt werden, ob das Design im Hoch- oder Querformat angelegt werden soll.

In diesem Beispiel wird das (Standard-) Hochformat für das Design der Anwendung verwendet.

Der Layout-Container

Eine Seite bietet für das Design einen „Layout-Container“ an, in dem standardmäßig alle Kind-Objekte *untereinander* angeordnet werden. Der Anwendungsentwickler braucht sich also nicht um die Anordnung der Objekte im Layout-Container (einer Seite) zu kümmern, da TD Mobile die Kindobjekte einer Seite immer untereinander anordnet.

Sprache

TD Mobile bietet die Möglichkeit, eine Internationalisierung der Anwendung vornehmen zu können. Die nationalsprachlichen Texte der Anwendung werden dafür in separaten Dateien mitgeführt. In diesem ersten Beispiel soll nicht mit der Internationalisierung der Anwendung gearbeitet werden. Dennoch soll festgelegt werden, dass die Anwendung auf Deutsch betrieben wird.

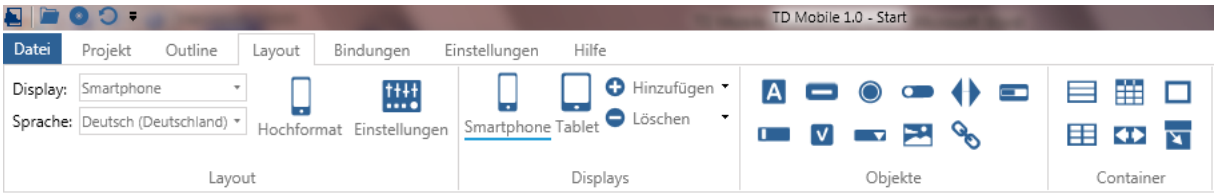


Abbildung 6: Festlegung der Sprache der Bedienoberfläche

Um Deutsch als Sprache der Anwendung festzulegen, muss auf den Reiter Datei in der Sektion Layout in der Combobox Sprache die Sprache Deutsch (Deutschland) oder eine andere nationale Variante von Deutsch ausgewählt und damit festgelegt werden.

Eine grundsätzliche, globale Einstellung kann in dem Dialog TD Mobile Einstellung vorgenommen werden, der auf dem ersten Reiter Datei und dem Menüpunkt Einstellungen aufgerufen werden kann. Die nachfolgende Abbildung zeigt den entsprechenden Dialog:

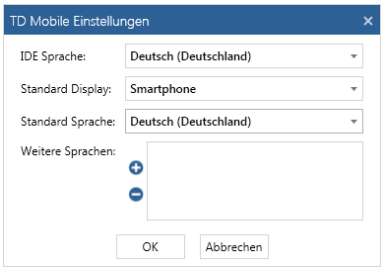


Abbildung 7: Der Dialog Einstellungen

Wie Sie sehen, kann nicht nur die Sprache der von Ihnen entwickelten Anwendungen eingestellt werden, sondern es kann auch die Bedienführungssprache von TD Mobile kann verändert werden.

In diesem Papier wird mit den abgebildeten Standard-Einstellungen für TD Mobile gearbeitet.

Design

Für das Design der Oberfläche stehen auf dem Reiter Layout alle Objekte in den Sektionen Objekte und Container zur Verfügung.

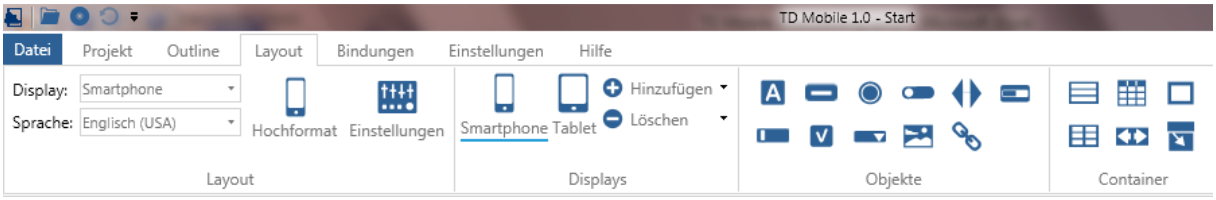


Abbildung 8: die Designobjekte für die Seitengestaltung

In diesem Beispiel verwenden wir zunächst nur das Text-Objekt und die Schaltfläche. Beide Objekte befinden sich in der Sektion Objekte.

Benennung Titel

Wechseln Sie im Arbeitsbereich auf den Reiter Smartphone Layout. Es gibt drei Reiter unten im Arbeitsbereich:

- Unter dem Reiter Outline wird die vorgegebene Codestruktur einer Seite zur Eingabe oder Änderung angezeigt
- Unter dem Reiter Smartphone Layout wird eine Näherung an das tatsächliche Layout einer Anwendung auf einem Smartphone angezeigt
- Unter dem Reiter Vorschau wird das aktuelle Design der Seite als Seitenvorschau (ohne Rahmen eines Smartphones) angezeigt.

Wir wechseln auf den Reiter Smartphone Layout.

Die Entwicklungsumgebung stellt sich uns nun folgendermaßen dar, nachdem in dem Smartphone Layout die Kopfzeile durch Anklicken ausgewählt wurde:

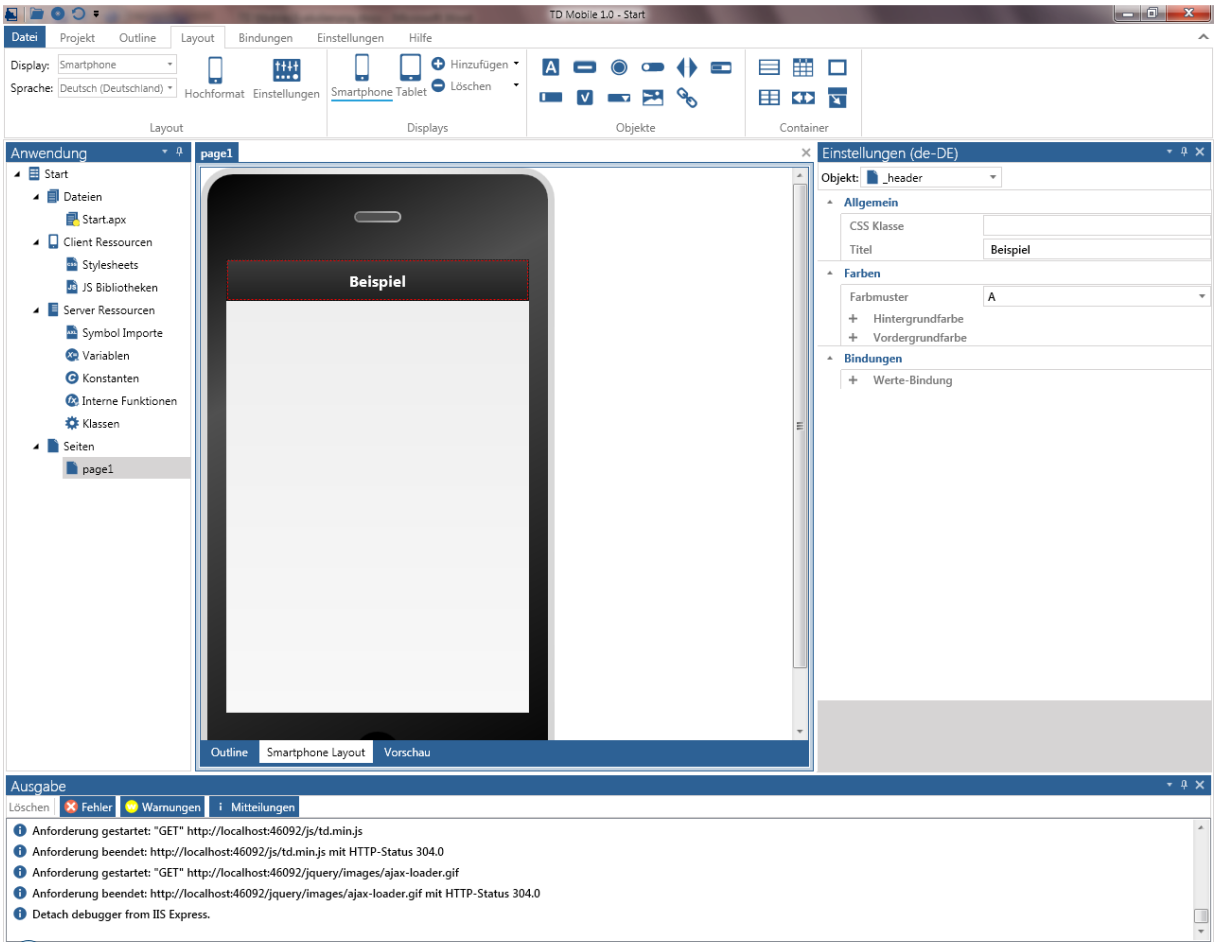


Abbildung 9: Benennung des „__header“-Objekts auf der Seite page1

Auf der linken Seite – im Fenster Anwendung – wird die Struktur des gesamten Projekts angezeigt. Der gelbe Stern unter dem Dateinamen Start.apx zeigt an, dass die Quellcode-Datei Änderungen aufweist, die noch nicht gespeichert wurden. In der Titelzeile von TD Mobile befinden sich links eine Reihe von Shortcuts für die Bedienung spezifischer Funktionen. Mit einem Klick auf die zweite Ikone wird der Quellcode gespeichert – der gelbe Stern unter dem Dateinamen verschwindet.

