Oracle APEX

Das umfassende Handbuch

Kapitel 6
Grundlegende APEX-Konzepte

Nachdem wir eine einfache Anwendung in APEX erstellt und die grundlegende Bedienung kennengelernt haben, sehen wir uns nun weitergehende Konzepte und wiederverwendbare Komponenten im Detail an und beleuchten die Verwendungsmöglichkeiten.

In diesem Kapitel wird es nicht in erster Linie darum gehen, neue Regionstypen und deren Behandlung vorzustellen. Wichtiger erscheint mir, verschiedene Grundkonzepte näher zu erläutern und den Gedanken dahinter so zu erläutern, dass Sie diese Konzepte bei den verschiedenen Komponententypen sicher anwenden können. Kennen Sie die in diesem Kapitel besprochenen Konzepte, können sie weit über 90 % der Funktionalität von APEX verstehen und benutzen. Beginnen möchte ich mit der Frage, wie APEX an die Daten kommt, die in den Anwendungen benötigt werden.

6.1 Komponenten mit Daten versorgen

In früheren Versionen von APEX war das Verfahren, Komponenten mit Daten zu versorgen, grundsätzlich anders als in den aktuellen Versionen. Ich finde, dass das APEX-Entwicklerteam hier eine großartige Arbeit gemacht hat, denn die unterschiedlichen Arten von Datenquellen, mit denen APEX arbeiten kann, sind so gekapselt worden, dass Sie als Entwickler mit den Interna der Datengewinnung im Regelfall nicht belastet werden.

Die Idee besteht darin, dass Daten aus lokalen Tabellen oder aus entfernten Datenquellen über einen Webservice stammen könnten. Unabhängig davon, dass diese Quellen unterschiedlich angesprochen werden müssen, liefern sie doch letztlich Daten, die in Form einer (logischen) Tabelle dargestellt werden können.

Dies ist der eine Blickwinkel in Richtung »Quelle der Daten«. Blicken wir in die andere Richtung, also Richtung APEX-Anwendung und dort auf die verschiedenen Regions- und Seitenelementtypen, so stellen wir fest, dass die meisten dieser Komponenten Daten benötigen, um zu funktionieren, seien es Auswahllisten, Formulare oder Berichte. Und daher bietet es sich an, auch die Versorgung dieser Komponenten zu standardisieren. Daher haben alle diese Komponenten (mit derzeit noch ganz wenigen Ausnahmen) auch identische Attribute zur Verwaltung der Datenquellen. Dies sorgt

für eine einfache und konsistente Bedienung unterschiedlichster Komponenten. Selbst wenn erfahrene APEX-Entwickler eigene, neue Komponententypen bereitstellen, kann die durch APEX entwickelte Abstraktion verwendet werden: Die neuen Komponenten verhalten sich exakt wie die mitgelieferten Komponenten von APEX.

Um APEX mitzuteilen, welche Daten verwendet werden sollen, beschreiben wir als Erstes, wo sich die Daten grundsätzlich befinden. Im zweiten Schritt wird dann die konkrete Datenquelle ausgewählt. Die Herkunft der Daten wird in einem Attribut festgelegt, das in der deutschen Version den Originalbegriff *Location* ein wenig unglücklich mit *Position* übersetzt, *Herkunft* finde ich besser. Zur Wahl stehen hier, ein wenig abhängig von der konkreten Komponente, die folgenden Optionen:

- `Lokale Datenbank`
  Dies ist der Normalfall, den wir in Kapitel 5, »Eine einfache Datenbankanwendung erstellen«, als einzige Option verwendet hatten. Wählen wir diese Quellen, haben wir die Wahl, eine Tabelle oder View auszuwählen oder eine SQL-Anweisung direkt auf die APEX-Seite zu schreiben. Als weitere, aber fortgeschrittene Option könnten wir auch eine Funktion in PL/SQL programmieren, die eine SQL-Anweisung zurückliefert, um noch mehr Flexibilität zu erhalten.
- `REST Enabled SQL`
  Wenn der Eigentümer der Daten nicht dem Workspace als Schema zugeordnet wurde, kann APEX dennoch auf diese Daten zugreifen, wenn der Eigentümer erlaubt hat, dass sich APEX über eine REST-Schnittstelle mit dem Eigentümer verbindet. In diesem Fall können dann alle SQL-Abfragen und direkten Zugriffe auf Tabellen oder Views durchgeführt werden, als wäre der Eigentümer ein Workspace-Schema. Diese Option werden wir in diesem Teil nicht näher besprechen, da hiermit lediglich ein anderer Weg angeboten wird, an Datenbankdaten heranzukommen. Die Optionen entsprechen ansonsten denen der lokalen Datenbank.
- `REST-Quelle`
  Diese Option verwendet »regulär entwickelte« Webservices als Datenquelle. Damit meine ich, dass nicht der Zugriff auf beliebige Daten eines Schemas ermöglicht werden soll, sondern dedizierte Services für spezielle Anwendungsfälle eingerichtet werden. Dies könnte zum Beispiel ein Abruf von Daten eines einzelnen Mitarbeiters sein oder ein Webservice, der dem Aktualisieren von Adressinformationen dient.
- `Regionsquelle`
  Bei Regionen war es in früheren Versionen möglich, SQL-Abfragen zur Datengewinnung als Attribut der Region einzustellen. Insbesondere selbst entwickelte Regionstypen könnten immer noch über dieses Attribut verfügen. Ebenfalls gilt dies

zum Beispiel für Auswahllisten, die mit älteren Versionen von APEX erstellt wurden. Diese können nicht automatisiert migriert werden und unterstützen daher die Vorgängerlösung noch. Für Neuanlagen von APEX-Anwendungen spielt diese Option keine Rolle mehr.

6.1.1 Lokale Datenbank

Diese Herkunft ist am wenigsten komplex. Einem Workspace, hatten wir gesagt, ist ein Datenbankschema zugeordnet (im Ausnahmefall können das auch mehrere Schemata sein). Alle Objekte, auf die dieses Schema zugreifen kann, stehen daher auch APEX zur Verfügung. Als Datenquelle interessant sind die Tabellen und Views, die dort eingerichtet sind.

Grundtechniken

Grundsätzlich können alle Tabellen und Views, die im Workspace-Schema vorhanden sind, genutzt werden. Über die Oberfläche kann gewählt werden, ob eine Tabelle oder View direkt verwendet oder ob eine SQL-Abfrage erfasst werden soll. Als weitere Option kann eine PL/SQL-Funktion angegeben werden, die eine `select`-Abfrage erzeugt. Dies ist kein Thema für den No-Code-Bereich dieses Buches, verspricht aber eine hohe Flexibilität, denn damit können basierend auf der konkreten Situation auf der Anwendungsseite unterschiedliche `select`-Abfragen erzeugt werden und grundsätzlich andere Daten im gleichen Bericht gezeigt werden.

Bitte beachten Sie, dass Sie, wenn Sie den Namen einer Tabelle oder View schreiben, diesen immer in Großbuchstaben angeben müssen.

[!]

Merke

Dass Tabellen- und Viewnamen in Großbuchstaben geschrieben werden müssen, hat seinen Grund darin, dass eine Tabelle in Oracle grundsätzlich auch mit einem Namen ausgestattet werden kann, die die konkrete Schreibweise beachten. Dies erreicht die Datenbank, indem der Tabellen- oder Viewname mit doppelten Anführungszeichen umgeben wird. Dies ist zwar ein Anti-Pattern und kommt extrem selten vor, ist aber nicht ausgeschlossen.

Daher muss APEX diese Variante in Betracht ziehen und umgibt jeden Tabellennamen mit den hierfür erforderlichen doppelten Anführungszeichen. Geben Sie nun den Namen nicht in Großbuchstaben ein, wird eine Tabelle mit genau dieser Schreibweise gesucht, und die gibt es nicht. Großbuchstaben sind andererseits der Standard in Oracle-Datenbanken, sodass in diesem Fall die Tabelle auch mit doppelten Anführungsstrichen um den Namen gefunden wird.

Besonderheiten

Das Workspace-Schema muss die Tabellen mit den Daten gar nicht unbedingt selbst besitzen, es reicht aus, dass das Workspace-Schema Zugriff auf diese Daten erhält. Damit ist gemeint, dass der Eigentümer der Tabellen dem Workspace-Schema Berechtigungen auf seine Tabellen einräumen kann, zum Beispiel das Recht, die Daten dieser Tabelle zu lesen.

Der Grund für dieses Vorgehen ist, dass dem Eigentümer einer Tabelle keine Rechte an den eigenen Tabellen entzogen werden können. Daher darf er die Tabelle zum Beispiel auch jederzeit löschen. Im Workspace-Schema darf APEX alle Aktionen auf Tabellen ausführen, inklusive des Löschens der Tabelle. Das ist auch gut so, denn nur so lassen sich die Beispielanwendungen überhaupt installieren. Diese Anwendungen legen neben der eigentlichen APEX-Anwendung auch Tabellen und andere Datenbankobjekte an. Das dürften sie nicht, wenn APEX im Workspace-Schema nicht als Eigentümer auftreten könnte.

In vielen Produktivumgebungen sind diese weitreichenden Rechte auf die Tabellen mit den Daten aber nicht durchsetzbar. Ein Grund hierfür können generelle Sicherheitsbedenken sein, ein anderer, dass die Daten, auf die die APEX-Anwendung zugreifen soll, schon für andere Anwendungen angelegt wurden und durch APEX nur mitgenutzt werden sollen.

Daher werden die Daten in Tabellen abgelegt, die einem anderen Datenbankbenutzer als dem Workspace-Schema gehören. Dieser andere Datenbankbenutzer erteilt nun zum Beispiel Leseberechtigungen an einer seiner Tabellen an das Workspace-Schema, wodurch die Daten in APEX verfügbar sind, aber eben nur lesend. So können Berechtigungen an Daten viel feiner verwaltet werden.

Eine Besonderheit ist in diesem Fall aber zu berücksichtigen: APEX kann zwar Daten dieser Tabellen lesen, nicht aber einen Assistenten auf diesen Tabellen ausführen lassen, denn APEX kann die Metadaten dieser Tabellen nur mit einem Trick lesen. Da der Assistent im Regelfall keinen Zugriff auf die Metadaten der Tabelle hat, die dem Workspace-Schema nicht gehören (welche Spalten enthält die Tabelle, welche Datentypen besitzen die Spalten und so weiter), ist ein Assistent nutzlos. Der Trick, dem Assistenten dennoch Zugriff auf diese Daten zu geben, ist simpel und entspricht zudem einer bereits bekannten Best Practice: Es wird eine View erstellt, die dem Workspace-Schema gehört. Da APEX die Metadaten der View lesen kann und diese aus der referenzierten Tabelle abgeleitet werden, funktioniert nun alles wieder problemlos.

Wenn der Eigentümer der Tabelle nicht nur Leserechte, sondern auch Schreibrechte vergibt (er kann frei wählen, ob `insert`, `update` und/oder `delete`-Rechte eingeräumt werden), kann die APEX-Anwendung Änderungen »durch die View« in die Tabelle schreiben, wenn es sich um eine »einfache« View handelt. Mit diesem Begriff bezeichnet Oracle eine View, die nicht die Daten mehrerer Tabellen darstellt (also Joins ent-

hält) oder durch eine Gruppenfunktion Aggregationen vornimmt. Das funktioniert auch dann, wenn die View für die Spalten abweichende Bezeichner (Spaltenaliase) deklariert hat.

Wie aber kann man vorgehen, wenn keine Schreibrechte auf einer Tabelle existieren, man aber dennoch Daten in diese schreiben möchte? Das mag zunächst widersinnig klingen, doch ist dies häufig der Fall, einfach weil man nicht möchte, dass ungeprüft Daten in die Tabellen geschrieben werden können, sondern nur nachdem die Daten durch den Eigentümer auf Plausibilität geprüft wurden.

In diesem Fall wird eine Programmierung in PL/SQL erforderlich, der Eigentümer der Tabellen bietet Methoden zum Schreiben an und räumt dem Workspace-Schema das Recht ein, diese Methoden ausführen zu dürfen. Idealerweise sind dies nicht einfach Methoden zum Schreiben, Aktualisieren und Löschen von Daten, sondern fachlich geschnittene Methoden für die jeweiligen Anwendungsfälle. Eine Methode hieße also nicht `insert_employee`, sondern vielleicht `hire_employee`. Wie man solche Methoden schreibt, ist selbstverständlich außerhalb des Fokus dieses Buches und insbesondere dieses Teils, der sich ja um die No-Code-Fähigkeiten von APEX bemüht, aber eben auch nicht vollständig uninteressant, weil diese Methoden durch Kollegen bereits geschrieben worden sein können. Und in APEX ist es tatsächlich möglich, diese Methoden ohne Programmierung rein deklarativ anzusprechen und zu verwenden. Wie das geht, sehen wir uns aber erst im Teil »Low Code« an.

6.1.2 REST-Quelle

Diese Option ist der beschriebenen Option, auf lokale Datenbanken über eine Speichermethode zuzugreifen, sehr ähnlich. Der Weg, Daten über einen Webservice zugänglich zu machen, ergibt nur dann einen Sinn, wenn die Daten nicht über Tabellen oder Views des Workspace-Schemas zugänglich sind, ansonsten wäre der Zugriff über SQL der direktere und schnellere Weg. Wir haben also wieder die Situation, dass ein anderer Datenbankbenutzer oder ein anderer Server Daten bereitstellt, für die uns ein Zugriff erlaubt werden muss. Ein Webservice ist seinem Wesen nach auch eine Methode, an die Parameter übergeben werden oder die Werte zurückliefert. Daher entspricht dieser Ansatz den PL/SQL-Methoden zum Schreiben auf Daten, nur dass die Methoden nicht direkt in der Datenbank, sondern über einen URL aufgerufen werden. Es ist sehr wohl möglich, diese Methoden so zu implementieren, dass nicht plausible Daten durch den Webservice abgelehnt werden, auch insofern ist der Weg vergleichbar.