Im Arbeitsbereich – das mittlere Fenster von TD Mobile – wird die Seite im Smartphone Layout angezeigt.

Im rechten Fenster Eigenschaften können die Eigenschaften eines ausgewählten Objekts angesehen und gegebenenfalls geändert, d.h. an die Vorstellungen des Entwicklers angepasst werden.

Beispielsweise verfügt das (standardmäßig auf jeder Seite vorhandene) Header-Objekt mit dem Namen __header über folgende Eigenschaften:

- CSS-Klasse: soll das Objekt ein vollkommen anderes Aussehen erhalten, so muss in einem eigenen Cascading Style Sheet eine Klasse definiert und der Name der Klasse dem Header-Objekt zugewiesen werden. Im Rahmen der Evaluierung werden wir auf diese Funktionalität nicht weiter eingehen.
- Titel: als Eigenschaft kann dem Objekt ein Titel zugeordnet werden. In dieser Anwendung soll der Titel Beispiel heißen. Das Wort Beispiel wird daher in das Feld Titel eingetragen. Damit ist der Seitentitel statisch hinterlegt. Wie ein Seitentitel dynamisch, d.h. während des Programmablaufs geändert werden kann, ist nicht Bestandteil dieses Papiers.

- Farbmuster: TD Mobile bietet fünf Farbmuster (Swatches) mit den Namen A, B, C, D und E an. Ein Farbmuster definiert aufeinander abgestimmt das Aussehen aller Objekte (und Container) in einer Anwendung. Grundsätzlich kann durch die zusätzliche Option für Kindobjekte „Eltern verwenden“ das Aussehen eines Toplevel-Objekts (einer Seite) an seine Kind-Objekte (Objekte und Container) „vererbt“ werden. Es ist aber auch möglich, innerhalb einer Anwendung den vordefinierten Stil eines Objekts durch Zuweisung anderer Farbmuster zu mischen. Darüber hinaus kann die Vordergrund- und die Hintergrundfarbe eines Objekts über einen RGB-Mischer individuell angepasst werden.[1] Wir verzichten im Rahmen dieses Papiers auf eine Abweichung vom vorgegebenen Standard von TD Mobile.
- Bindung: das Thema Bindung wird weiter unten (siehe: Vorüberlegungen (Bindungen), Seite 20) in diesem Papier ausführlicher behandelt.

Dem Header-Objekt __header wird also lediglich der Titel Beispiel zugewiesen.

Design des Text-Objekts

Als nächstes soll ein Text-Objekt angelegt und mit einem statischen Text belegt werden. Das Design der Anwendung wird dann folgendermaßen aussehen.

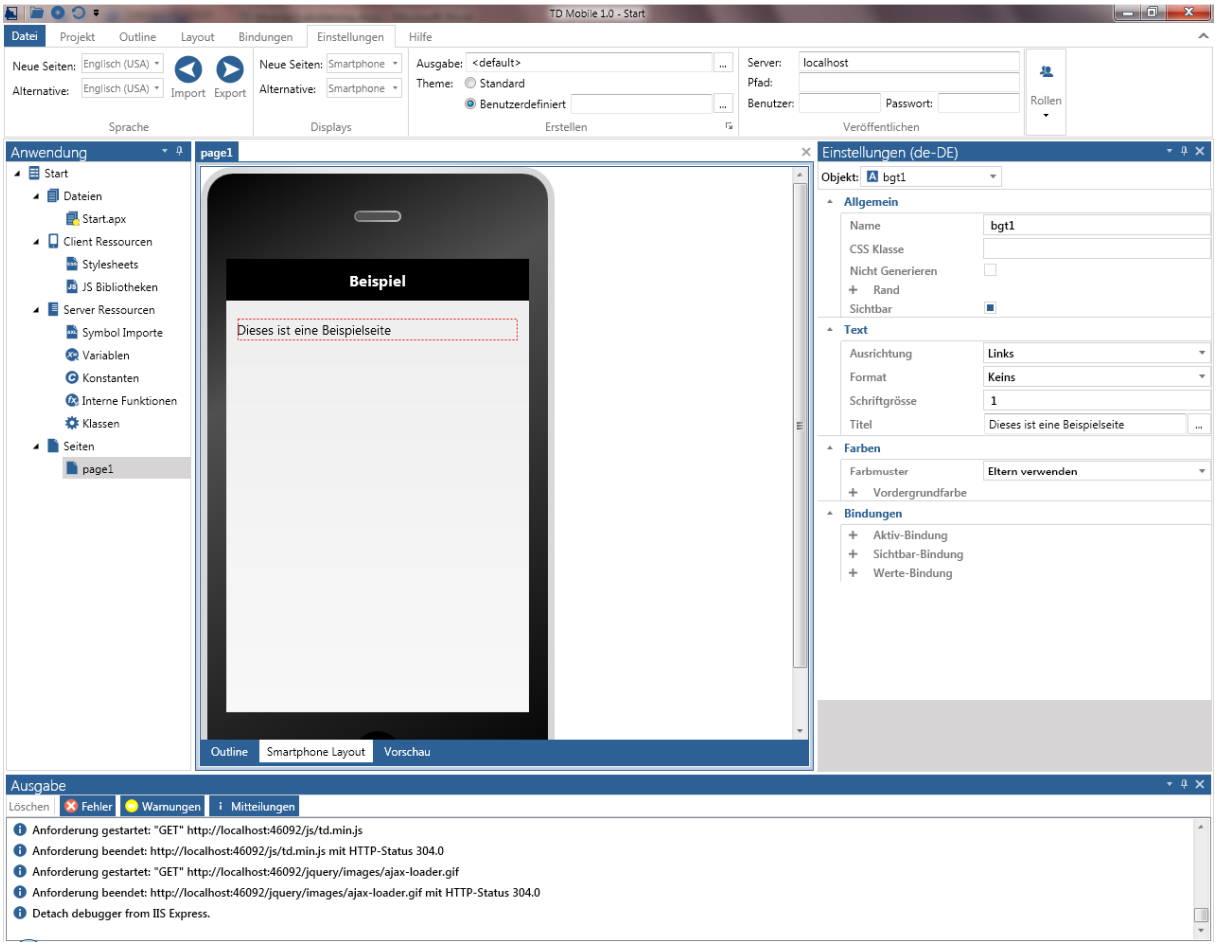


Abbildung 10: Design nach Anlegen des Text-Objekts

[1] Auf der Seite themeroller.com kann man weitere individuelle Farbmuster erzeugen, die Ergebnisdatei herunterladen und das entstandene Cascasding Style Sheet unter dem Reiter Einstellungen, Erstellen das indivuelle Farbmuster in die TD Mobile Anwendung einbinden.

Als erstes klicken wird das Objekt Text in der Sektion Objekte in der Multifunktionsleiste an und ziehen es – mit weiterhin gedrückter Maustaste – auf das SmartPhone Layout und lassen die Maustaste los: als Ergebnis wird ein (leeres) Textobjekt mit der Standard-Formatierung auf der Vorschau platziert.

In den Einstellungen wird in der Sektion Text im Feld Titel der Text „Dieses ist eine Beispielseite“ eingegeben. Diese Einstellungen genügen um die vorgesehene Beispielanwendung zu erstellen.

Anlegen der Schaltfläche

Als nächstes wird das Schaltflächen-Objekt in der Multifunktionsleiste Layout, Objekte ausgewählt und mit gedrückter Maustaste unter das Text-Objekt in dem Smartphone Layout gezogen und die Maustaste losgelassen: es befindet sich nun eine Schaltfläche auf der Oberfläche. Als Titel des Schaltflächen-Objekts wird Klick mich eingegeben.

Das Design der Beispielanwendung sieht nunmehr folgendermaßen aus:

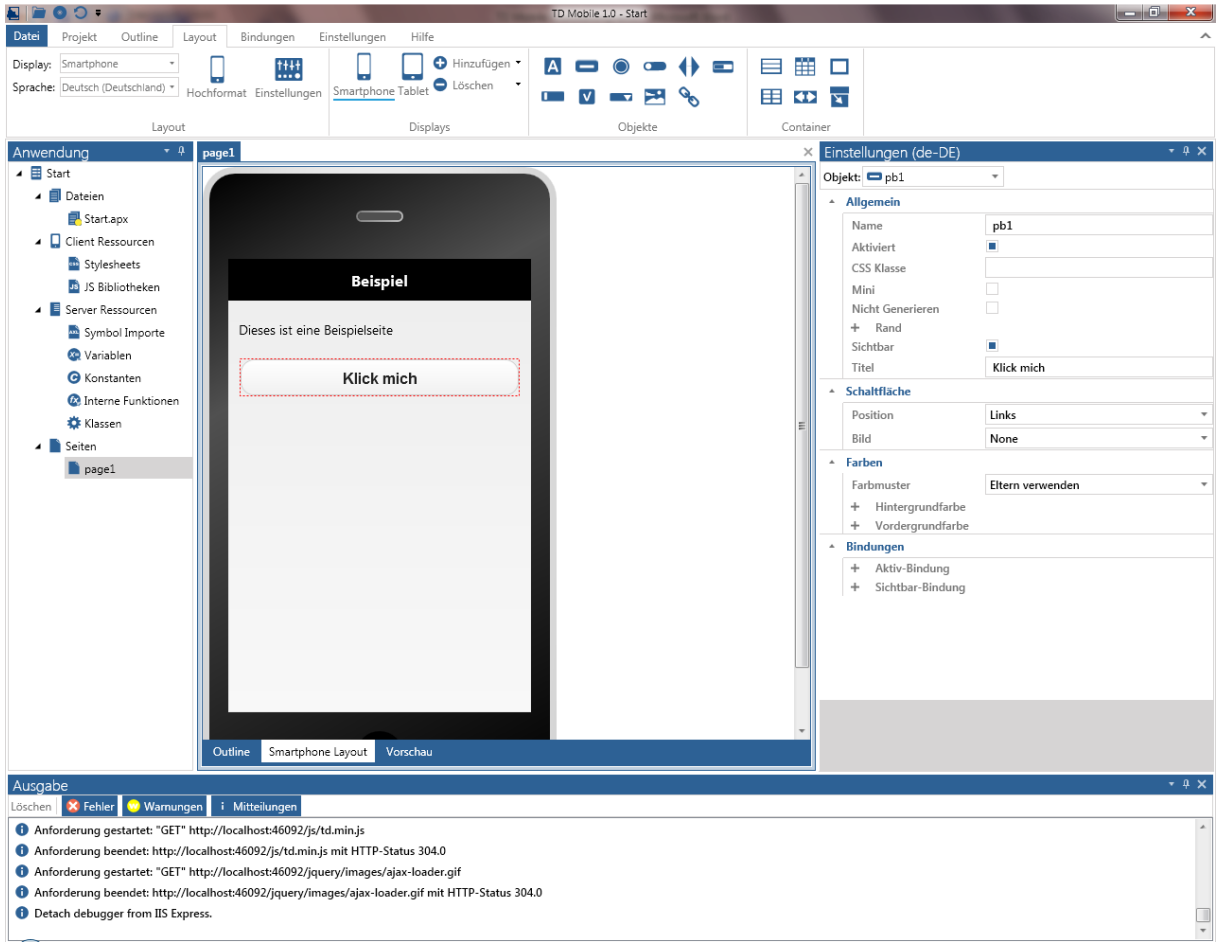


Abbildung 11: Design der Schaltfläche

Zusammenfassung Design

Beim Design einer Seite fungiert die Seite als Layout-Container, der alle Objekte automatisch untereinander anordnet. Bei einigen Container-Objekten, die wir bisher noch nicht verwendet haben, kann auch eine horizontale Anordnung bzw. eine komplexere Anordnung definiert werden. Im Rahmen des ersten Beispiels wird auf ein komplexeres Layout verzichtet.

Die auf einer Seite platzierten Objekte verfügen über eine Reihe von Eigenschaften, die noch nicht vollständig für alle Objekte vorgestellt wurden. Dieses wird in diesem Einführungspapier auch nicht vorgenommen.

Das Design der ersten Seite ist abgeschlossen – allerdings sollte auf dieser ersten Seite auch eine (clientseitige) Aktion vorgenommen werden können: nach dem Klick auf die Schaltfläche soll eine Nachricht angezeigt werden.

Codierung einer Aktion

Um eine Aktion codieren zu können, muss auf die Sicht Outline gewechselt werden. Die nachfolgende Abbildung zeigt im Arbeitsbereich die (vordefinierte) Struktur einer Seite.

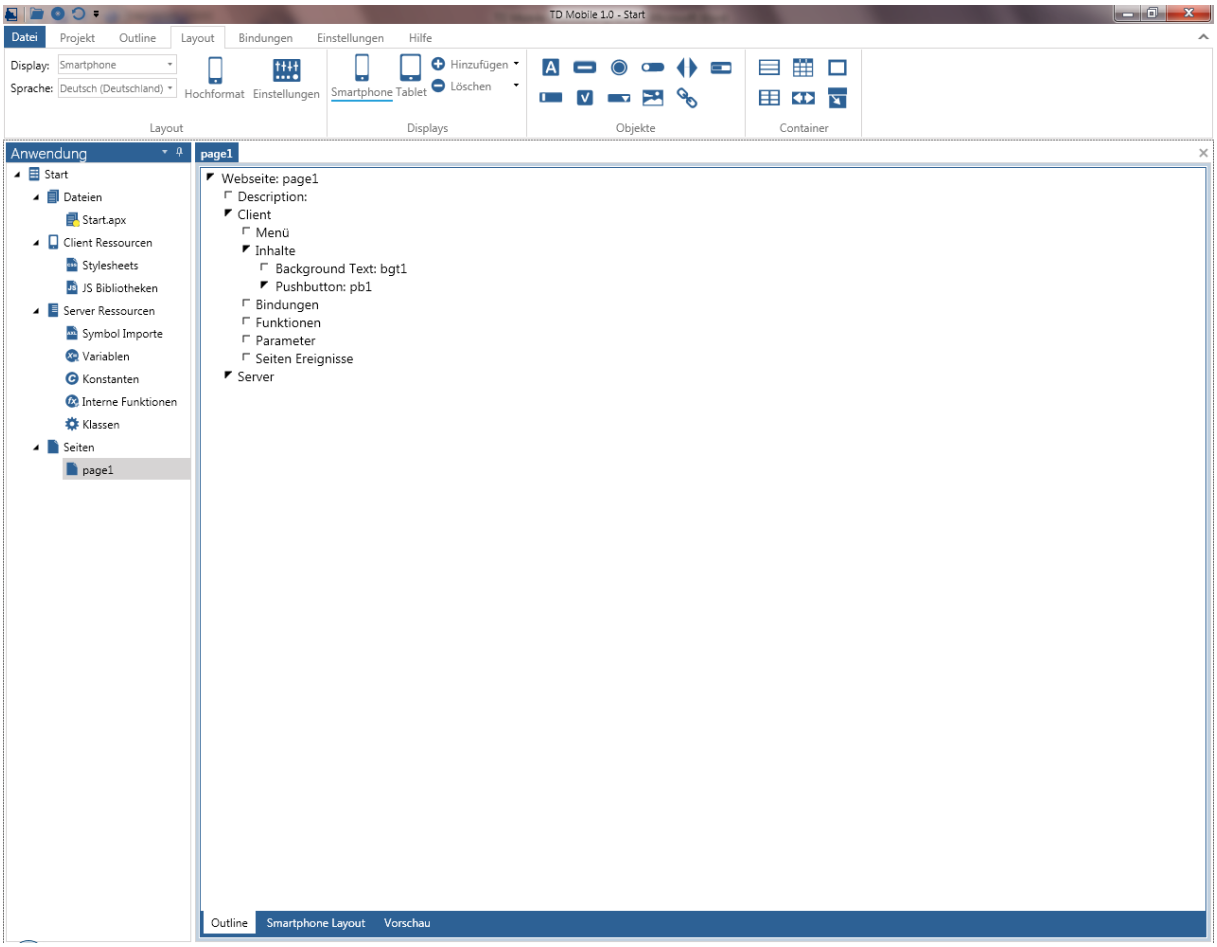


Abbildung 12: Sicht auf die Codestrukur einer Seite

Eine Seite (Webpage) besteht aus einem Client- und einem Server-Teil. Der Client-Teil einer Seite wird in der laufenden Anwendung aus HTML (5) Layout, möglicherweise modifiziert durch ein eigenes Stylesheet und vordefinierten und eigenen Javascript und JQuery Mobile Scripten bestehen, während im Server-Teil Operationen codiert werden, die in der ablauffähigen Anwendung als JSON-Webservices auf einem Internet Information Server bereitgestellt werden.

Im Rahmen des ersten Beispiels wird lediglich eine in TD Mobile vorkonfigurierte Aktion verwendet. Bei der Schaltfläche pb1 wird ein Doppelklick ausgeführt – der Knoten wird expandiert und es wird die Sektion Ereignisse angezeigt.

Ein Doppelklick auf Ereignisse oder die Taste Einfg führt dazu, dass die bei der Schaltfläche definierten Ereignisse angezeigt werden. Wir wählen die Option Click aus. Nach Drücken der Taste Einfg wird das Kontextmenü angezeigt, in der alle in TD Mobile vorkonfigurierten (Client-)Aktionen zur Auswahl angezeigt werden.

Die nachfolgende Abbildung zeigt das Kontextmenü mit allen in TD Mobile vorkonfigurierten (Client-) Aktionen.