Webservices werden sowohl für den lesenden als auch für den schreibenden Zugriff bereitgestellt. In APEX können solche REST-basierten Datenquellen deklarativ erstellt werden und zum Beispiel Tabellendaten anbieten oder Datensätze erstellen. Der Vorteil eines Webservices wird von vielen Teams darin gesehen, dass zwischen der Daten-

quelle und der Anwendung eine standardisierte Schicht besteht, die die Koppelung zwischen den beiden Ebenen reduziert. Ob dies im Umfeld eines Oracle-basierten Webservices in der gleichen Datenbank wie die APEX-Anwendung wirklich ein Vorteil ist, würde ich gern mit den Verfechtern diskutieren. In jedem Fall ist aber die Möglichkeit, auch entfernte Datenquellen in eine APEX-Anwendung zu integrieren, ein immenser Vorteil. Ich erinnere mich zum Beispiel an eine APEX-Anwendung, in der lokale Daten mit Daten aus einem Ticketsystem und einem Versionierungssystem kombiniert wurden. Das geht auf denkbar einfache Weise.

Ein Webservice, der durch die deklarativen Möglichkeiten von APEX erstellt wurde, bietet den Vorteil, dass APEX-intern eine standardisierte Abbildung der Daten auf das JSON-Format stattfindet, sodass die Rückumwandlung der JSON-Daten in Daten für Tabellen ohne Programmierung möglich ist. Dabei enthalten die JSON-Daten nicht nur die eigentlichen Daten, sondern auch Metainformationen, zum Beispiel für die Paginierung umfangreicher Datenmengen. Es ist im Moment jedoch noch außerhalb des Fokus dieses Teils, zu erklären, wie genau eine Tabelle als Webservice bereitgestellt werden kann. Ich möchte allerdings zeigen, wie ein bestehender Webservice referenziert wird, um ihn anschließend als Datenquelle für einen Bericht zu verwenden.

In einem anderen Workspace, `BUCH_REST`, werden Daten weiterer Mitarbeiter als Webservice zur Verfügung gestellt. Da die Authentifizierung dieser Daten nicht durch die Datenbank, sondern außerhalb erfolgt und ich keinen Zugriff auf diese Ressourcen habe, nehmen wir für das Beispiel hin, dass diese Daten öffentlich zugänglich sind und keine Authentifizierung benötigen. Der Webservice wird auf dem gleichen Server veröffentlicht, auf dem auch APEX ausgeführt wird, daher können Sie sich in Ihrem Workspace eine Referenz auf diesen Webservice selbst erstellen. Gehen Sie hierfür in die <Gemeinsamen Komponenten> • Datenquellen • REST-Datenquellen, und legen Sie dort eine neue Datenquelle an. Der Assistent erinnert an den Assistenten zur Anlage einer Auswahlliste. Wählen Sie auf der ersten Seite, dass Sie eine neue Datenquelle einfügen wollen, und klicken Sie auf Weiter >. Sie kommen zum Dialog in Abbildung 6.1.

Abbildung 6.1 Referenzierung einer REST-Datenquelle

Als REST-Datenquellentyp nutzen Sie den Eintrag `Oracle REST Data Services` und profitieren so von der innerhalb von APEX standardisierten Struktur der JSON-Antwort. Der erste Teil des URL-Endpunkts ist der gleiche wie in der aktuellen APEX-Adresse, bis einschließlich `/ords/`. Dann folgt `rest/hr/employees`. Achten Sie auf korrekte Schreibweise, Groß- und Kleinschreibung werden beachtet. Wenn Sie auf Weiter > klicken, wird geprüft, ob der Webservice erreicht werden kann. Ist dies erfolgreich, wird der Server für diesen Webservice vom eigentlichen Service getrennt (der URL wird entsprechend geteilt, sodass der URL bis einschließlich `/rest/` als Server und der Rest als Service identifiziert wird). Das ist so korrekt und wird dazu führen, dass wir anschließend weitere Services dieses Servers einfacher konfigurieren können. Klicken Sie daher im folgenden Dialog auf Weiter >. Im folgenden Dialog »Einstellungen« könnten wir uns um die Kontrolle der Seitennummerierung kümmern, wir nehmen aber die Standardeinstellungen, klicken auf Weiter > und ändern auch nichts am letzten Dialogfenster, in dem wir die Authentifizierung kontrollieren könnten. Stattdessen klicken wir auf Ermitteln >.

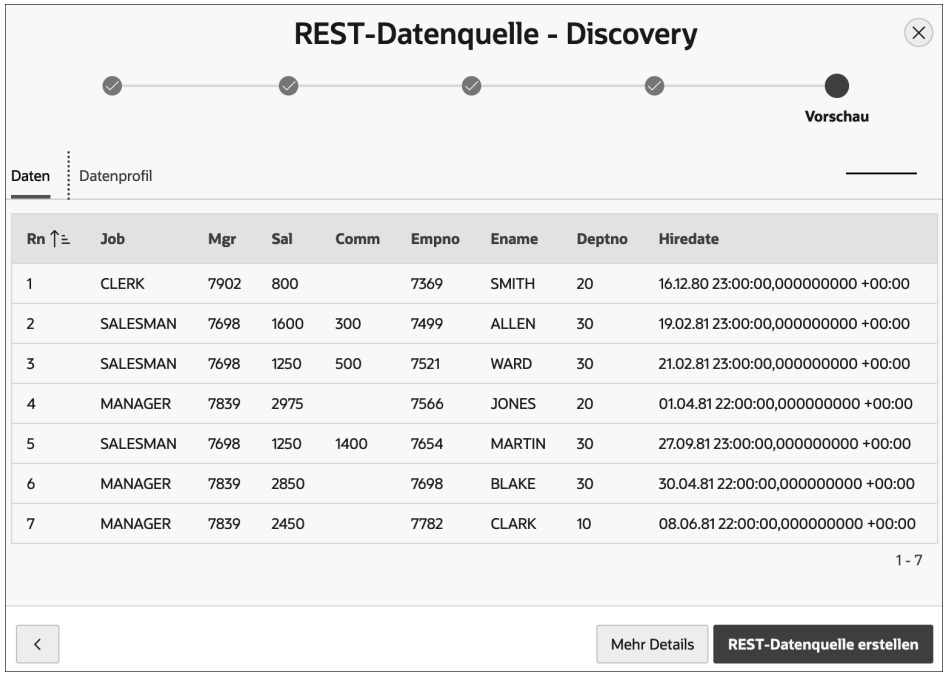


Abbildung 6.2 Automatisch erkannte Spalten der Datenquelle

Nun kontaktiert APEX den Webservice und analysiert automatisch die Antwort. Sie können das Ergebnis der Analyse unmittelbar sehen und kontrollieren. Zudem können Sie im Tabulator Datenprofil die erkannten Spaltentypen einsehen. Es sollte alles korrekt sein, schließlich haben wir einen rein APEX-internen Webservice verwendet. Klicken Sie zur Bestätigung auf REST-Datenquelle erstellen.

Die Verwendung dieser REST-Datenquelle ist dann unspektakulär. Sie können zum Beispiel auf <Home> einen Bericht hinzufügen, dessen Attribute der Quelle so eingerichtet werden wie in Abbildung 6.3 gezeigt.

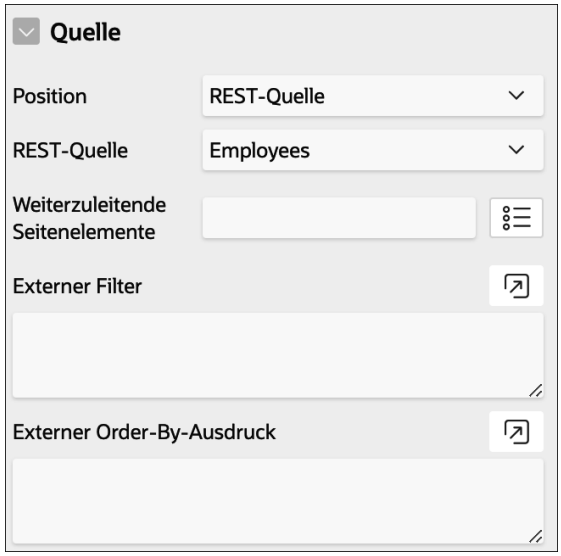


Abbildung 6.3 Referenzierung der REST-Datenquelle im Bericht

Schon die Ermittlung der Spaltenwerte hatte gezeigt, dass der REST-Webservice als Quelle für einen Bericht interpretiert werden kann, denn auch der Assistent hat ja einen Bericht mit den Spaltenwerten angezeigt. Dass dies nun auch in eigenen Anwendungen möglich ist, überrascht nicht weiter. Dennoch ist es sehr schön, dass diese Möglichkeit besteht, zumal sie nicht auf lokale Webservices beschränkt ist, sondern beliebige Webservices als Datenquellen interpretieren kann.

Das Problem dabei ist, dass keine allgemeingültigen Standards bezüglich des Formats der Antwort existiert. Es ist nicht einmal sichergestellt, dass ein Webservice JSON liefert, sondern es können beliebige Formate geliefert werden, von XML über CSV bis hin zu XLS oder PDF. Daher müssen Webservices, die APEX nicht bekannt sind, aufwendiger auf Spalten gemappt werden als APEX-interne Webservices. Hier helfen dann das Datenprofil und die in ihm möglichen Optionen.

Sehen wir uns ein nettes Beispiel hierfür an, das ich in einem Blog des Oracle-Mitarbeiters Roberto Capancioni gefunden habe (*https://tinyurl.com/yc2j2ufd*). Es nutzt den kostenlosen REST-Service *openfootball*. Sehen wir uns in einem ersten Schritt an, wie wir die Daten als REST-Datenquelle verfügbar machen können. Die Visualisierung nehmen wir dann in weiteren Abschnitten des Buches auf. Damit wir diese Funktionalität von unserer Mitarbeiterverwaltung trennen, bitte ich Sie, eine neue Anwendung `Fußball` mit dem Alias `FIFA` anzulegen. Wir benötigen hierfür keine besonderen Features, sondern kümmern uns gleich um die Anbindung an die REST-Datenquelle.

Allerdings müssen wir der Datenbank erlauben, den URL dieses Service ansprechen zu dürfen. Hierfür legen wir in der Datenbank einen *ACL*(*Access Control List*)-Eintrag an. Der folgende Code muss als Administrator der Datenbank ausgeführt werden (dieses Skript wurde bei der Installation der Datenbankobjekte bereits ausgeführt):

```
begin
  dbms_network_acl_admin.append_host_ace(
    host => 'raw.githubusercontent.com',
    lower_port => 80,
    ace => xs$ace_type(
              privilege_list => xs$name_list('http'),
              principal_name => 'BUCH_NO_CODE',
              principal_type => xs_acl.ptype_db));
end;
/
```

Listing 6.1 Befehl zum Anlegen eines ACL-Zugriffs

Ein Zugriff auf eine `https`-Ressource benötigt zudem ein Zertifikat, das in der Oracle-Datenbank in einem *Wallet* genannten Tresor verwahrt wird. Wie dieses Wallet einzurichten ist, ist außerhalb des Fokus dieses Buches und setzt in den meisten Fällen wohl auch die Mitarbeit des Datenbank-Administrators voraus. Da zudem ein sehr guter deutschsprachiger Blog zu diesem Thema existiert (siehe *https://tinyurl.com/nmkh5uvh*), verweise ich darauf und setze im Folgenden voraus, dass `https` genutzt werden kann.

Als URL für den REST-Service nutzen wir *https://raw.githubusercontent.com/openfootball/football.json/master/2020-21/de.1.json* (ich bitte Roberto um Nachsicht, dass ich hier die 1. Bundesliga verwende und nicht die Serie A). Geben Sie diesen URL im Browser ein, erhalten Sie ein JSON-Dokument mit folgendem Aufbau zurück:

```
{
  "name": "Bundesliga 2020/21",
  "matches": [
    {
      "round": "Spieltag 1",
      "date": "2020-09-18",
      "team1": "Bayern München",
      "team2": "FC Schalke 04",
      "score": {
        "ft": [
          8,
          0
        ]
```

```
      }
    },
    {
      "round": "Spieltag 1",
      "date": "2020-09-19",
      "team1": "Eintracht Frankfurt",
      "team2": "Arminia Bielefeld",
      "score": {
        "ft": [
          1,
          1
        ]
      }
    },
...}
```

Listing 6.2 JSON-Antwort des openfootball-Webservices

Aufbauend auf diesem URL können wir nun eine REST-Datenquelle erstellen. Gehen Sie hierfür zu <GEMEINSAME KOMPONENTEN> • DATENQUELLEN • REST DATENQUELLEN, und legen Sie eine neue Datenquelle an.

Abbildung 6.4 Anlegen der Openfootball-REST-Schnittstelle

Wir brauchen keine Authentifizierung und können daher alle anderen Assistentenseiten mit ihren Standardeinstellungen akzeptieren, bis wir zu dem Punkt kommen, den Service durch Klick auf die Schaltfläche ERMITTELN vorbereiten zu lassen.

In meiner Version von APEX hat sich dieser Prozess standhaft geweigert, das Ergebnis als JSON zu identifizieren, und schlug stattdessen CSV vor. Sollte das bei Ihnen auch passieren, akzeptieren Sie dies für den Moment, und klicken Sie auf die Schaltfläche ERSTELLEN, wir korrigieren das dann anschließend.