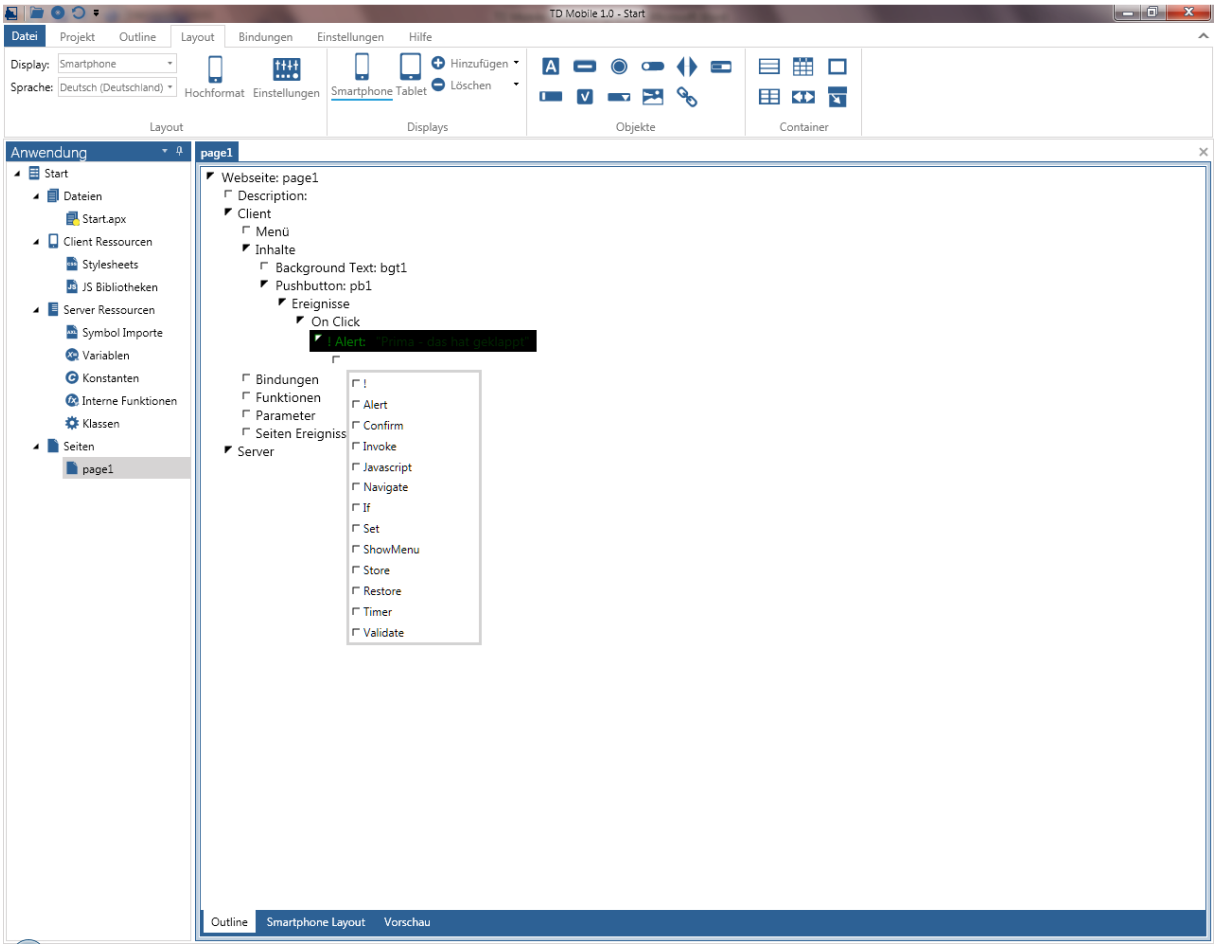


Abbildung 13: Kontextmenü mit allen in TD Mobile verfügbaren Client-Aktionen

Wir wählen die Option Alert aus und fügen (in doppelten Anführungszeichen) den Text „Prima – es hat geklapt ein“. Die Aktion Alert zeigt einen (HTML formatierten) Text in einem Standard-Dialog mit einer Schaltfläche an.

Wie man im Kontextmenü sehen kann, stellt TD Mobile eine Reihe von vorkonfigierten Client-Aktionen zur Verfügung, wobei wir bisher lediglich die Alert-Aktion behandelt haben.

- Eine andere Variante eines Dialogs ist Confirm. Bei dieser Aktion wird ebenfalls ein Dialog, in diesem Fall aber mit zwei Schaltflächen angezeigt.
- Invoke ist eine sehr wichtige Aktion: nach Invoke muss eine serverseitige Aktion angegeben werden, d.h. es wird ein JSON-Webservice aufgerufen.
- Javascript: mit dieser Aktion wird ein vom Entwickler (in der Client-Sektion Funktionen) definiertes Javascript ausgeführt.
- Navigate: mit dieser Aktion wird eine andere Seite aufgerufen. (siehe unten)

- If: mit dieser Aktion kann eine Bindung (siehe unten) geprüft und damit eine clientseitige Programmsteuerung vorgenommen werden.
- Mit Set kann der Wert einer Bindung (siehe unten) clientseitig gesetzt werden.
- Show Menu: mit dieser Aktion wird ein auf der gleichen Seite definiertes Menü angezeigt.
- Store und Restore: mit dieser Aktion können Werte bzw. Wertestrukturen im Offline-Betrieb in dem Browser-Speicher zwischengespeichert werden. Diese Aktionen werden im Rahmen dieser Papiers nicht behandelt.
- Mit dieser Aktion kann eine Veränderung auf einer Seite zeitgesteuert vorgenommen werden. Diese Aktion wird im Rahmen dieses Papier nicht behandelt.
- Validate: mit dieser Aktion kann die Validierung eine Benutzereingabe (z.B. vor der Weitergabe an einen Webservice nach den in TD Mobile hinterlegten Regeln erzwungen werden.

Wichtig ist das clientseitige Aktionen asynchron ausgeführt werden. Soll eine synchrone Ausführung von mehreren Aktionen sichergestellt werden, so müssen diese Aktionen im Quellcode untereinander eingerückt codiert werden.

Zusammenfassung

Das Design und die Codierung der ersten Beispielanwendung ist abgeschlossen. Die Anwendung kann getestet werden. Klicken Sie auf die Schaltfläche Testen.

Abbildung 14: Die im Testmodus gestartete Bespielanwendung

Veröffentlichen der Anwendung

Um eine Anwendung auf einem Microsoft Internet Information Server bereitzustellen, muss die Anwendung veröffentlicht werden. Bevor die Veröffentlichung vorgenommen werden kann, müssen die Einstellungen für die Veröffentlichung gesetzt werden.

Starten Sie TD Mobile aus dem Start-Menü auf jeden Fall als Administrator – es reicht nicht aus, nur die Administratorrechte zu haben. Laden Sie dann ihre Source Code-Datei und wechseln Sie in der Multifunktionsleiste auf den Reiter Einstellungen.

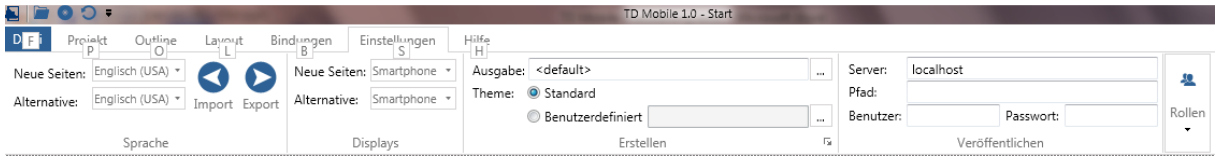


Abbildung 15: der Reiter Einstellungen

Geben Sie den Servernamen, den Pfad, wo das Projekt unter wwwroot veröffentlicht werden soll und ggf. den Administrator-Namen und sein Passwort ein.

Mit diesen Angaben wird die Anwendung auf dem Webserver publiziert: im Browser des Smartphones muss nun die IP-Adresse des Weebservers gefolgt von / und Start eingegeben werden:

[IP-Adresse des Webservers]/Start

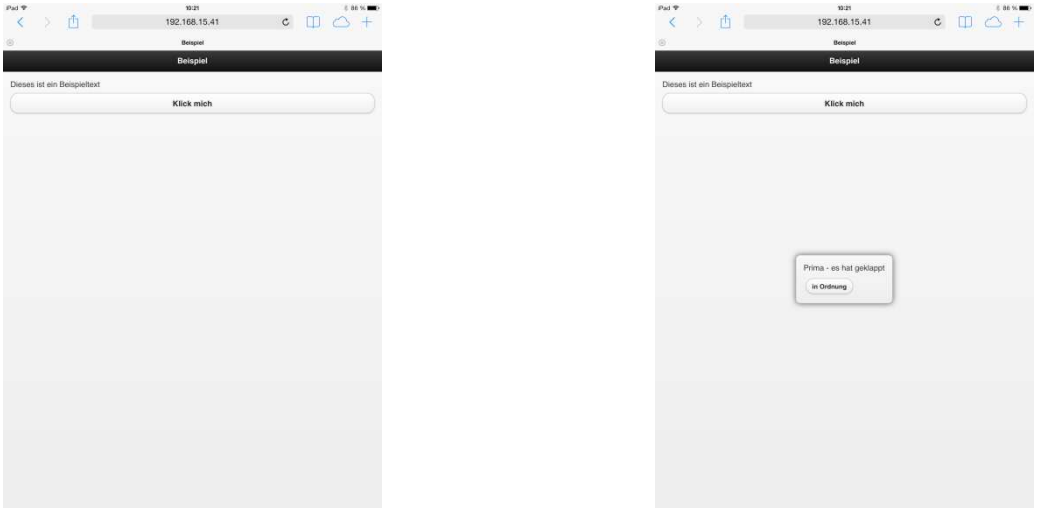


Abbildung 16: die beiden Zustände der Anwendung (vor und nach dem Klick auf die Schaltfläche9

Die zweite Beispielanwendung

In diesem Abschnitt werden die Themen Bindungen, Datenbankzugriffe und komplexere Oberflächen-Objekte behandelt.

Die Anwendung

Nach dem Start der Anwendung wird die erste Seite der Anwendung mit Angaben zu allen Firmen aus der Tabelle COMPANY (in drei untereinander stehenden Zeilen) angezeigt. Klickt der Anwender auf einen Eintrag in der Liste, wird eine zweite Seite mit Angaben zu den Ansprechpartnern aus der Tabelle CONTACT angezeigt.

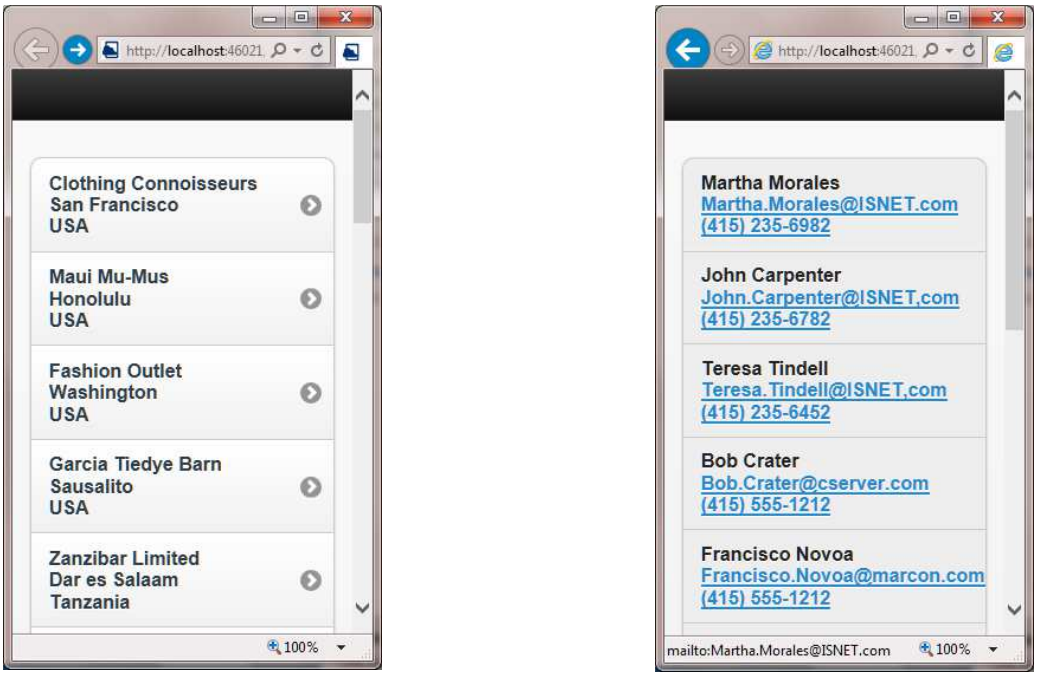
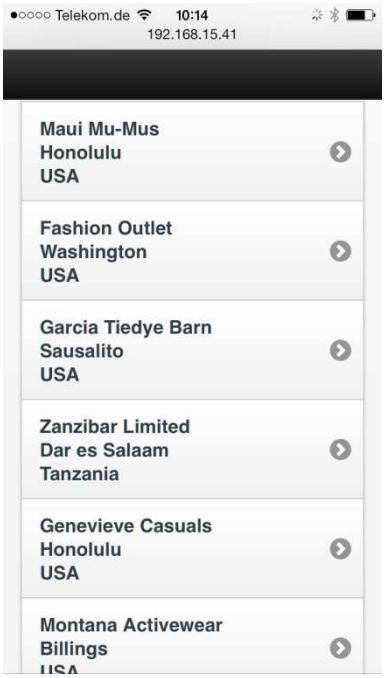
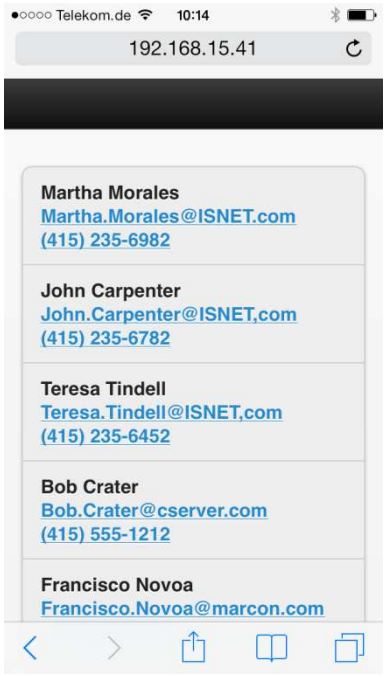


Abbildung 17: die zwei Seiten der zweiten Beispielanwendung

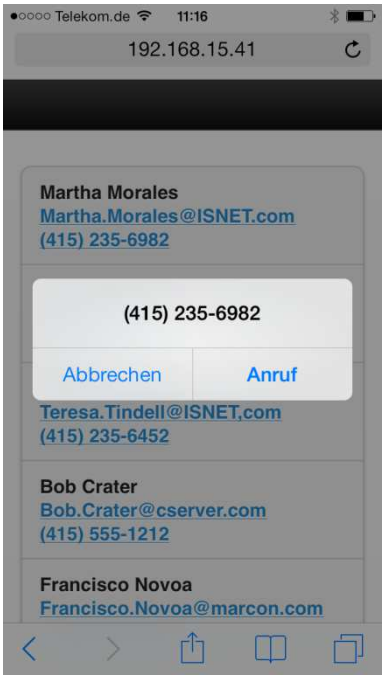
Die zweite Beispielanwendung bietet folgende Funktialitäten an:



Auf der ersten Seite werden alle Firmen aus der Tabelle COMPANY (und weitere Angaben der Adresse) angezeigt. Ein Eintrag in der Liste kann angeklickt werden. Nach einem Klick wird die Seite page2 angezeigt, auf der alle Ansprechpartner der ausgewählten Firma angezeigt werden.



Auf der Seite page2 werden die Namen der Ansprechpartner, deren E-Mail-Adresse und deren Telefonnummer angezeigt. Die beiden letzten Einträge sind Links, die angeklickt werden können.



Nach der Auswahl einer Telefonnummer wird die Nummer in einem Smartphone herangezogen, um die eingeblendete Telefonnummer anzurufen. Auf einem Tablet (ohne Telefonkarte) wird die Nummer angeboten, um eine SMS an die Telefonnummer zu schicken.

Nach Anklicken einer E-Mail-Adresse wird das Mail-Programm auf dem Smartphone (oder Tablet) aufgerufen, eine neue E-Mail erzeugt und als Empfänger wird die ausgewählte E-Mail-Adresse eingesetzt.

Vorüberlegungen (Bindungen)

Auf der ersten Seite sollen die Angaben Firma, Stadt und Land einer Firma angezeigt werden. Die Angaben befinden sich in der Datenbank ISLAND in der Tabelle COMPANY (COMPANY_NAME, CITY, COUNTRY). Es sollen mehrere Angaben angezeigt werden.

Das wichtige Konzept in TD Mobile, mit dem Daten vom Server mit Daten an der Oberfläche verbunden werden, heißt in TD Mobile Bindungen. Bindungen können einfache Datentypen (Boolean, Binary, String, usw.), aber auch komplexe Datentypen als Instanz einer Klasse sein.

Wichtig für das erste Beispiel mit Daten aus der Datenbank ist, dass bei der (serverseitigen) Ausführung von SELECT-Anweisungen selbstverständlich mehrere Datensätze selektiert und über eine Webserver Operation an die Client-Anwendung übergeben werden können. Daher ist es auch in TD Mobile erlaubt, Arrays von einfachen und komplexen Datentypen zu verwenden und einer (Client-) Bindung zuzuweisen.

TD Mobile kennt eine vier Arten von Bindungen:

Name	Art
Aktiv-Bindung	Ist eine Bindung initialisiert, wird das Objekt bedienbar (enabled) geschaltet
Listen-Bindung	Eine Liste wird in der Regel ein Array von Bindungen zugewiesen
Sichtbar-Bindung	Ist eine Bindung initialsiiert, wird das entsprechende Objekt sichtbar gemacht
Werte-Bindung	Eine Werte-Bindung ist entweder ein Datensatz oder – bei einfachen Datentypen ein einzelner Wert

Tabelle 1: Bindungstypen in TD Mobile

Im Folgenden werden zwar nicht alle Bindungstypen im Einzelnen vorgestellt, es wird aber vorgestellt, wie man insbesondere mit Listen- und Werte-Bindungen arbeitet.