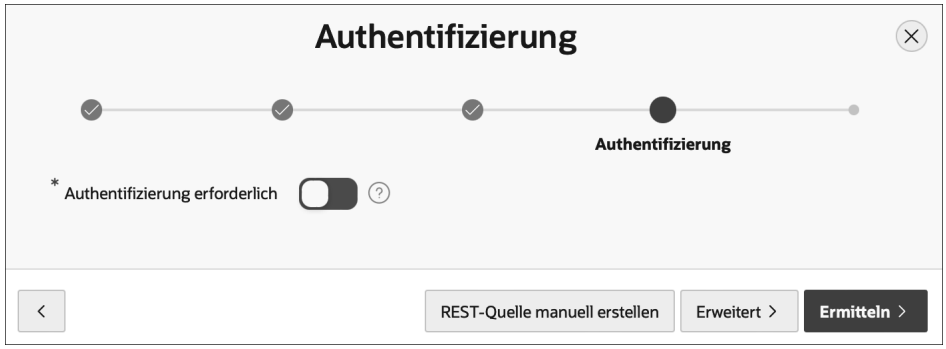


Abbildung 6.5 Automatische Analyse des REST-Services starten

Merke

Zur nachträglichen Korrektur speichern wir zunächst das JSON-Ergebnis des URL im Browser in einer lokalen Datei. Dann klicken wir zur Korrektur auf die erstellte REST-Datenquelle und bearbeiten anschließend das Datenprofil durch Klick auf die Schaltfläche DATENPROFIL BEARBEITEN.

Stellen Sie nun im Attribut DATENPROFIL • FORMATIEREN den Eintrag `JSON` ein, wie in Abbildung 6.6 gezeigt. Es erscheint das Attribut DATENPROFIL • ZEILENSELEKTOR. Ein Blick in die JSON-Datei zeigt, dass das JSON-Attribut `matches` ein Array von Spielen enthält. Daher tragen wir diesen Namen in exakt dieser Schreibweise in das Attribut ein.

Im Bereich ERNEUTE RECOVERY ziehen wir nun die gespeicherte JSON-Datei in den Dateiupload-Bereich und klicken auf die Schaltfläche DATENPROFIL ERNEUT ERKENNEN. Nun sollte es besser aussehen, und die Spalten `DATE_`, `ROUND`, `TEAM1` und `TEAM2` sollten erscheinen. Klicken Sie abschließend auf die Schaltfläche DATENPROFIL ERSETZEN, um die alten Spalten durch die neuen zu ersetzen.

Abbildung 6.6 Korrektureinstellungen des Datenprofils

Nachdem nun die Struktur des Webservices aus der JSON-Antwort ermittelt wurde, sehen wir einige unschöne Dinge, die wir anschließend ergänzen und verbessern:

- Die Daten benötigen einen Primärschlüssel.
- Die erste Spalte hat den Namen `DATE_`, weil das JSON-Attribut einen geschützten SQL-Bezeichner als Namen trägt.
- Die Spalten `TEAM1` und `TEAM2` könnten Unterstriche zwischen dem Namen und den Ordinalzahlen vertragen.
- Die Tore fehlen!

Zur Korrektur bearbeiten wir wieder das Datenprofil und editieren die vorhandenen Spalten durch Klicks auf die Bleistiftsymbole. Dabei ist es einfach, die Spaltenbezeichner anzupassen. Bitte verwenden Sie die Spaltennamen `PLAY_DATE`, `ROUND`, `TEAM_1` und `TEAM_2`, um Probleme im weiteren Verlauf zu vermeiden. Für die Spalten `PLAY_DATE` (`DATE_`) und `TEAM_1` (`TEAM1`) legen Sie bitte noch fest, dass sie Primärschlüsselspalten sind, indem Sie das Attribut Spalte • Primärschlüssel anschalten.

Um die Tore hinzuzufügen, müssen wir ein wenig mehr tun.

Die Tore befinden sich im JSON in einem strukturierten Attribut namens `score`, wie in Listing 6.3 gezeigt.

```
"score": {
  "ft": [1, 1]
}
```

Listing 6.3 Ausriss aus der JSON-Datei

Um diese Informationen als Spalten abzubilden, erstellen wir im Datenprofil eine neue Spalte durch Klick auf die Schaltfläche Spalte hinzufügen.

Der entscheidende Punkt ist der Eintrag in das Attribut Quelle • Selektor. Der Ausdruck `score.ft[0]` zeigt auf das erste Arrayelement im Kindelement `ft` von `score` (JavaScript zählt immer von 0 beginnend). Den Datentyp definieren Sie als `NUMBER`. Beachten Sie bitte auch, dass Sie die Spalte im Attribut Spalte • Sichtbar anschalten müssen, ansonsten wären die Tore im Bericht nicht zu sehen. Analog legen Sie nun noch eine zweite Spalte mit dem Selektor `score.ft[1]` an und nennen diese `GOALS_2`.

Damit Sie die Früchte Ihrer Arbeit ernten können, erstellen Sie nun noch schnell auf <HOME> eine neue Region vom Typ `interaktiver Bericht` und legen die Quelle auf die REST-Datenquelle fest. Das Ergebnis sehen wir in Abbildung 6.8.

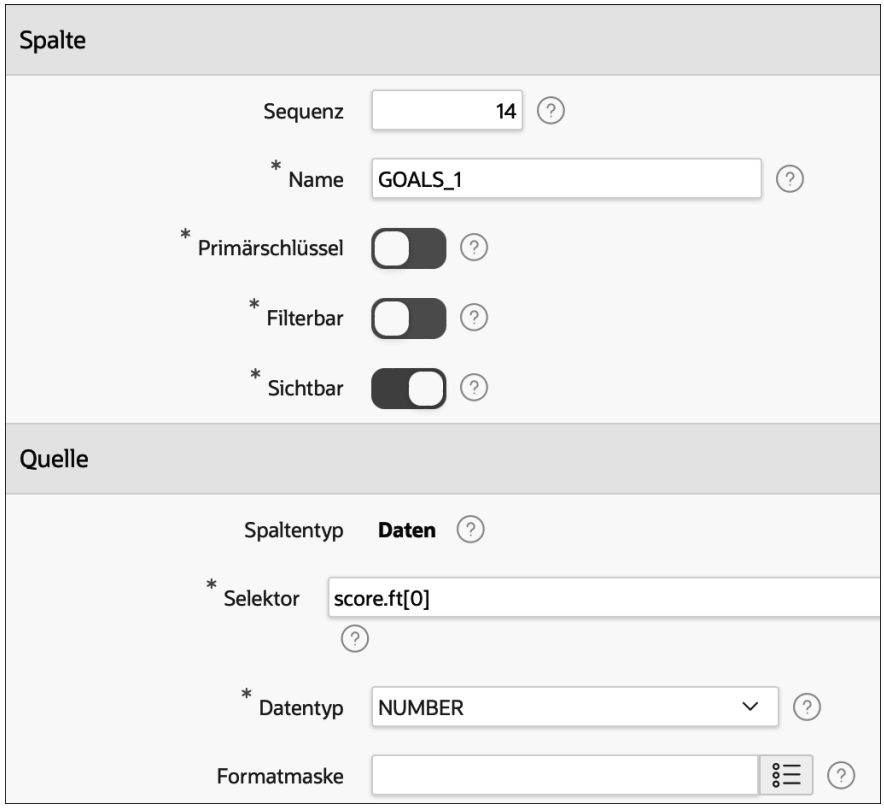
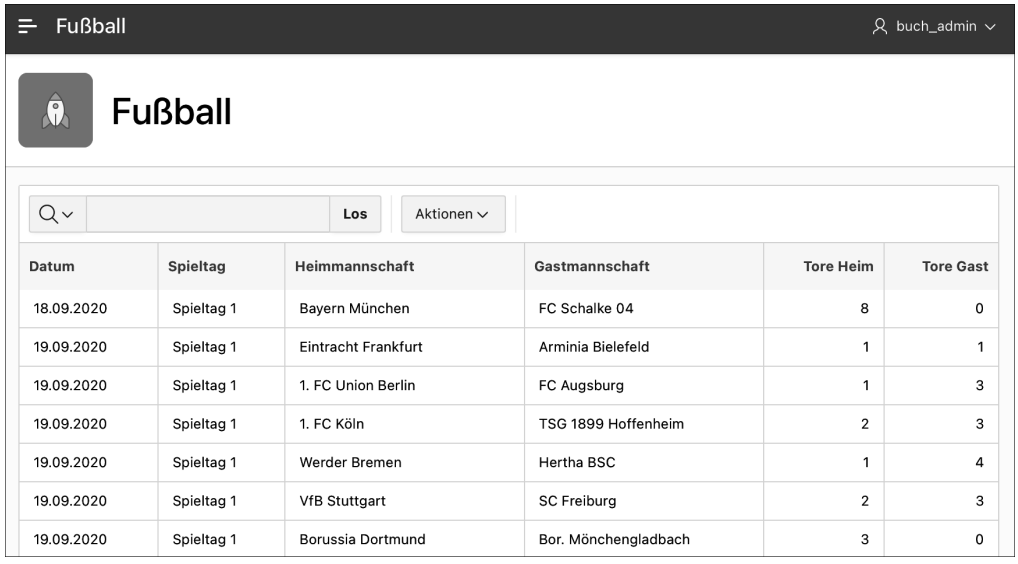


Abbildung 6.7 Dialog zum Hinzufügen einer Spalte



Datum	Spieltag	Heimmannschaft	Gastmannschaft	Tore Heim	Tore Gast
18.09.2020	Spieltag 1	Bayern München	FC Schalke 04	8	0
19.09.2020	Spieltag 1	Eintracht Frankfurt	Arminia Bielefeld	1	1
19.09.2020	Spieltag 1	1. FC Union Berlin	FC Augsburg	1	3
19.09.2020	Spieltag 1	1. FC Köln	TSG 1899 Hoffenheim	2	3
19.09.2020	Spieltag 1	Werder Bremen	Hertha BSC	1	4
19.09.2020	Spieltag 1	VfB Stuttgart	SC Freiburg	2	3
19.09.2020	Spieltag 1	Borussia Dortmund	Bor. Mönchengladbach	3	0

Abbildung 6.8 Bericht über die REST-Datenquelle

Ich weiß, dass man hier noch vieles schöner machen könnte. Darauf komme ich auch noch zurück, allerdings möchte ich zunächst eine technische Verbesserung mit Ihnen besprechen.

Derzeit lesen wir die Ergebnisse direkt vom Webservice. Die Informationen, das können wir dem URL entnehmen, beziehen sich auf die Saison 2020–21 und sollten sich daher nach meiner Kenntnis des Regelwerks wohl nicht mehr ändern. Daher wäre es doch effizienter, diese Daten in einer lokalen Datenbank zu speichern und nicht jedes Mal online nachzuschlagen.

Auch das können wir deklarativ erreichen. Wir gehen dafür zu <GEMEINSAME KOMPONENTEN> • DATENQUELLEN • REST-DATENQUELLEN. Im Bericht sehen wir neben dem Namen unserer Datenquelle den Eintrag `Synchronisiert` • `Nein`. Klicken Sie auf den Link, um einen Assistenten zum Synchronisieren dieser Datenquelle zu öffnen (Abbildung 6.9).

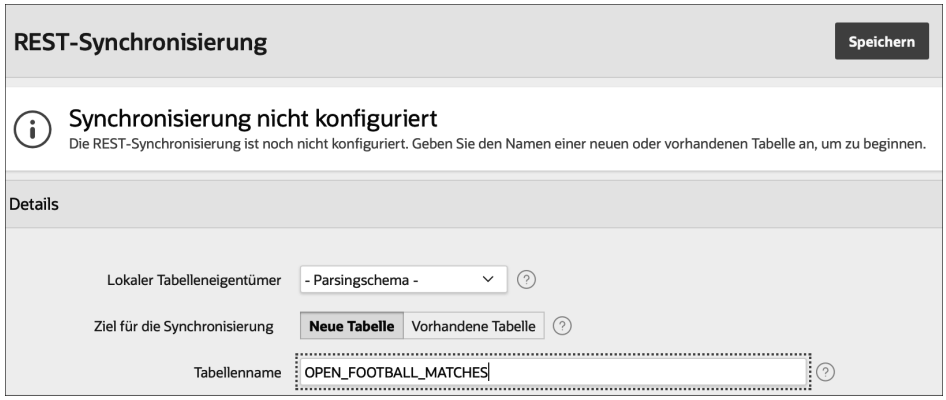


Abbildung 6.9 Synchronisierung einer REST-Datenquelle

Tragen Sie bitte den Namen der Tabelle genauso ein wie in Abbildung 6.9, weil ich für diesen Namen Views für die spätere Visualisierung vorbereitet habe. Achten Sie insbesondere auf die Großschreibung. Wenn Sie die Einstellungen speichern, werden Sie darauf hingewiesen, dass diese Tabelle noch nicht existiert. Bevor Sie sich die Tabelle erstellen lassen, können Sie die Schaltfläche SQL anzeigen klicken, um sich die SQL-Anweisung zum Erstellen der Tabelle anzeigen zu lassen. Beachten Sie, dass die Tabelle einen Primärschlüsselconstraint erhalten wird, da wir ja im Datenprofil die entsprechenden Einstellungen vorgenommen hatten (Abbildung 6.10). Klicken Sie anschließend auf die Schaltfläche TABELLE ERSTELLEN.

Abbildung 6.10 Erstellung einer Synchronisierungstabelle

Als Nächstes müssen wir noch festlegen, in welchen Intervallen die lokale Tabelle mit dem Webservice synchronisiert werden soll. Nun zahlt sich aus, dass wir einen Primärschlüssel angelegt hatten, denn dadurch haben wir die Option, die Daten des Webservices mit der lokalen Tabelle zusammenführen zu lassen. Klicken Sie für dieses Verfahren auf SYNCHRONISIERUNGSTYP • `Zusammenführen` (❶ in Abbildung 6.11) und anschließend auf die Symbolschaltfläche zur Erstellung eines Zeitintervalls (❷ in Abbildung 6.11), um einzustellen, in welchem Intervall die Synchronisierung erfolgen soll.