Design der ersten Seite – erste Phase

Auf der ersten Seite page1, die standardmäßig mit einem neuen Projekt angelegt wird, wird als erstes aus der Sektion Container (Layout, Container) eine Liste ausgewählt und auf die Seite gezogen. Eine Liste dient dazu, mehrere gleiche Daten(sätze) anzuzeigen.

In die Liste wiederum wird ein Layout Grid gezogen und in den Eigenschaften festgelegt, dass das Layout Grid drei Zeilen und (lediglich) eine Spalte enthalten soll.

Hiermit ist das Design der ersten Seite vorläufig abgeschlossen – wie werden das Design der ersten Seite abschließen, wenn wir die Codierung abgeschlossen haben.

Codierung der ersten Seite, Setzen der Bindungen

Um Daten aus der Datenbank auszulesen und an den Client zu übertragen, müssen wir zunächst folgende Schritte erledigen:

- Die Verbindungsaufnahme mit der Datenbank codieren
- Die Struktur der INTO-Variablen einer SELECT-Anweisung definieren und implementieren

Die wiederverwendbare interne Funktion MitDatenbankVerbinden

Da in der Anwendung mehrfach auf die Datenbank zugriffen werden soll, definieren wir uns eine interne Funktion mit dem Namen MitDatenbankVerbinden. Der Code der internen Funktion ist nachfolgend dargestellt.

```
Set sVerbindungsstring =
"servername=server1;hostname=localhost;port=2155;database=island;user=sysadm;password=sysadm;poolsize=10;c
onnectionlifetime=20"
If SqlConnectDotNet(p_hSql, sVerbindungsstring, "", 9)
   Set bOk = TRUE
Else
   Set bOk = FALSE

Return bOk
```

Source Code 1: die interne Funktion MitDatenbankVerbinden

Wichtig ist, dass TD Mobile die Funktion SqlConnnectDotNet() zur Verfügung stellt, in der über einen Connection-String als erstem Parameter alle notwendigen Informationen zur Verbindungsaufnahme mit der Datenbank zur Verfügung gestellt werden.

In diesem Beispiel wird ein statischer Connection-String übergeben – in der Praxis werden beispielsweise die Informationen zu username und password vermutlich aus der Bedienoberfläche (über eine entsprechende Bindung) übergeben und in den Connection-String „eingearbeitet“ werden.

Wenn die Verbindungsaufnahme zur Datenbank funktioniert, gibt die Funktion (als Rückgabewert) TRUE zurück.

Abbildung eines Datensatzes als Vorlage für INTO-Variablen

Nachdem wir eine interne Funktion definiert haben, mit der die Verbindungsaufnahme zur Datenbank vorgenommen werden kann, soll diese Funktion ja unter anderem verwendet werden, um die Firmenangaben aus der Datenbank auszulesen. Hierfür benötigen wir eine Struktur eines Datensatzes, um später einen Array dieser Struktur von der Serveroperation an die Client-Anwendung – genauer: an die eine entsprechende Bindung – auf der Seite zu übertragen.

Wir legen daher unter Klassen eine Funktionale Klasse mit dem Namen clsFirma an und definieren drei Strings, die wir Firma, Stadt und Land nennen.

Diese Klasse clsFirma wird auf der Seite page1 *zweifach* als Bindung instantiiert: unter dem Namen EINEFIRMA wird eine Struktur (von clsFirma) definiert, während unter dem DIEFIRMEN[*] ein Array dieser Struktur definiert wird. Was wir mit diesen Bindungen machen, wird klarer werden, wenn wir die (serverseitige) Operation FirmenEinlesen() erläutert haben.

```
Operation: FirmenEinlesen
   Description:
   Parameter:
   Returns:
      clsFirma:[*]
         Binding:DIEFIRMEN
   Lokale Variablen:
      Sql Handle: hSql
      String: sStatement
      Number: nReturn
      Number: n
      clsFirma: oFirma[*]
   Aktionen
      If MitDatenbankVerbinden(hSql)
         Set sStatement = "SELECT COMPANY_ID, COMPANY_NAME, CITY, COUNTRY
         FROM COMPANY ORDER BY 1
```

```
    INTO :oFirma[n].Firmennr, :oFirma[n].Firmenname, :oFirma[n].Stadt, :oFirma[n].Land "
    If SqlPrepareAndExecute(hSql, sStatement)
       Set n = 0
       While SqlFetchNext(hSql, nReturn )
          Set n = n + 1
       If SqlDisconnect(hSql)
          Return oFirma
```

Source Code 2: die Server-Operation FirmenEinlesen

Als Rückgabewert (Returns) wird festgelegt, dass eine Struktur von clsFirma zurückgegeben wird, wobei durch [*] festgelegt wird, dass es sich um ein (dynamischen) Array von Strukturen handelt. Als Bindung wird die clientseitig definierte (Array-) Bindung DIEFIRMEN festgelegt. So einfach ist es, eine Verbindung zwischen Daten einer Operation auf dem Webserver mit einer clientseitigen Bindung zu verbinden.

Sollte die Bindung beim Return-Bindung nicht auswählbar sein, wurde sie sicherlich noch nicht im Client-Teil der Seite als Array-Bindung definiert.

Als lokale Variablen sind zum Einen die Standard-Variablen für den Datenbankzugriff hSql, sStatement und nReturn vereinbart. Zusätzlich wurde ein Array der Struktur clsFirma mit dem Namen oFirma definiert und eine Laufvariable n.

Wenn die interne Funktion MitDatenbankVerbinden() erfolgreich ausgeführt werden konnte, so ist im Aktion-Teil der Operation festgelegt, wird die SELECT-Anweisung (als String) definiert, wobei als INTO-Variablen der SELECT-Anweisung ein Array-Eintrag der Struktur clsFirma deklariert wird.

Um die Ergebnismenge in unterschiedliche Array-Elemente zu lesen, braucht beim SqlFetchNext () lediglich die Laufvariable n erhöht zu werden.

Nachdem die Ergebinsmenge der SELECT-Anweisung aus der Datenbank in die Operation (genauer: in Array-Elemente des Objekts oFirma) eingelesen wurden, kann die Verbindung zur Datenbank beendet werden: die komplette Ergebnismenge wird als Rückgabewert definiert und wird – da die Bindung an DIEFIRMEN hinterlegt wurde – an die Client-Bindung zurückgegeben. Soweit sind wir aber noch nicht, da die Operation FirmenEinlesen() zwar definiert ist, aber noch nicht aufgerufen wird.

Aufruf und Visualisierung der Bindung(en)

Bei den Seiten Ereignissen der Seite page1 wird das Ereignis Create abgefangen und dort die Aktion Invoke, d.i. Aufrufen einer Webservice Operation hinterlegt.

```
Invoke FirmenEinlesen
```

Source Code 3: Aufruf der Operation FirmenEinlesen

Nun haben wir einen Zustand bei der Codierung erreicht, bei dem die Seite generiert, der entsprechende Code an den Browser geschickt wird, aber noch keine Daten angezeigt werden. Das liegt daran, dass wir die Bindungen DIEFIRMEN und EINEFIRMA noch nicht den entsprechenden Anzeigeobjekten zugewiesen haben.

Wir kehren also noch einmal zum Design der Seite zurück, um die entsprechenden Bindungen bei den Objekten zu setzen.

Design der ersten Seite – zweite Phase

Die nachfolgende Abbildung zeigt das Design der ersten Seiten im Smartphone Layout und das Eigenschaftenfenster der Liste lv1.