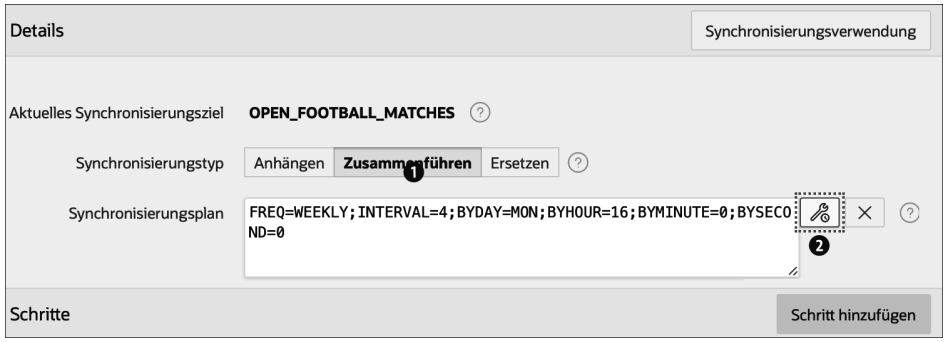


Abbildung 6.11 Einstellungen zur Synchronisierung mit dem REST-Webservice

Die Synchronisierung erfolgt direkt durch die Datenbank über ein dort integriertes System zur Ausführung zeitgesteuerter Jobs. Daher ist es nicht erforderlich, dass die APEX-Anwendung läuft, und es gilt immer die Zeitzone des Datenbankservers. Mithilfe des Dialogs aus Abbildung 6.12 können Sie nun das Aktualisierungsintervall frei definieren.

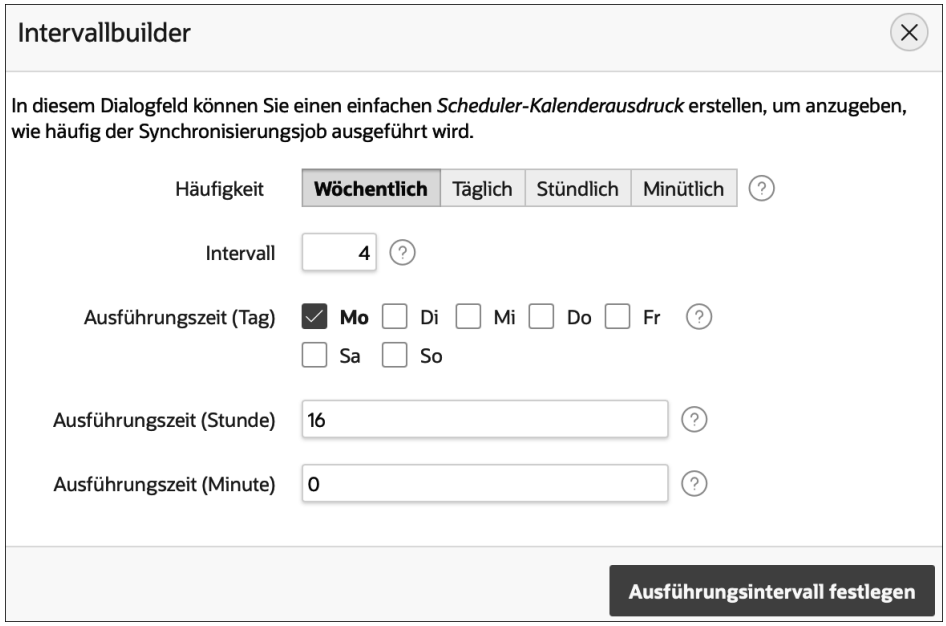


Abbildung 6.12 Dialog zur Definition von Zeitintervallen

Im Beispiel vereinbaren wir eine automatisierte Aktualisierung im Vier-Wochen-Rhythmus, jeweils um 16:00 Uhr. Bestätigen Sie Ihre Einstellungen mit der Schaltfläche SPEICHERN UND AUSFÜHREN. Sofort wird die Datenquelle synchronisiert, und die Daten werden in die Tabelle geladen. Im Dialog können Sie zudem sehen, wann das nächste Mal der REST-Service aktualisiert werden wird.

Abbildung 6.13 Synchronisierung des REST-Webservices

Eine letzte Änderung führen wir nun noch durch. Wenn wir zum Bericht auf die Seite <HOME> gehen und uns dort die <BERICHTSATTRIBUTE> • QUELLE ansehen, hat sich ein neues Attribut eingeschlichen. Sie können nun durch einfachen Klick auf den Schalter SYNCHRONISIERUNGSTABELLE VERWENDEN entscheiden, ob der Bericht die

lokal zwischengespeicherten Daten oder die aktuellen Daten des Webservices verwenden soll.

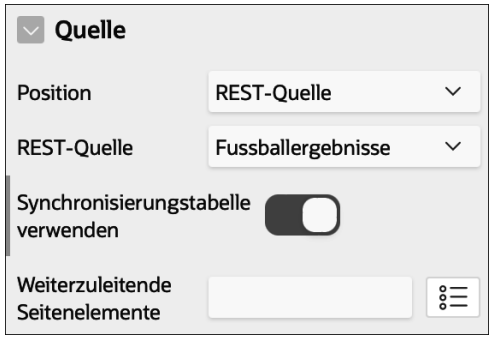


Abbildung 6.14 Verwendung der Synchronisierungstabelle

Das spart den Roundtrip zum Webservice und erhöht so die Performanz, kann aber jederzeit wieder rückgängig gemacht werden, wenn Sie aktuelle Daten benötigen.

6.1.3 Identifikation einer Zeile

Für viele Verwendungen ist es erforderlich, dass ein Schlüssel zur Identifikation einer Zeile einer Datenquelle bereitsteht. Dies gilt insbesondere für Formulare, die auf Basis dieser Informationen entscheiden, ob eine Bearbeitung bestehender Daten vorliegt (die Primärschlüsselinformation ist gefüllt) oder eine Neuanlage (die Primärschlüsselinformation ist leer).

Daher muss eine Möglichkeit existieren, APEX mitzuteilen, auf welche Art ein Datensatz eindeutig identifiziert werden kann. Dies ist je nach Datenquelle etwas unterschiedlich, doch existieren für die lokale Datenbank zwei Strategien, die ich hier vorstellen möchte.

Primärschlüsselspalte

Bei der ersten Variante wird die Primärschlüsselinformation der Tabelle verwendet. APEX unterstützt deklarativ die Verwendung von bis zu zwei Spalten für diese Information. Dieser Weg bietet sich insbesondere dann an, wenn die Primärschlüsselwerte durch eine Autowertspalte oder eine Sequenz automatisch in der Datenbank erzeugt werden. Der Grund ist, dass das Formular das Editieren dieser Spaltenwerte unterbinden wird, weil APEX davon ausgeht, dass beim Speichern ein neuer Wert gebildet wird. Zudem soll ein bestehender Wert nicht änderbar sein.

Es gibt aber Situationen, in denen dieses Verhalten nicht anwendbar ist, wie zum Beispiel bei Lookup-Tabellen, die häufig einen alphanumerischen und frei wählbaren Primärschlüsselwert besitzen. Nehmen wir als Beispiel eine Tabelle zum Nachschla-

gen eines Ländercodes, deren Primärschlüsselwert zum Beispiel aus einem dreistelligen Wert gemäß *ISO 3166 Alpha-3* abgeleitet wird. Hier kann die Datenbank keinen neuen Primärschlüsselwert generieren.

In diesen Fällen kann man unterschiedlich reagieren, wie wir uns in Kapitel 10, »Formulare«, ansehen werden. Eine mögliche Strategie besteht in der Verwendung der `ROWID`.

ROWID

Die `ROWID` ist eine sogenannte Pseudospalte, also eine Spalte, die in jeder `select`-Anweisung abgefragt werden kann, aber nicht Teil der Tabelle ist. Es handelt sich hierbei um die physikalische Speicheradresse der Zeile innerhalb der Oracle-Datenbank und ist grob mit dem Speicherort einer Datei im Dateisystem vergleichbar. Insbesondere teilt sie mit diesem Vergleich die Eigenschaft, die Information auf schnellstmögliche Weise zugreifbar zu machen, denn es ist egal, wie viele Dateien gespeichert sind. Wenn wir den konkreten Speicherort kennen, ist der Zugriff darauf maximal schnell.

Die `ROWID` ist kein vollwertiger Ersatz für einen Primärschlüssel, weil der Wert sich über die Zeit ändern kann, ähnlich wie sich auch der Speicherort einer Datei ändern kann. Diese Änderungen sind allerdings sehr selten und treten nur auf, wenn administrative Arbeiten an den Datenbanktabellen durchgeführt werden. Es wird daher im Regelfall toleriert, dass diese Informationen volatil sind, weil nicht davon ausgegangen wird, dass in der kurzen Zeit, die für die Bearbeitung einer Zeile aufgewendet werden muss, ein solches Ereignis eintreten wird.

Die `ROWID` kann also in diesen Grenzen als Ersatz für eine Primärschlüsselinformation verwendet werden. Auf dem Formular wird nun die »eigentliche« Primärschlüsselinformation nicht mehr geschützt und kann frei editiert werden. Soll ein Wert nicht mehr verändert werden können, nachdem der Datensatz angelegt wurde, kann dies dadurch erreicht werden, dass das Eingabefeld mit dieser Information schreibgeschützt wird, falls es nicht leer ist. Dies erreichen Sie durch das Attribut Schreibgeschützt • Typ • `Element ist NOT NULL` und der Angabe des Elementnamens im Attribut Schreibgeschützt • Element.

Neu angelegte Zeilen können dann einfach auf einen eindeutigen Wert gesetzt, nachträglich aber nicht mehr geändert werden.

6.2 Das Layout kontrollieren

Eine APEX-Anwendungsseite, das hatten wir bereits bei der Erstellung unserer eigenen Anwendung bemerkt, platziert die Regionen auf dem Bildschirm auf Basis eines Rasters aus 12 Zeilen und beliebig vielen Zeilen. Mit Version 21.2 ist eine Erweiterung hinzugekommen, die es erlaubt, von einer recht strengen Hierarchie Seite => Region

=> Eingabefeld/Schaltfläche abzuweichen. Mit dieser Version ist es möglich, alle Komponenten wahlfrei überall auf der Seite zu verteilen. Grund genug, sich einmal das Konzept anzusehen, denn es gilt für alle Komponenten.

6.2.1 Anordnung von Regionen auf der Seite

Beschäftigen wir uns zunächst mit der Platzierung von Regionen im Anzeigebereich `Body`. Dort werden oft sehr viele Regionen platziert, um zum Beispiel ein zweispaltiges Layout mit Regionen, die unabhängig voneinander neben- und untereinander dargestellt werden können, zu realisieren, oder ein statisches Layout von zum Beispiel vier Regionen, zwei nebeneinander, zwei untereinander, mit jeweils gleichen Größen. Damit Sie die Möglichkeiten in Anwendung sehen und mit den einzelnen Funktionen vertraut werden, werden wir die Startseite unserer Anwendung um zwei weitere Regionen erweitern, die wir dann auf der Seite hin- und herschieben. Um den Überblick nicht zu verlieren, nennen wir die Regionen, die derzeit auf der Seite liegen, `Region 1` (`Klassischer Bericht`) und `Region 2` (`Diagramm`). Alle weiteren Regionen, die nun nach und nach auf der Seite platziert werden, werde ich analog ansprechen.

Das Layout einer Seite basiert beim Universal Theme auf einem System aus zwölf Spalten, die die Seite in gleicher Breite unterteilen. Die Zahl 12 bietet sich an, weil sie durch 2, 3 sowie 4 teilbar ist. Dadurch entstehen viele Kombinationsmöglichkeiten.

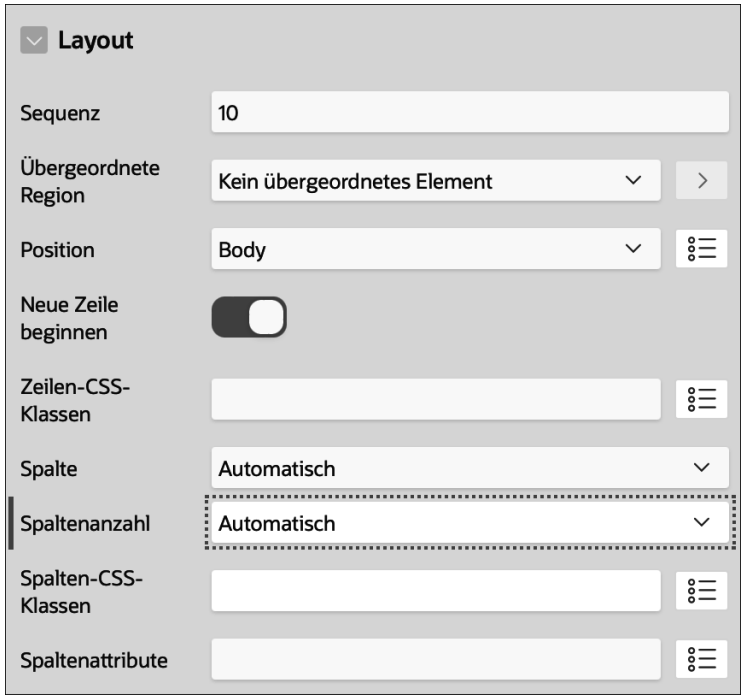


Abbildung 6.15 Layoutattribute einer Region

Derzeit liegen die beiden Regionen auf <Home> noch nebeneinander. Lassen Sie uns das ändern, indem Sie das Attribut Layout • Neue Zeile beginnen von Region 2 auf Ja stellen und das Attribut Layout • Spaltenazahl von Region 1 auf den Wert Automatisch zurückstellen. Nun sind die beiden Regionen auf der Seite in ihrer »Grundposition«. Starten Sie die Seite, und sehen Sie sich das Ergebnis an. Beide Regionen nehmen den gesamten Bereich des Anzeigepunkts Body ein. Mittels <Entwicklerleiste> • Informationen • Layoutspalten anzeigen erkennen wir die zwölf Spalten, in die die Anzeigebreite unterteilt ist.

Versuchen wir als Nächstes, Region 1 zu zentrieren. Sie erreichen dies, indem Sie das <allgemeine Regionsattribut> • Layout • Spalte auf 3 und Layout • Spaltenanzahl auf 8 stellen. Die Seitenvorschau passt die Darstellung von Region 1 an: Rechts und links von der Region wird nun freier Platz angezeigt. Hovern Sie einmal die Regionsüberschrift im Canvas oder dem linken Seitenbereich. Es erscheint ein Tooltipp mit den Angaben zu den Layouteinstellungen. Noch etwas: Als Sie das Attribut Layout • Spalte auf 3 geändert hatten, standen Ihnen im Attribut Layout • Spaltenanzahl nur noch die Einträge 1 bis 10 zur Verfügung. APEX kalkuliert automatisch, wie viele Spalten auf der Seite noch zur Verfügung stehen und gibt einen Fehler aus, falls die Zahl 12 überschritten würde.

Speichern Sie die Einstellungen, und sehen Sie sich das Ergebnis an: Region 1 wird schmaler und zentriert auf der Seite dargestellt.

Stellen Sie das ursprüngliche Layout wieder her, bei dem die beiden Regionen nebeneinander dargestellt wurden. Hierzu müssen die Spaltenfestlegungen von Region 1 wieder auf Automatisch zurückgenommen werden und Region 2 nicht auf eine neue Zeile gestellt werden. Das ist das zweite Konzept des Grid Layouts: Regionen eines Bereichs werden in Zeilen organisiert. Wenn zwei Regionen untereinander angeordnet werden, befinden sich diese in unterschiedlichen Zeilen. Da die rechte Region die Einstellung erhalten hat, keine neue Zeile anzulegen, wird sie in der gleichen Zeile wie die linke Region platziert. Da beide Regionen zudem keine expliziten Einstellungen bezüglich der Spaltenanzahl machen, die sie benötigen, werden sie gleichmäßig auf der Seite verteilt.

Kombinieren wir nun die beiden Möglichkeiten, indem wir Region 2 breiter darstellen. Ändern Sie für Region 2 das Attribut Layout • Spaltenanzahl auf 8, speichern Sie die Änderungen, und sehen Sie sich das Ergebnis an. Region 2 nimmt nun zwei Drittel des verfügbaren Platzes der Seite ein.

Erstellen Sie eine dritte Region (ID • Überschrift • Region 3) vom Typ Statischer Inhalt, und platzieren Sie sie so, dass sie unterhalb der beiden existierenden Regionen in einer eigenen Zeile angezeigt wird. Sehr einfach geht das Erstellen von der Hand, wenn Sie mit der rechten Maustaste auf den Eintrag Body der hierarchischen Seitenübersicht klicken und den Befehl Region erstellen wählen. Speichern und betrach-

ten Sie das Ergebnis. Region 3 nimmt nun den gesamten Platz unterhalb der beiden vorherigen Regionen ein. Das horizontale Layout wird also pro Zeile neu berechnet. Fügen Sie eine vierte Region (Region 4) rechts neben Region 3 hinzu, und legen Sie fest, dass Region 4 nur 4 Spalten breit sein soll. Es ergibt sich ein Layout mit einer 1:2-Verteilung für die obere und einer 2:1-Verteilung für die untere Zeile.

Legen Sie nun fest, dass beide Zeilen die gleiche Verteilung haben, zum Beispiel beide eine 1:2-Verteilung. Betrachten Sie die Seite. Zwei Dinge fallen auf: Zum einen sind die Regionen der oberen Zeile unterschiedlich hoch, Region 2 ist höher, weil die Grafik mehr Platz benötigt als die Berichtsregion. Zum Zweiten fällt auf, dass oberhalb von Region 3, die ja nun die gleiche Breite wie Region 2 hat, freier Raum gezeigt wird, denn Region 1 ist nicht so hoch wie Region 2. Dennoch bleibt Region 3 in den Grenzen ihrer Zeile, die durch die höhere der beiden vorangegangenen Regionen bestimmt wird. Stellen Sie sich das Layout der Seite so vor, als wäre um die Regionen einer Zeile eine weitere Region (tatsächlich ist da auch eine, die allerdings von APEX automatisiert eingefügt wird). Die Regionen können sich also nur in den Grenzen der umgebenden Region bewegen und sind innerhalb dieser Regionen am oberen Rand platziert. Sehen wir uns diese umgebende Region einmal an, indem wir uns den Quelltext der Seite darstellen lassen. Sie hat die Definition aus Listing 6.4.

```
<div class="container">
  <div class="row ">
    <div class="col col-4 apex-col-auto col-start">
      ... Region 1
    </div>
    <div class="col col-8  col-end">
      .. Region 2
    </div>
  </div>
  <div class="row ">
    <div class="col col-4 apex-col-auto col-start">
      ... Region 3
    </div>
  </div>
  <div class="col col-8  col-end">
    ... Region 4
  </div>
</div>
```

Listing 6.4 DIV-Elemente um die Regionen einer Zeile

Das ist gemeint, wenn von einem *DIV-basierten Layout* gesprochen wird: Die Anordnung der Seite geben nur div-Elemente vor, nicht aber wie früher eine große Tabelle

mit Spalten und Zeilen. Dieser alte Ansatz hatte den Nachteil, dass sich die Regionen kaum umsortieren lassen, denn die Spalten und Zeilen einer Tabelle sind sehr statisch. `div`-Elemente sagen zwar etwas über die Struktur der Seite aus (welche Region enthält welche Subregionen etc.), nicht aber über die konkrete Darstellung auf der Seite. Wie viel Platz die Regionen erhalten und ob sie neben- oder übereinander dargestellt werden, wird ausschließlich über CSS-Klassen gesteuert. Der Code aus Listing 6.4 verdeutlicht dies: Das äußere `DIV`-Element hat die Klasse `row`, die beiden in ihm enthaltenen `DIV`-Elemente haben die Klassen `col col-4` bzw. `col col-8`. Wie diese Regionen visualisiert werden, regeln die entsprechenden CSS-Klassen. Und welche Klassen wiederum zur Darstellung herangezogen werden, entscheidet die Seite über sogenannte *Media Queries*, die, basierend auf unterschiedlichen Kriterien, eine CSS-Regel anwenden oder nicht. Das folgende Listing zeigt ein Beispiel für eine von APEX verwendete Media Query:

```
@media only screen and (min-width: 641px)
```

Diese Media Query reagiert also nur, wenn die Darstellung auf einem Bildschirm erfolgt (und nicht etwa eine Sprachausgabe angefordert wurde) und dieser Bildschirm über mindestens 641 px Auflösung verfügt. Sie sehen, dass hinter der Gestaltung der CSS-Klassen für die Seitenberechnung ein erheblicher Aufwand und viel Fachwissen stecken. Seien wir (ich beziehe mich und alle, die nicht ihren Schwerpunkt auf CSS-Gestaltung haben, in dieses *wir* mit ein) froh, dass APEX dies alles für uns geklärt hat und wir die Früchte dieser Arbeit einfach nur verwenden können.

Fortgeschrittene Techniken des Regionslayouts

Lassen Sie uns die Seite noch etwas weiter verfeinern. Erstes Ziel ist es, eine gleichmäßige Verteilung der Regionen auf der Seite zu erreichen. Dazu sollen die Regionen der rechten und linken Seite die gleiche Höhe bekommen. Ein Weg, dies zu erreichen, besteht in der Verwendung von Templateoptionen.

Wählen Sie <Entwicklertools> • Schnellbearbeitung, und bearbeiten Sie die Templateoptionen der Regionen `1` und `2`. Beide Regionen sollen das Attribut Allgemein • Body Height • `320px` erhalten. Beide Regionen sind nun unabhängig von ihrem Inhalt genau 320 Pixel hoch. Würde der Inhalt einer Seite mehr Platz als diese 320 Pixel in Anspruch nehmen, regelte die Template-Option Erweitert • Body Overflow das Verhalten in diesem Fall. Standardmäßig steht dort `Scroll - Default`, es wird also ein Scrollbalken am rechten Regionsrand angezeigt. Optional können Sie zusätzlichen Inhalt aber auch einfach verstecken lassen. Im Ergebnis haben wir also den Platz zwischen `Region 1` und `Region 3` dadurch entfernt, dass wir die Regionen `1` und `2` auf gleiche Höhe gezwungen haben, eventuell zulasten des Inhalts oder eines Scrollbalkens. Was ist aber, wenn wir die Höhen der beiden Regionen erhalten, aber dennoch keinen Platz zwischen den Regionen dulden möchten, wenn also die Regionen

gleicher Breite vertikal direkt aneinanderschließen sollen? Wir möchten unsere Seite also als zweispaltiges Layout darstellen.

Diese Anforderung setzen wir mit zwei zusätzlichen Regionen um, eine für die linke und eine für die rechte Spalte. Linke und rechte Spalte werden nebeneinander dargestellt und die bisherigen Regionen als Subregionen in die entsprechenden Spalten integriert. Im Einzelnen:

- Ziehen Sie zwei weitere Regionen vom Typ `Statischer Inhalt` auf die Anwendungsseite, oder erstellen Sie sie über das Kontextmenü. Platzieren Sie beide Regionen in einer Zeile über den anderen Regionen, und ordnen Sie die beiden Regionen in dieser Zeile nebeneinander an. Benennen Sie die beiden Regionen mit `Spalte links` bzw. `Spalte rechts`.
- Legen Sie für Region `Spalte links` eine Breite von `4` Spalten fest.
- Ziehen Sie anschließend `Region 1` in den Bereich `Sub Regions` der Region `Spalte links`, und lassen Sie sie dort fallen. Alternativ erreichen Sie das gleiche Ziel, indem Sie das Attribut LAYOUT • ÜBERGEORDNETE REGION auf `Spalte links` stellen.
- Stellen Sie die Spaltenzahl der Regionen 1–4 auf `Automatisch` zurück, um den gesamten Platz innerhalb der Region `Spalte links` zu nutzen, und setzen Sie das Attribut LAYOUT • NEUE ZEILE BEGINNEN • `Ja`. Setzen Sie die Templateoption ALLGEMEIN • BODY HEIGHT • `Auto - Default`.
- Ziehen Sie anschließend `Region 3` auf einen Platz unterhalb von `Region 1`. Achten Sie darauf, dass die Region in einem eigenen Kasten unterhalb der Region dargestellt wird. Alternativ legen Sie einfach in `Region 3` das Attribut LAYOUT • ÜBERGEORDNETE REGION auf `Spalte links` fest.
- Verfahren Sie mit den Regionen `Region 2` und `Region 4` analog für die rechte Spalte.
- Revidieren Sie die Templateoptionen der Regionen `1` und `2`, und stellen Sie die Höhe auf `Auto` zurück.

Speichern Sie das Ergebnis, und sehen Sie sich die Seite an. Sie sehen nun, wie Sie Subregionen nutzen können, um Regionen auf der Seite anzuordnen. Störend ist, dass dieser »Trick« auf der Anwendungsseite zu sehen ist, denn die beiden Regionen `Spalte links` und `Spalte rechts` haben eine sichtbare Überschrift und einen störenden Rand. Lassen Sie uns auch dies noch beseitigen, indem wir beide Regionen auf das Regionstemplate `Blank with Attributes` stellen. Speichern Sie die Änderungen, und begutachten Sie das Ergebnis: Der Trick ist unsichtbar, die Regionen fließen in zwei Spalten, unabhängig von den Zeilen. Natürlich existiert das Zeilenkonzept immer noch, aber jeweils innerhalb der Regionen `Spalte links` und `Spalte rechts`. Dies können Sie nutzen, um komplexe Layouts zu realisieren. Natürlich können Sie diese Technik auch auf drei Spalten ausweiten oder ein zweispaltiges Layout innerhalb einer Subregion schachteln. Beachten Sie, dass einer Subregion, zum Beispiel innerhalb von

Spalte links, wiederum zwölf Spalten zur Verfügung stehen, die nun natürlich erheblich schmaler sind als die Spalten der Seite, denn die übergeordnete Region hat ja nicht die volle Breite der Seite.

Das sind die Grundtechniken, die es Ihnen ermöglichen, beinahe beliebige Layouts auf einer Seite zu erstellen. Denken Sie nicht zu kompliziert, sondern suchen Sie die einfachste Lösung, die zum Erfolg führt. Zum Beispiel könnten Sie die Regionen `Spalte links` und `Spalte rechts` auch dadurch unsichtbar machen, dass Sie in den Templateoptionen die Überschrift ausblenden und das Body Padding entfernen. Der noch sichtbare Rand ließe sich durch den Style `Remove Borders` entfernen. Gut, das geht, aber warum? Einfacher ist es, das Template zu ändern. Natürlich nur, wenn Sie an diese Option gedacht hatten, aber diese einfachen Lösungen zu finden, sollte das Ziel der Beschäftigung mit APEX sein.

Dann ist die Benennung einer solchen Region, die auf der Seite unsichtbar sein wird, natürlich nicht unbedingt erforderlich. Aus Dokumentationsgründen ist das aber in jedem Fall empfehlenswert und noch aus einem weiteren Grund: Wählen Sie eine der Regionen, die als Subregionen nun in der linken oder rechten Spalte liegen. Klicken Sie auf das Attribut <ALLGEMEINE REGIONSATTRIBUTE> • LAYOUT • ÜBERGEORDNETE REGION. Alternativ zum Verfahren, Regionen per Drag-and-drop ineinander zu schachteln, könnten wir diese Einstellung ja auch hier vornehmen. Aber welche Region wählen Sie nun als übergeordnete Region, wenn fünf weitere Regionen einfach nur `Neu` heißen?

Gridlayout für Eingabefelder

Eingabefelder werden im Grunde ebenso wie Regionen kontrolliert. Das Attribut LAYOUT • SEQUENZ steuert die Darstellungsreihenfolge, die Sie auch mit Drag-and-drop im Seitendesigner einstellen können. Das Attribut LAYOUT • REGION wiederum steuert die Zugehörigkeit des Seitenelements zu einer Region und kann ebenfalls durch Drag-and-drop kontrolliert werden.