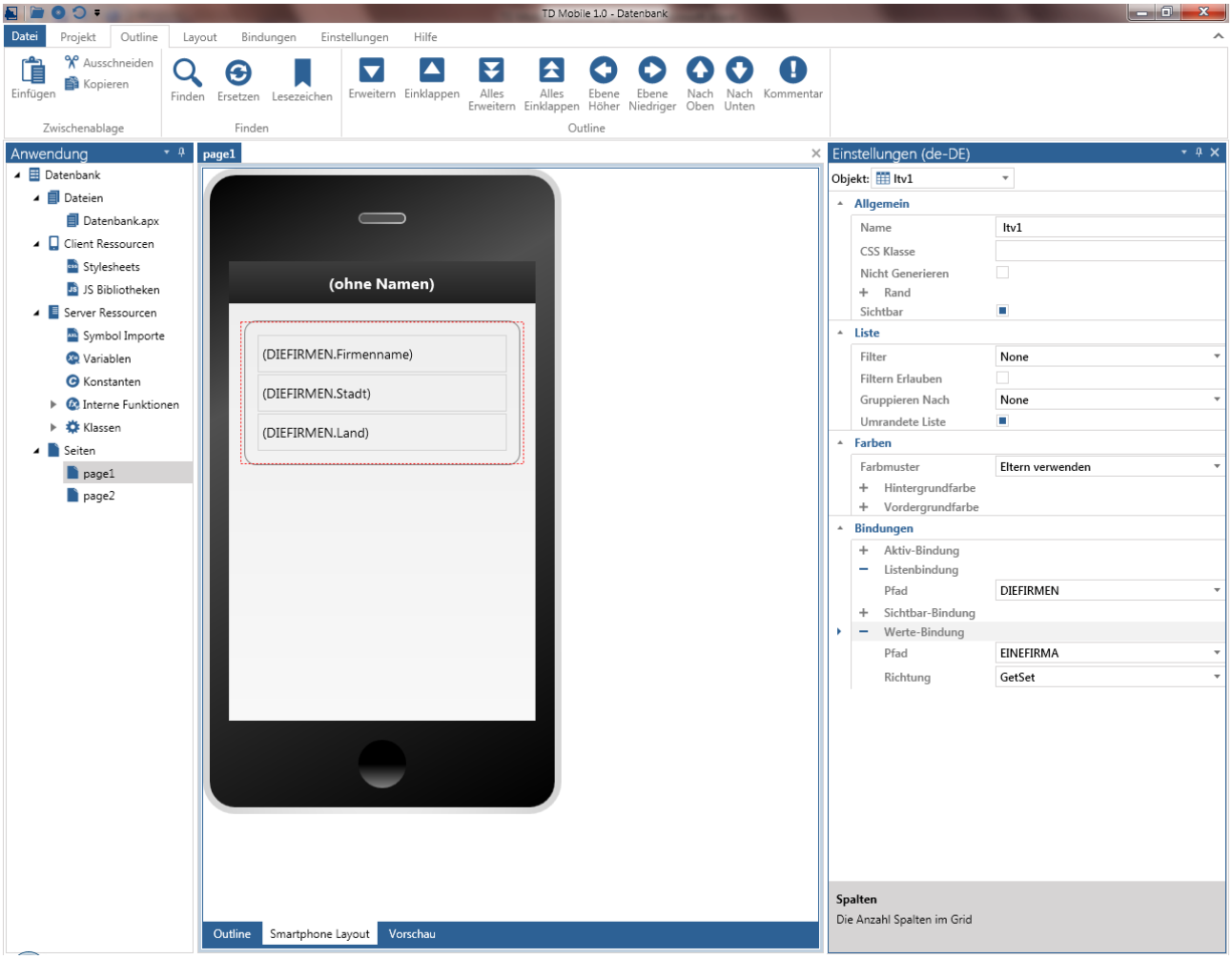


Abbildung 18: Zuweisung der Bindungen an den Listen-Container

Dem Listen-Container wird zur Visualisierung der Ergebnismenge die Listenbindung DIEFIRMEN und die Werte-Bindung EINEFIRMA zugewiesen. In die drei Zeilen der Liste werden nacheinander Text-Objekte gezogen, platziert und als Wertebindung ein Element der Struktur DIEFIRMEN – im abgebildeten Beispiel DIEFIRMEN.Firmenname – zugewiesen.

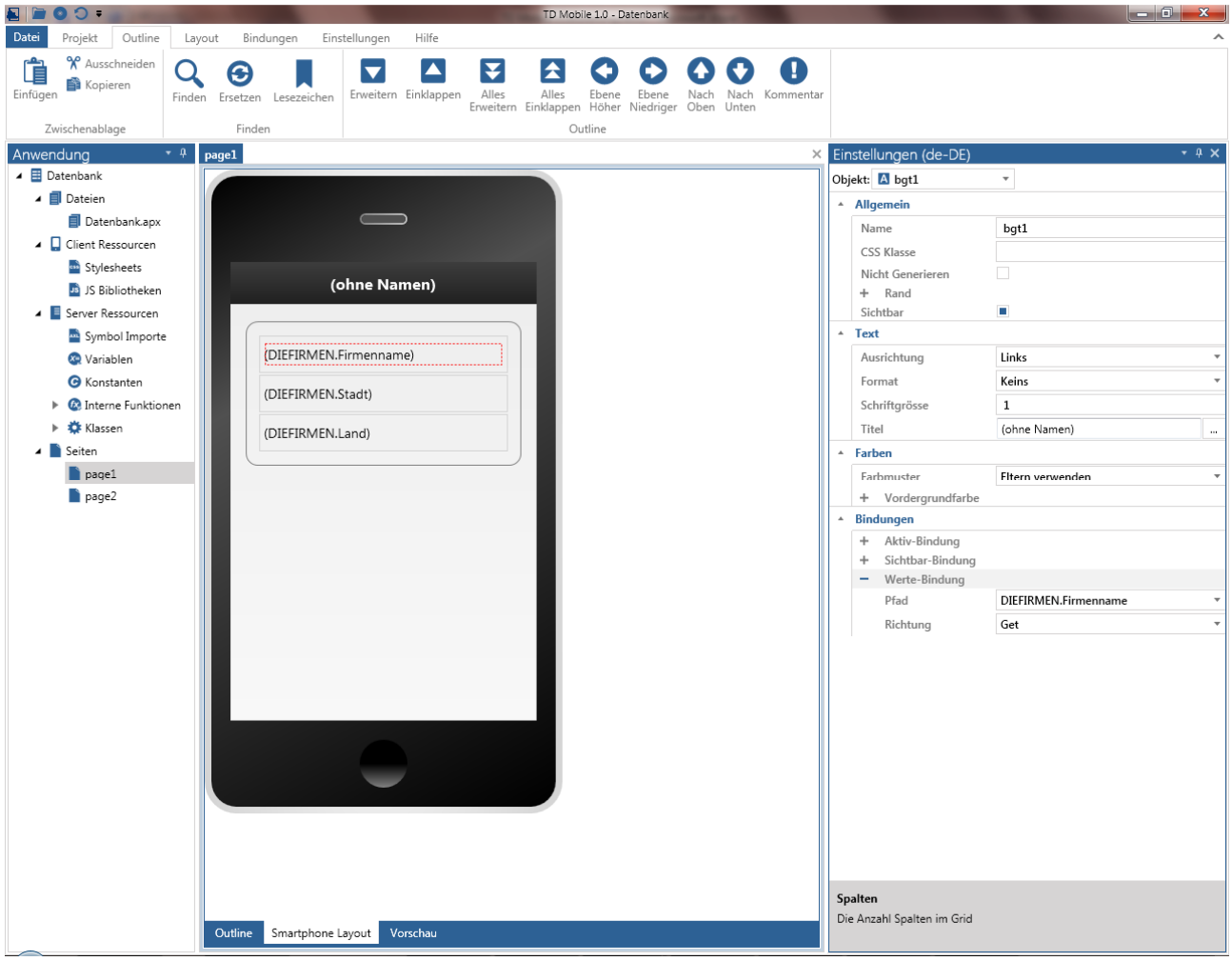


Abbildung 19: Zuweisung der Werte-Bindung an das Objekt bgt1 (Firmenname)

Die zweite Beispielanwendung ist fertig designed und codiert – sie kann getestet werden.

Die nachfolgende Abbildung zeigt die Seite 1 der Beispielanwendung im Testmodus.

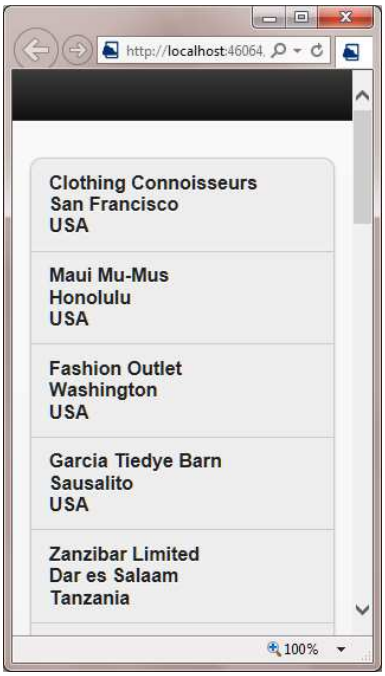


Abbildung 20: die zweite Beispielanwendung im Testmodus

Zusammenfassung

In diesem, etwas komplexeren Beispiel wurde gleich mit komplexen Datentypen gearbeitet, da für die Ausführung einer SELECT-Anweisung für die Rückgabe einer Ergebnismenge beliebig viele Instanzen einer solchen Struktur benötigt werden. Die Klasse clsFirma wurde als Funktionale Klasse mit drei Instanzvariablen definiert und mehrfach verwendet:

1. Als Client-Bindung zur Verwaltung eines Arrays von Strukturen DIEFIRMEN[*]
2. Als Client-Bindung zur Referenzierung eines einzelnen Datensatzes EINEFIRMA
3. Als Bindung des Rückgabewerts als Array in der Operation FirmenEinlesen() und
4. Als Array einer lokalen Variable oFirma in den Aktionen der Operation, um sich durch Erhöhung der Laufvariable n dynamisch zu vergrößern

Die Client-Bindungen DIEFIRMEN[*] und EINEFIRMA wurden zunächst dem Listen-Objekt (als Listen- und Werte-Bindung) zugewiesen, um danach den einzelnen Text-Objekten (nicht editierbar) die entsprechende Werte-Bindung zuzuweisen.

Bindungen stellen somit ein mächtiges Bindeglied zwischen den Anzeige-Objekten der Bedienoberfläche und den serverseitigen Daten zur Verfügung. In diesem Papier können nicht alle Bindungszuweisungen wie Sichtbar- und Aktiv-Bindung erläutert werden. Ebenso ist es nicht möglich, die Möglichkeiten der Initialisierung, der Validierung, der Hinterlegung von allgemeinen oder spezifischen, in Javascript geschriebenen individuellen Erweiterungen, wie sie auf dem Reiter Bindungen in der Multifunktionsleiste angeboten werden, zu erörtern – hier ist erheblicher Spielraum für applikationsspezifische Anpassungen gegeben.