Wie beschrieben, steht uns in einer Region ein Layout mit zwölf Spalten zur Verfügung, in das wir nun die Seitenelemente einfügen. Bedenken Sie, dass abhängig von der Breite der umschließenden Region die Breite der Spalten nun schmaler sein wird. Von der Regionsbreite wird zudem der innere Rand abgezogen (der über eine Template-Option der Region allerdings auch auf `0` gesetzt werden kann) und der verbleibende Platz in zwölf gleiche Spalten aufgeteilt. Eine Analogie zum Regionslayout findet sich auch in der Tatsache, dass es innerhalb der Region Zeilen gibt, die das vertikale Layout kontrollieren. Normalerweise wird ein neues Element innerhalb der Region auch in einer eigenen Zeile dargestellt, doch können Sie dies auch für ein Element durch das Attribut LAYOUT • NEUE ZEILE STARTEN • `Nein` untersagen. Diese Einstellung ist für das erste Seitenelement nicht von Interesse (diese Einstellung würde keine Änderung der Darstellung bewirken), sondern ab dem zweiten Element.

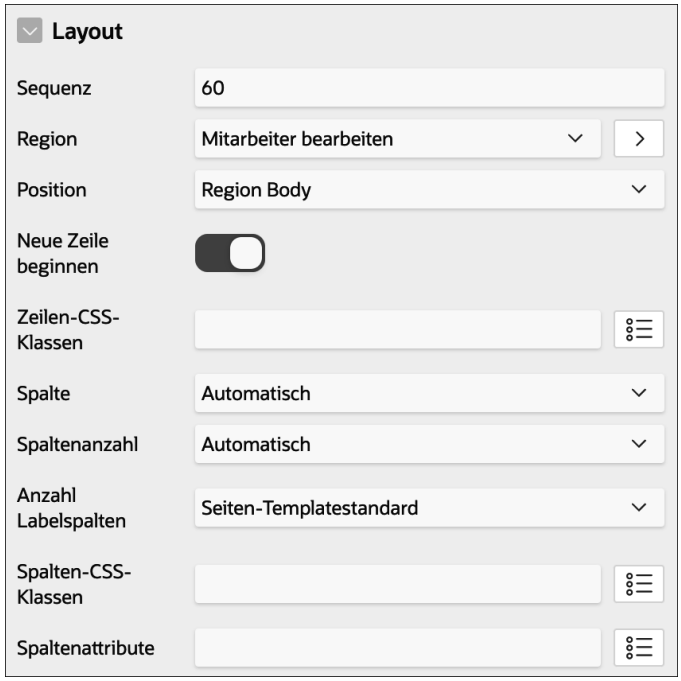


Abbildung 6.16 Layoutoptionen für Eingabefelder

Zu beachten ist aber, dass ausreichend viele Spalten zur Darstellung der Elemente nebeneinander zur Verfügung stehen müssen. Wie das berechnet wird, werden die folgenden Attribute erläutern.

Zunächst steuert das Attribut LAYOUT • SPALTE, ab welcher Spalte das Element dargestellt werden soll. Wichtig hierbei: Diese Angabe bezieht sich auf Label und Elementwert zusammen. Wenn Sie diesen Wert größer `1` einstellen, wird links vom Label Platz geschaffen und das gesamte Element nach rechts verschoben. Analog sorgt das Attribut LAYOUT • SPALTENANZAHL für eine Einrückung von rechts. Wichtig, und im Vergleich zu den Regionsoptionen neu, ist das Attribut LAYOUT • ANZAHL LABELSPALTEN, das angibt, wie viele Spalten der Region für den Elementnamen reserviert werden.

6.2.2 Freie Platzierung von Elementen auf der Seite

Neu in Version 21.2 hinzugekommen ist, dass Elemente weitgehend frei auf der Anwendungsseite platziert werden können. Zwar unterliegen sie immer noch den Vorgaben des Gridlayouts, aber Schaltflächen und Eingabefelder sind nun nicht mehr an Regionen gebunden, sondern können auch in Anzeigebereichen auf der Seite platziert werden. So ist es nun also möglich, eine Schaltfläche in den Breadcrumb-Bereich oder neben die Navigationsleiste zu stellen oder Text freistehend auf der Seite zu platzieren. Neu hinzugekommen ist auch ein Regionsanzeigebereich `Full Width Content`.

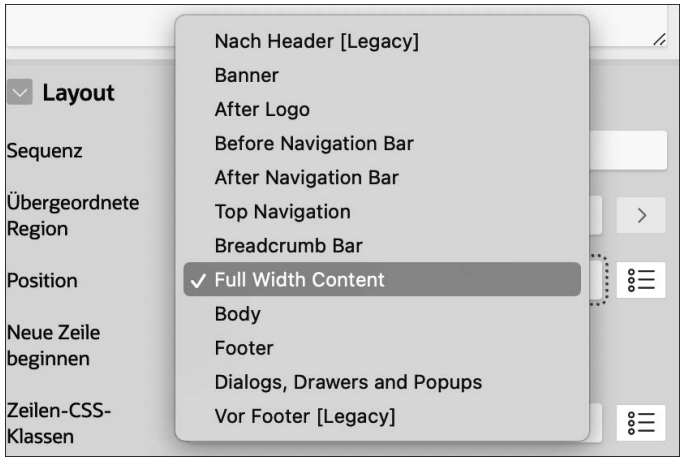


Abbildung 6.17 Steuerung der Platzierung von Regionen und Seitenelementen

Zu den neuen Anzeigeregionen gehören `Banner` (oberhalb der Navigationszeile) und der `Full Width Content`, der die gesamte Seitenbreite einnimmt und insofern nicht an das Gridlayout gebunden ist. Für Schaltflächen und Eingabefelder gibt es interessante Positionen, wie etwa rechts neben dem Anwendungslogo oder rechts neben der Navigationsleiste (wo in unserer Anwendung angezeigt wird, welcher Benutzer angemeldet ist). Abbildung 6.18 zeigt einige der neuen Positionen.

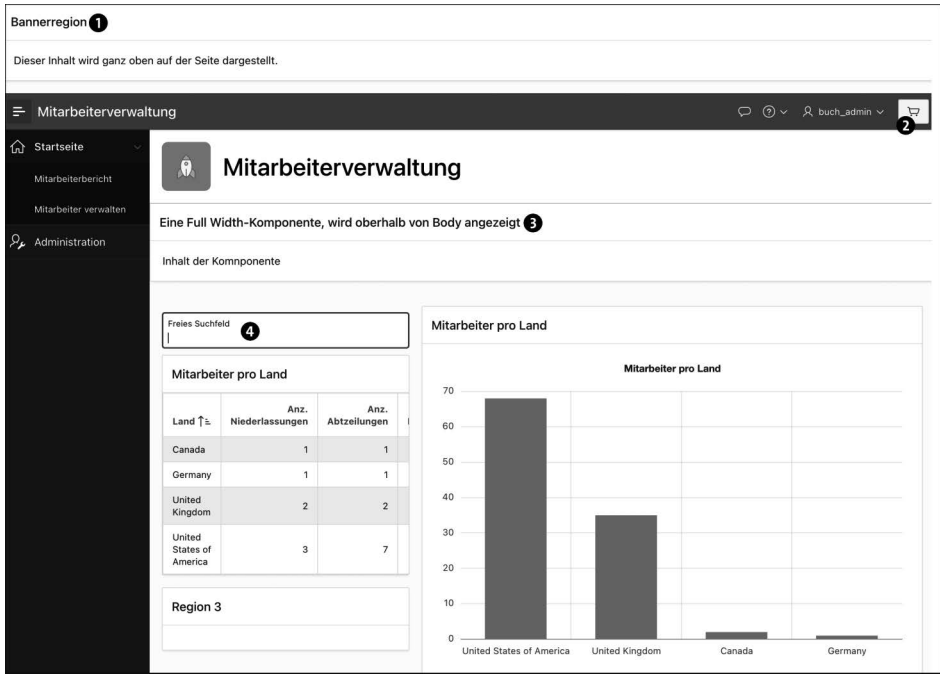


Abbildung 6.18 Einige der neuen Positionierungsoptionen

❶ Banner-Region

❷ Schaltfläche neben Navigationsleiste

❸ Komponente, die die gesamte Bildschirmbreite einnimmt

❹ Frei positioniertes Eingabefeld

Durch diese Möglichkeiten wird viel bislang erforderlicher JavaScript-Code eingespart und gleichzeitig die Visualisierungsmöglichkeit verbessert.

6.3 Darstellung der Komponenten steuern

Seiten, Regionen und Seitenelemente werden, wie wir gesehen haben, über Templates in HTML-Code »übersetzt«, der dann auf der Anwendungsseite dargestellt werden kann. Templates und ihre Optionen spielen daher eine zentrale Rolle bei der Visualisierung einer APEX-Anwendungsseite.

6.3.1 Templates

Templates in APEX dienen der Dekoration der Daten mit HTML-Elementen. Sie sind durch ein reiches CSS-Klassensystem angereichert und steuern so das optische Erscheinungsbild.

Seitentemplates

Anwendungsseiten besitzen ebenso wie alle anderen Komponenten, die eine Visualisierung im Browser benötigen, ein Template. Welches Template verwendet wird, haben wir nur mittelbar eingestellt, indem wir beim Assistenten zur Erstellung der Anwendung das grundsätzliche Layout vorgegeben hatten. Die Vorgaben könnten durch Klick auf die Optionen des Attributes Darstellung (❶ in Abbildung 6.19) eingesehen und im Bereich Navigation ❷ geändert werden.

Die Einstellungen, die hier vorgenommen wurden, setzen aber lediglich den Standardwert für neue Seiten. Jede Komponente verfügt über ein Standardtemplate, und welches das ist, können Sie sich anzeigen lassen. Navigieren Sie zu <Gemeinsame Komponenten > • Themes, und wählen Sie in der Liste das `Universal Theme 42` (es ist das einzige angebotene). Klicken Sie im folgenden Dialog auf den Regionselektor Komponentenstandards.

Auf der Anwendungsseite finden Sie das aktuell eingestellte Template im Attribut Darstellung • Seitentemplate. Die Optionen sind an beiden Stellen gleich, doch gilt die Einstellung auf der Anwendungsseite natürlich nur für die aktuelle Seite. Ändern Sie das Standardtemplate in den Attributen des Themes, hat dies Auswirkungen auf all die Anwendungsseiten, die hier das Standardtemplate über den Eintrag `Theme-Standard` referenzieren.

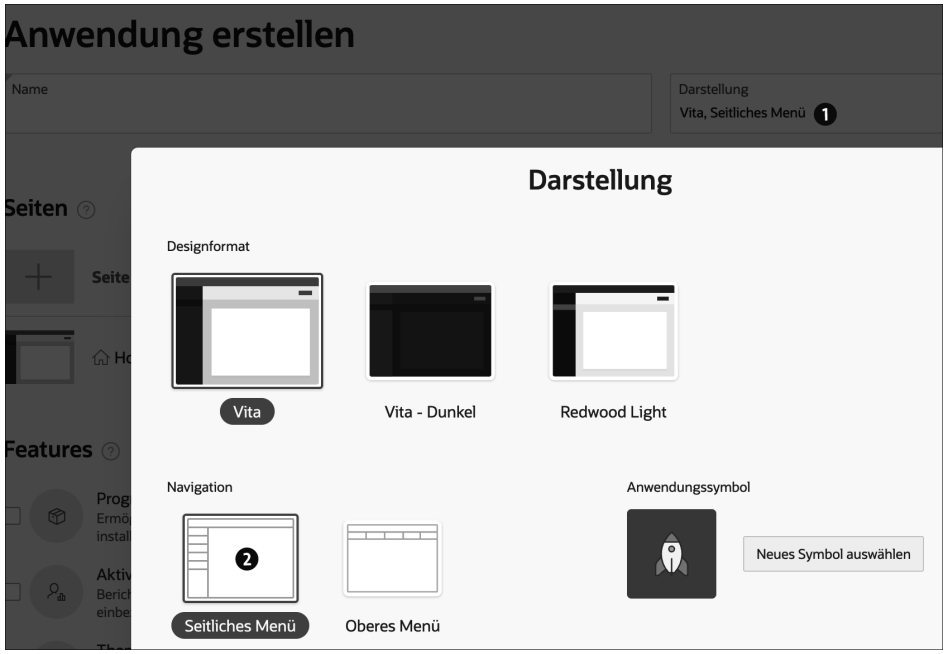


Abbildung 6.19 Indirekte Einstellung des Seitentemplates

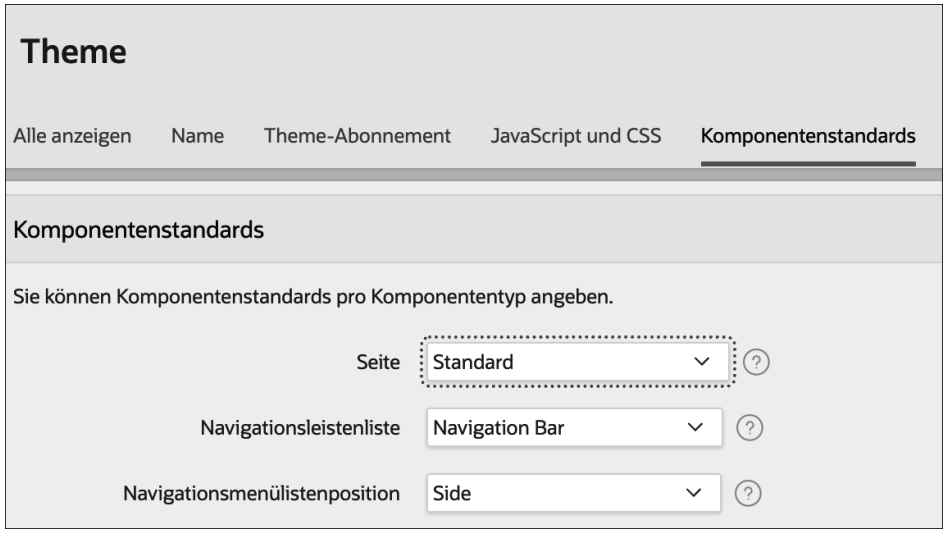


Abbildung 6.20 Festlegung der Standardtemplates für Komponenten

Das Seitentemplate kontrolliert das Gesamtlayout der Seite, legt also fest, wo die Titelleiste, die Navigationsleiste, das Navigationsmenü, der Breadcrumb und so weiter angezeigt werden. Die Optionen beschäftigen sich vor allem mit den Seitenbereichen, die links, links und rechts oder nur rechts dargestellt werden können. Manchmal

kann es sinnvoll sein, einen rechten Seitenbereich zusätzlich anzeigen zu lassen, um zum Beispiel allgemeine Aufgaben dort unterzubringen. Sie kennen diese Optik von der Entwicklungsumgebung, wo diese Aufteilung häufig verwendet wird.

Die verfügbaren Seitentemplates und ihr grundlegendes Aussehen können Sie in der <Universal Theme Anwendung> unter Design • Page Templates einsehen.

Unabhängig vom Seitentemplate kann eine Anwendungsseite mit dem Attribut Darstellung • Seitenmodus als »normale« Anwendungsseite oder als modales oder nicht modales Dialogfeld angezeigt werden. Der Unterschied zwischen modal und nicht modal besteht darin, dass ein modales Dialogfeld keine Interaktion mit der Seite, über der das modale Dialogfeld angezeigt wird, zulässt, das nicht modale Dialogfeld dagegen schon. Als Anwendungsfall für ein nicht modales Dialogfeld könnten Sie sich eine Seite mit Hilfstexten vorstellen, die die Verwendung der aktuellen Seite ermöglicht. Das nicht modale Dialogfeld könnte in diesem Fall hin- und hergeschoben und die darunter befindliche Seite normal bedient werden.

Bitte beachten Sie, dass die Anwendungsseite nicht direkt gestartet werden kann, wenn Sie den Seitenmodus auf ein modales oder nicht modales Dialogfeld einstellen. Der technische Grund hierfür besteht darin, dass diese Seite als Dialog von einer anderen Seite aus gestartet werden muss.

Regionstemplates

Auch Regionen verfügen über entsprechende Templates. Wichtig zu verstehen ist, dass Regionstemplates unabhängig vom konkreten Regionstyp den Bereich kontrollieren, in dem der konkrete Regionstyp seine Daten visualisiert. Damit meine ich, dass eine Region vom Typ `Interaktives Grid` das gleiche Template besitzen kann wie eine Region vom Typ `Statischer Content`, denn das Regionstemplate organisiert lediglich das Aussehen des umgebenden Rahmens der konkreten Daten der Region. Einige Regionstypen verfügen selbst über typspezifische Templates, die sich um die Visualisierung der konkreten Daten innerhalb der Region kümmern.

Das Regionstemplate kontrolliert also, ob die Region eine Überschrift anzeigen wird und wie groß der innere Abstand des Inhalts der Region vom Rand ist, ob ein Rahmen um die Region gezogen wird oder nicht. Eine wesentliche Aufgabe des Regionstemplates ist die Umsetzung des Gridlayouts, es kontrolliert, wo die Region beginnt und wie breit sie dargestellt wird. Daraus folgt, dass jede Region ein Template besitzt, selbst wenn es optisch gar nicht zu sehen ist. Es hat in diesem Fall zum Beispiel das Regionstemplate `Blank with Attributes`. Dieses Template zeigt keine Regionsüberschrift, keinen Rand und keine sichtbaren Spuren auf der Anwendungsseite, unterliegt aber den Regeln des Gridlayouts. Wir hatten dieses Template bei der Anlage eines zweispaltigen Layouts verwendet.

Die einzelnen optischen Unterschiede zwischen den Regionstemplates sind hier weniger interessant, das lässt sich leicht selbst ausprobieren. Wichtiger ist zu verstehen, dass die Regionstemplates sich nicht um die unterschiedlichen Regionstypen kümmern, sondern lediglich um den Rahmen, der die Daten der Region umgibt.

Regionstyptemplates

Da die Regionstemplates sich nicht um die Anforderungen der unterschiedlichen Regionstypen kümmern, sondern eher die Integration der Region in das Gridlayout kontrollieren, benötigen einige Regionstypen noch ein Template, um die Daten entsprechend ihrem Typ unterschiedlich darstellen zu können. Einige Regionstypen haben dabei eine große Anzahl unterschiedlicher Templates und können ihr Aussehen massiv ändern, andere besitzen überhaupt keine Regionstyptemplates.

Verfügt ein Regionstyp über eigene Templates, finden diese sich, wie zu erwarten, unter den Regionstyp-spezifischen Attributen. Ein Regionstyp, der sich chamäleonartig verändern kann, ist der klassische Bericht. Wir haben diesen Regionstypen auf <Home> • `Region 1` eingesetzt, daher können wir die Regionstyptemplates an diesem Beispiel untersuchen.

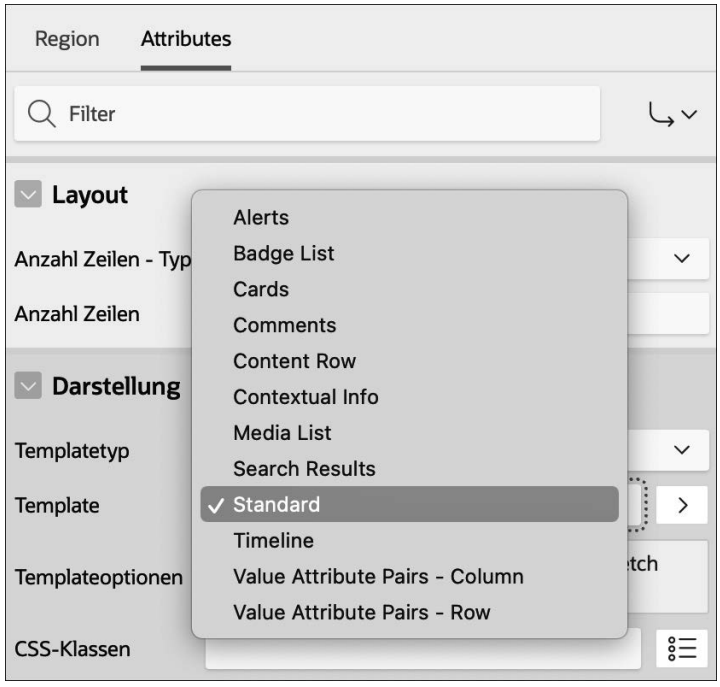


Abbildung 6.21 Die Regionstyptemplates von Region 1

Abbildung 6.21 zeigt die verfügbaren Templates für einen klassischen Bericht. Klassische Berichte sind simple Konstrukte ohne besondere JavaScript-Funktionalität. Das

macht sie zunächst weniger attraktiv als die leistungsfähigeren interaktiven Berichte und Grids, doch andererseits auch optisch flexibler, da nicht auf eine entsprechende Funktionalität Rücksicht genommen werden muss. Und so können klassische Berichte sich bis zur Unkenntlichkeit verkleiden. Beispiele finden sich in der <Universal Theme Anwendung> an verschiedenen Stellen, die Sie im Menüeintrag Components unter den Bezeichnern der Regionstyptemplates finden.

Dies sind nur Beispiele, auch andere Regionstypen haben ähnliche Funktionalität. So können zum Beispiel Listen ebenso unterschiedlich aussehen, tatsächlich zeigt die <Universal Theme Anwendung> an einigen Stellen Implementierungen entweder als klassischer Bericht oder als Liste, so zum Beispiel bei Components • Media List.

Seitenelementtemplates

Auch Seitenelemente wie Eingabefelder oder Schaltflächen besitzen Templates. Diese Templates steuern bei Seitenelementen vor allem die Position des Bezeichners, zum Teil aber auch das Verhalten. Dies können Sie beim Seitenelementtemplate `Optional – Floating` beobachten, das zur Folge hat, dass der Elementbezeichner in das Element integriert wird und sich dynamisch verkleinert, wenn das Element einen Wert enthält oder in das Element geschrieben wird.

Schaltflächentemplates kontrollieren vor allem, ob die Schaltfläche als Symbolschaltfläche oder mit einem Text dargestellt wird. Auch eine Kombination ist möglich.

6.3.2 Templateoptionen

Unabhängig von der Komponente, für die sie erstellt wurden, verfügen viele Templates über Optionen zur Parametrierung der optischen Darstellung. Jedes Template kann bei APEX eigene Optionen definieren. Technisch bedeutet dies, dass dem HTML-Code, der durch das Template erzeugt wird, CSS-Klassen hinzugefügt oder von diesem entfernt werden.

Sie haben mit den Templateoptionen bereits gearbeitet, als wir einen Dialog zu Kontrolle dieser Optionen mit der <Entwicklerleiste> • Schnellbearbeitung dargestellt hatten, indem wir auf das Schraubenschlüsselsymbol der Region geklickt hatten. Alternativ finden Sie im Attribut Darstellung der Komponente eine Schaltfläche, die alle vom Standard abweichenden Optionen im Titel führt und Sie zu einem analogen Dialog in der Entwicklungsumgebung führt (❶ in Abbildung 6.22).

Da jedes Template eigene Templateoptionen definieren kann, ist es unmöglich, hier einen generellen Überblick zu geben. Diesen Überblick zu geben, ist eine der Hauptaufgaben der <Universal Theme Anwendung>, denn unterschiedliche Optionen werden dort nebeneinander dargestellt und die Einstellungen dokumentiert. Ein weiterer Grund, sich diese Anwendung zu installieren oder einen Favoriten auf die Online-Version zu legen.

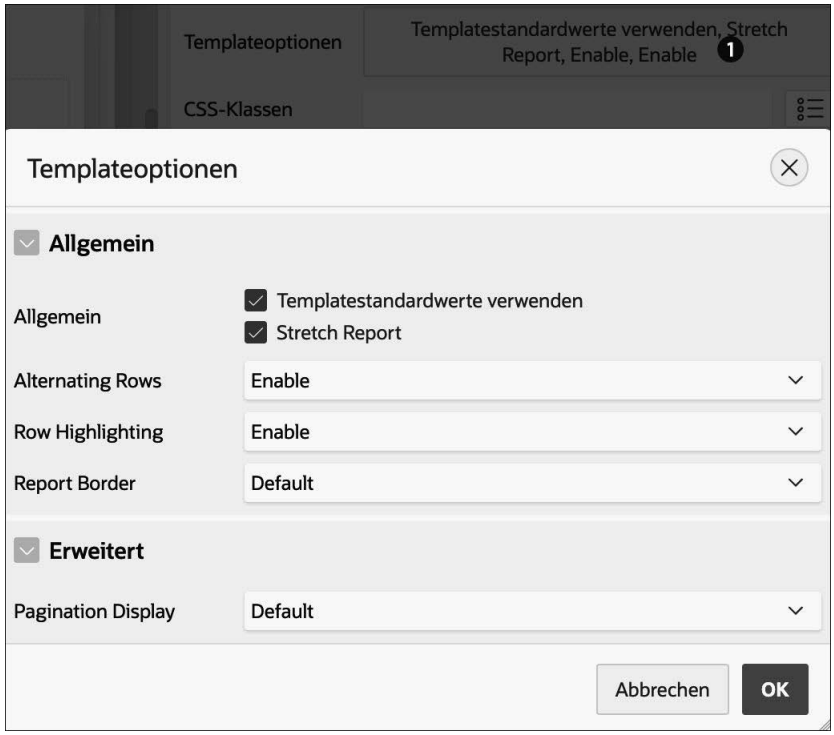


Abbildung 6.22 Optionen eines Templates

Wenn Sie mit den Mitteln der Schnellbearbeitung die Templateoptionen bearbeiten, ist die Unterteilung in Regionstemplate und Regionstyptemplate ebenfalls vorhanden. Die Templateoptionen des Regionstemplates finden Sie im Reiter REGION, während die Optionen im Reiter ATTRIBUTE sich auf die Regionstyptemplates beziehen. Allerdings ist es in diesem Dialog nicht möglich, das Regionstyptemplate selbst zu ändern, dies geht nur in der Entwicklungsumgebung.

6.3.3 Neue Entwicklungen

Das Template-Konzept ist in APEX seit den ersten Versionen implementiert und hat über die Zeit eine kontinuierliche Weiterentwicklung erfahren. Neuere Entwicklung läuten aber eventuell eine Zeitenwende ein: Die Anforderungen an die Visualisierung der Komponenten steigen, und das APEX-Entwicklerteam kommt zunehmend zu der Einsicht, dass ein dynamisch berechneter HTML-Code auf Basis von Attributen der Komponenten mehr Flexibilität schaffen könnte als das bisherige System. Vielleicht entwickelt sich APEX also mehr und mehr von diesem Mechanismus weg. Ein Beispiel für diese Tendenz sehen wir bei den sehr komplexen interaktiven Grids. Die unterschiedlichen Möglichkeiten sind dort bereits nicht durch Templates umgesetzt. Ob

sich dieser Trend fortsetzen wird und welche Auswirkungen dies auf die Bedienung haben wird, ist derzeit aber noch nicht abzuschätzen.

6.3.4 Anpassung des Layouts

APEX bietet viele Möglichkeiten, das Layout der Anwendungsseite und einzelner Komponenten zu kontrollieren, ohne Sie zu handgemachtem CSS zu zwingen. Zwar ist auch das möglich, doch sollten Sie den anderen Möglichkeiten unbedingt den Vorzug geben. Grund hierfür ist, dass APEX sich darum kümmern wird, dass diese Funktionalität erhalten bleibt und auch bei neuen APEX-Versionen funktionieren wird, selbst wenn die unterliegende Technik sich ändert. Passen Sie CSS selbst auf Ihre Bedürfnisse an, ist dies nicht sichergestellt, und ein Versionswechsel in APEX kann unter Umständen erhebliche Aufwände verursachen. Und bitte glauben Sie mir: Ich weiß, wovon ich spreche, ich habe das mehrfach erlebt.