Allerdings sollte anhand des einfachen Beispiels deutlich werden, in welcher Form Datenbank-Operationen in Seiten einer Anwendung angeboten werden: um Daten zwischen Client- und Operationen austauschen zu können, müssen entsprechende client-seitige Bindungen (einfache, komplexe) definiert und mit den Visualizern (als Aktiv-Bindung) verbunden werden. Die Richtung der Bindung wird durch Get, Set und

GetSet definiert. Diese Bindungen werden als Parameter an (Webservice) Operationen verbunden, damit die Daten aus der Datenbank gelesen (SELECT) oder in die Datenbank geschrieben werden können (INSERT, UPDATE und DELETE). Auch eine entsprechende Fehlerbehandlung von Datenbank-Fehlern in (Webservice) Operationen ist selbstverständlich vorgesehen, wird aber in diesem Papier nicht weiter behandelt.

Vorüberlegungen zweite Seite

Auf der zweiten Seite page2 wiederholt sich vieles, was schon auf der ersten Seite page1 der zweiten Beispielanwendung besprochen wurde: es muss eine Klasse clsAnsprechpartner mit den notwendigen Instanzvariablen angelegt werden. Danach muss diese Klasse als Bindung EINANSPRECHPARTNER und DIEANSPRECHPARTNER[*] instantiert werden. Die Bindung DIEANSPRECHPARTNER[*] wiederum muss als Rückgabewert der Operation AnsprechpartnerEinlesen() als Array definiert werden und – last but not least – muss als lokale Variable der Operation AnsprechpartnerEinlesen() ein Array dieser Klasse angelegt werden. Die Aktionen des Datenbankzugriffs werden analog zum ersten Beispiel codiert.

Design der zweiten Seite

Beim Design der zweiten Seite wird ebenfalls eine Liste angelegt (die mit den Bindungen DIEANSPRECHPARTNER[*] (Listen-Bindung) und EINANSPRECHPARTER (Werte-Bindung) verbunden wird. In die Liste wird ein dreizeiliges, einspaltiges Layout Grid gezogen. In die erste Zeile wird ein Text-Objekt gezogen und die Werte-Bindung DIEANSPRECHPARTNER.Name vorgenommen, während in die beiden weiteren Zeilen jeweils ein Link-Objekt (Link Type: Email und Phone) gezogen wird. Damit wird erreicht, dass in der produktiven Anwendung die Telefonnummer für Anrufe genutzt wird und die E-Mail-Adresse als Empfänger automatisch in eine neue E-Mail-Nachricht auf dem Smartphone eingetragen wird.

Codierung der zweiten Seite – Wechsel der Seiten

Um von der ersten Seite auf die zweite Seite zu gelangen, muss die zweite Seite von einem Eintrag aus der Liste aufgerufen werden, wobei mindestens die Firmennummer von der ersten auf die zweite Seite mit übergeben werden muss.

In diesem Beispiel wird beim Klick auf einen Listeneintrag der Firmen die clientseitige Aktion Navigate ausgeführt.

```
Navigate page2 EINEFIRMA
```

Source Code 4: Navigate auf Seite 2

Mit der Aktion Navigate wird eine andere Seite aufgerufen. Es wird daher aus der Liste nach dem Navigate-Eintrag die Seite page2 ausgewählt. Anstatt nur die Firmennummer zu übergeben, wird in diesem Beispiel gleich das ganze Objekt EINFIRMA als Parameter mit übergeben.

Auf der Seite page2 wird im Client-Teil als Parameter eine Instanz der Klasse clsFirma mit dem Namen P_EINEFIRMA definiert. Als Parameter der Operation AnsprechpartnerEinlesen() wird in diesem Beispiel eine Number definiert, die als Bindung den numerischen Wert P_EINEFIRA.Firmennr zugewiesen wird, die wiederum innerhalb der Operation unter dem Namen p_nFirmennr referenziert wird.

```
Operation: AnsprechpartnerEinlesen
  Description:
  Parameter:
    Number: p_nFirmennr
      Binding: P_EINEFIRMA.Firmennr
  Returns
```

```
    clsAnsprechpartner[*]
    Binding: DIEANSPRECHPARTNER
  Lokale Variablen:
    Sql Handle: hSql
    String: sStatement
    Number: nReturn
    Number: n
    clsAnsprechpartner: oAnsprechpartner[*]
  Aktionen
    If MitDatenbankVerbinden(hSql)
      Set sStatement = "SELECT CONT_FIRST_NAME, CONT_LAST_NAME, CONT_TITLE, CONT_PHONE, CONT_EMAIL
      FROM CONTACT
      WHERE COMPANY_ID = :p_nFirmennr
      INTO :oAnsprechpartner[n].Vorname, :oAnsprechpartner[n].Nachname, :oAnsprechpartner[n].Titel,
      :oAnsprechpartner[n].Telefon, :oAnsprechpartner[n].EMail  "
      If SqlPrepareAndExecute(hSql, sStatement)
        Set n = 0
        While SqlFetchNext(hSql, nReturn)
          Call oAnsprechpartner[n].NamenZusammensetzen()
            Set n = n + 1
        If SqlDisconnect(hSql)
          Return oAnsprechpartner
```

Source Code 5: die Operation AnsprechpartnerEinlesen (Seite 2)

In den Aktionen der Operationen ist lediglich neu, dass nach dem Fetchen eines Datensatzes eine Funktion NamenZusammensetzen() aufgerufen wird. Diese Funktion ist als Funktion der Klasse clsAnsprechpartner definiert.

Die Bewandnis dabei ist, dass in TD Mobile-Anwendungen viele Anwendungsfälle auftreten, bei denen Daten, so wie sie in der Datenbank verwaltet werden, nicht in der gleichen Form in der Oberfläche präsentiert werden können. Es müssen also „Umformungen“ vorgenommen werden. Der einfachste Fall liegt vor, wenn ein Datum in eine Date/Time-Variable übertragen wird. Die Darstellung einer solchen Information in einem Text-Objekt genügt oft nicht den Anforderungen der Anwender.

Als anderes Beispiel wird nach dem Fetchen eines Datensatzes aus der Datenbank die Funktion NamenZusammensetzen() ausgeführt, um Vor- und Nachname in einer Instanzvariablen Name zusammenzuführen. Der Code wird nachfolgend (aufs Wesentliche verkürzt) dargestellt.

```
Funktion: NamenZusammensetzen
  Description
  …
  Aktionen
    Set Name = Vorname || " " || Nachname
```

Source Code 6: die Funktion NamenZusammensetzen der Klasse clsAnsprechpartner

Zusammenfassung

Wir haben in einem zweiten Beispiel eine Anwendung die aus zwei Seiten besteht, erstellt, wobei auf beiden Seiten Listen zum Anzeigen von Firmen- bzw. Ansprechpartnerdaten verwendet wurden. Es wurden einige Oberflächen-Objekte und einige Container-Typen vorgestellt.

Das Thema Bindungen wurde anhand der komplexesten SQL-Anweisung, einer SELECT-Anweisung, behandelt: eine SELECT-Anweisung ist deshalb komplex, weil dabei mehrere Datensätze aus der Datenbank

an die Client-Anwendung übertragen werden müssen. In diesem Zusammenhang wurden (funktionale) Klassen und deren Instantierung an verschiedenen Stellen im Source Code behandelt.

Um die Bildung eines Cursors für die Arbeit mit der Datenbank zu vereinfachen, wurde eine interne Funktion MitDatenbankVerbinden() vorgestellt, in dem auch die neue SqlConnectDotNet()-Funktion erläutert wurde, die in jedem Fall bei der Erstellung einer mobilen Anwendung (anstelle SqlConnect) verwendet werden sollte.

Ausblick

Es wurden zwei kleine Beispielanwendungen Schritt für Schritt entwickelt, um Ihnen als Nutzer der Testversion von TD Mobile einen einfachen Zugang zur Entwicklungsumgebung zu ermöglichen.

Diese Papier kann selbstverständlich nicht eine tiefergehende Einführung in TD Mobile nicht ersetzen.

- Es wurden nicht alle Objekte – beispielsweise die Integration beliebiger HTML-Controls - und Container behandelt.
- Die Arbeit mit den „offenen“ Schnittstellen wie Integration von .Net-Bibliotheken, die Erstellung eigener Java- oder JQuery Mobile Scripte, das Einbindungen von eigenen Cascading Style Sheets, usw. wurden nicht behandelt und auch das Thema Debugging von Operationen sollte an dieser Stelle nur kurz erwähnt werden.
- Funktionalitäten wie Geopositionierung, Zwischenspeichern von Daten im Offline-Betrieb, usw. können hier nur erwähnt, aber nicht vollständig „geprototyped“ werden.
- Auch die Authentifizierung für den beschränkten Zugang zu einer Anwendung konnte im Rahmen dieser kurzen Einführung nicht behandelt werden.

Dennoch hoffen wir, Ihnen mit dieser kleinen Einführung die Evaluierung ihrer Testversion von TD Mobile leichter gemacht zu haben.