Über die Möglichkeiten der direkten Parametrierung eines Templates über die Templateoptionen hinaus stellt Ihnen APEX eine große Anzahl vordefinierter CSS-Klassen zur Verfügung, mit deren Hilfe Sie viele der üblichen Anpassungsarbeiten erledigen können.

Vorgefertigte CSS-Klassen

Wieder einmal ist eine Beispielanwendung Ihre Informationsquelle, nämlich die Anwendung *Universal Theme Reference*. Das Verständnis der Beispiele mag manchmal nicht leicht sein, das hängt mit der Flexibilität der CSS-Klassen zusammen. An einem Beispiel möchte ich verdeutlichen, wie die Dokumentation zu lesen ist. Zur Kontrolle des äußeren Abstands um ein Element finden sich unter dem Bereich Reference • Layout Modifiers • Margin einige CSS-Klassen, die nach dem System `margin` – *Richtung* – *Abstand* zusammengestellt werden können. Als Trennzeichen werden Minuszeichen verwendet. Für die beiden Optionen Richtung und Abstand sind unterschiedliche Werte möglich: `top` – `right` – `bottom` – `left` für die Richtungen und `none` – `sm` (small) – `md` (medium) – `lg` (large) und `auto` für den Abstand. Aus diesen Optionen können die CSS-Klassen nun zusammengestellt werden: `margin-top-lg` liefert also einen großen oberen Abstand.

Ähnlich funktioniert das Ganze, wenn CSS-Klassen für Prozentzahlen oder Pixelwerte definiert werden. Im Bereich Reference • Layout Modifiers • Width werden CSS-Klassen vorgestellt, die die Darstellungsbreite von zum Beispiel Schaltflächen oder Eingabefeldern kontrollieren können. Die CSS-Klassen werden dort zum Beispiel `wamount` genannt, was auf den ersten Blick verwirrt. Gemeint ist `w`(width)`<amount>`, wobei `amount` einen Wert zwischen 10 und 800 Pixel in Zehnerschritten annehmen kann, also zum Beispiel `w400`. Prozentangaben werden nach dem gleichen Prinzip, aber mit

nachgestelltem `p`, also `w40p`, dargestellt. Analog funktioniert das für die übrigen CSS-Klassen `mnw` (minimum width) und `mxw` (maximum width).

Die mitgelieferten CSS-Klassen finden Sie an unterschiedlichen Stellen, zum Beispiel im Bereich Design • Grid Layout • Visibility Classes. Ein weiteres Beispiel sehen Sie im Bereich Reference • Layout Modifiers. Dort können Sie den Fluss der Elemente, die Randeinstellungen außen (`margin`) und innen (`padding`) kontrollieren, sowie die Breite, Höhe und weitere Eigenschaften.

Content Modifiers kümmern sich um Schriftformatierung, Color and Status Modifiers stellen ein konsistentes Farbschema bereit. Ein weiteres, besonders mächtiges Konstrukt sind CSS Variables • Component Variables, mit denen Sie die Voreinstellungen für alle mitgelieferten Komponenten an Ihre Wünsche anpassen können.

Verwendung der Klassen in APEX

Die CSS-Klassen werden an unterschiedlichen Stellen hinterlegt. Welche Stelle genau für Ihren Einsatzzweck die richtige ist, ist nicht immer leicht herauszufinden. Da hilft nur, ein wenig zu experimentieren. Überall gilt, dass mehrere CSS-Klassen, getrennt durch Leerzeichen, übergeben werden können.

Hier zunächst die möglichen Attribute, die CSS-Klassen aufnehmen können:

- Auf Seitenniveau
  Für Anwendungsseiten bietet sich vor allem das Attribut Darstellung • CSS-Klassen an. Zwar existiert ein eigener Bereich CSS bei den Seitenattributen, dieser dient jedoch eher dazu, seitenweit gültiges CSS inline zu definieren. Das erste Attribut nimmt CSS-Klassen für die Darstellung der Seite auf.
- Auf Regionsniveau
  Eine Ebene tiefer, auf Regionsniveau, existiert ebenfalls ein Attribut Darstellung • CSS-Klassen, das den umgebenden Container um den Regionsinhalt beeinflusst. Abhängig vom konkreten Regionstyp können weitere CSS-Attribute vorhanden sein. Bei einer Berichtsregion vom Typ `interaktives Grid` zum Beispiel steht je ein Attribut Layout • Zeilen-CSS-Klassen bzw. Layout • Spalten-CSS-Klassen zur Verfügung. Zudem kann jede Spalte mit einem eigenen Attribut Darstellung • CSS-Klassen ausgestattet werden.
- Auf Seitenelementniveau
  Ähnliche Attribute stehen auch für Seitenelemente zur Verfügung. Eingabefelder haben gleich zwei Möglichkeiten, einmal beim Attribut Darstellung • CSS-Klassen und dann noch beim Attribut Erweitert • CSS-Klassen. Der Unterschied: Das

erste Attribut bezieht sich auf den umschließenden Bereich um das gesamte Seitenelement inklusive des Labels, während das zweite Attribut CSS-Klassen für das HTML-Eingabeelement liefert.

Schaltflächen verfügen ebenfalls über ein entsprechendes Attribut DARSTELLUNG • CSS-KLASSEN.

CSS-Enthusiasten finden mit diesen Optionen sicherlich weitreichende Möglichkeiten, APEX optisch an ihre Bedürfnisse anzupassen. Einen kleinen Warnhinweis möchte ich allerdings auch loswerden: Verlieren Sie sich nicht in diesen Möglichkeiten, wenn Sie irgendwann noch einmal fertig werden wollen mit Ihrer Anwendung. Lassen Sie sich vor allem nicht verleiten, APEX dazu zu bringen, »auszusehen wie ...«, sondern versuchen Sie zunächst, mit den mitgelieferten Möglichkeiten zu leben. Es sei denn, das Layout Ihrer Anwendung ist von zentraler Bedeutung, und Sie haben mindestens einen Kollegen, der fließend CSS spricht und solche Arbeiten liebt.

6.3.5 Verwendung von Icons

In eine ähnliche Kerbe wie die mitgelieferten CSS-Klassen schlägt auch die Empfehlung, sich zunächst bei den mitgelieferten Icons umzuschauen. Basierend auf der Library FontAwesome hat APEX vor einigen Releases seine eigene Version FontApex erstellt. Beiden Systemen gemeinsam (und für solche Zwecke mittlerweile Standard) ist, dass es sich bei den Icons um Schriftarten handelt, ähnlich vielleicht der legendären Zapf Dingbats. Icons als Schriften zu generieren, ist deshalb sinnvoll, weil damit flexible Möglichkeiten der Änderung des Aussehens einhergehen können. Für diese Möglichkeiten bietet APEX in den neueren Versionen ab 21.1 nun eine zentrale Anlaufstelle. Diese finden Sie in <UTILITYS> • FONT APEX SYMBOLE. Die Seite zeigt einen Bericht, der mittels einer Facettensuche durchstöbert werden kann. Jedes Icon kann nach Wunsch weiter editiert werden (❶ in Abbildung 6.23). Hilfreich sind hier insbesondere die MODIFIKATIONEN und STATUS-Attribute, wie in ❷ in Abbildung 6.23 zu sehen.

Nach Auswahl aller Einstellungen können Sie im unteren Bereich sowohl die erforderlichen CSS-Klassen als auch einen beispielhaften HTML-Code auswählen, um dieses Icon auf Ihrer Seite zu platzieren. Solche Icons können Sie an sehr vielen Stellen in APEX verwenden, zum Beispiel in Berichtsspalten, Schaltflächen oder auch in Navigationslisten. Haben Sie sich einmal in die Verwendung eingearbeitet, ist es auch nicht schwer, solche CSS-Klassen dynamisch zu erzeugen und in Berichten anzeigen zu lassen. In Kapitel 8, »Berichte«, werden wir diese Möglichkeiten eingehender untersuchen.

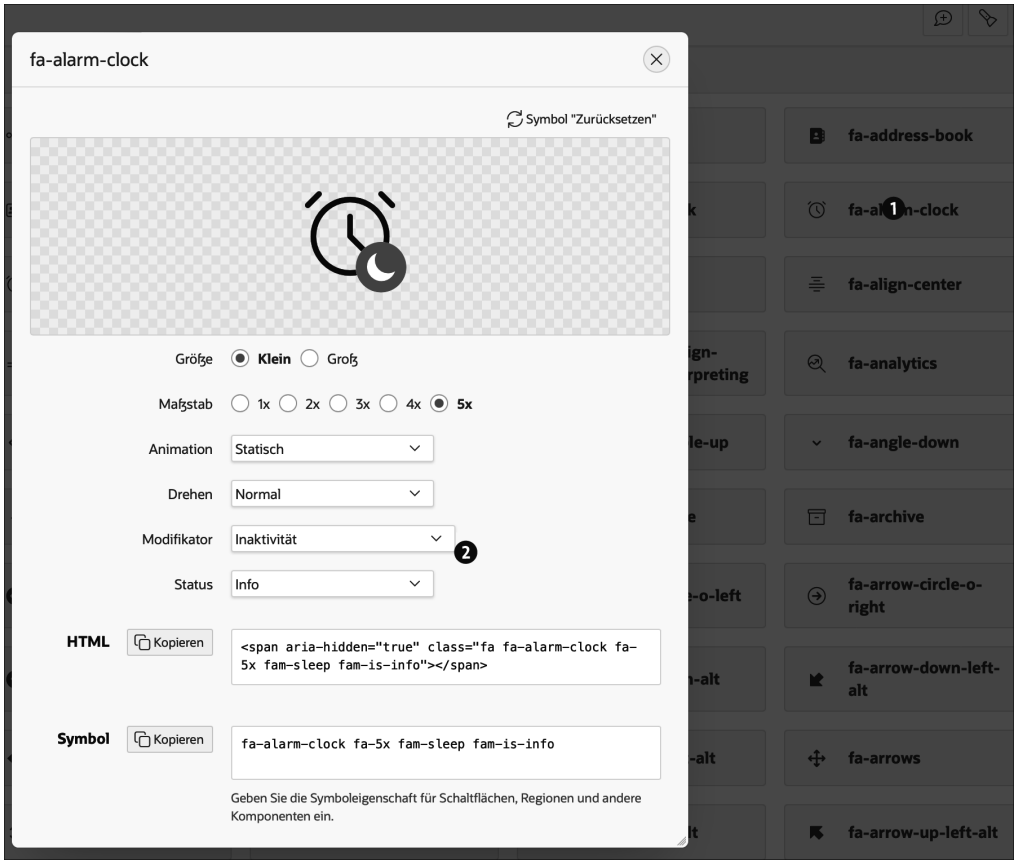


Abbildung 6.23 Einstellmöglichkeiten für Icons

6.3.6 Verwendung von Bildern

Version 22.1 von APEX erweitert die Behandlung von Icons und Bildern in APEX grundlegend. Schon früher war es möglich, einer Anwendung ein Anwendungs-Icon zuzuordnen, das war auch für Regionen möglich, doch mit dieser Version ist der Prozess vereinfacht und gleichzeitig deutlich erweitert worden.

Anwendungs-Icon

Wird eine neue Anwendung erstellt, ist die Wahl eines Icons für diese Anwendung nun direkt im Assistenten für die Anwendung möglich. APEX liefert eine große Anzahl vordefinierter Icons mit, aus denen gewählt werden kann, zudem kann jedes Icon über unterschiedliche Hintergrundfarben angepasst werden. Alternativ ist es auch möglich, eigene Icons aus dem Dateisystem hochzuladen, sogar eine Skalierung und Beschneidung des Icons für die Anwendung ist möglich.

Dieses Icon wird an unterschiedlichsten Stellen der neu erstellten Anwendung eingesetzt:

- im PageDesigner, in der Symbolansicht aller Anwendungen
- im Anmeldebildschirm der Anwendung
- als *Favicon*, also in der Adressleiste des Browsers
- im Seitentitel, also im Reiter des Browsers, der unsere Anwendung enthält
- in der Breadcrumb-Region der Anwendung

Einmal gewählte Icons können in den <Gemeinsamen Komponenten> • Benutzeroberfläche • Benutzeroberflächenattribute auch nachträglich bearbeitet und geändert werden. Es ist sogar möglich, das Anwendungs-Icon mittels der <Entwicklerleiste> • Anpassen • Anwendungssymbol bearbeiten direkt und dynamisch zu editieren.

Regions-Icon und -bilder

Zur Verwaltung von Icons und Bildern in Regionen steht seit Version 22.1 ein eigener Eintrag in den <Regionsattributen> • Bild zur Verfügung. Bislang (und auch in Version 22.1) existiert nur ein Attribut <Regionsattribute> • Darstellung • Symbol, mit dem das Symbol als CSS-Klasse definiert werden konnte. Die neue Attributgruppe ermöglicht nun, beliebige Dateien direkt hochzuladen und als Icon zu verwenden.

Noch mächtiger ist die Möglichkeit, Bilder in Regionen einzusetzen, zum Beispiel als Hintergrundbild der Anwendungsseite. Hierzu stellt APEX eine neue Darstellungsposition im <Regionsattribut> • Layout • Position zur Verfügung: `Background Image`. Diese Position nimmt dann den gesamten Hintergrund der Anwendungsseite ein. Um dort ein Bild zu platzieren, verwenden Sie als Template der Region das ebenfalls neue Template `Image`. Durch dieses Template erhalten Sie nicht nur die Möglichkeit, im Attribut <Regionsattribute> • Bild • `URL der Datei` ein beliebiges Bild als Hintergrundbild der Anwendungsseite zu laden, sondern können dessen Darstellung durch Templateoptionen noch an Ihre Wünsche anpassen. Diese Templateoptionen haben es in sich, denn Sie können nicht nur die Größe des Bildes beeinflussen, sondern zum Beispiel auch Farbbilder in Schwarz-Weiß- oder Sepia-Bilder ändern, die Form ändern und vieles mehr. Auch diese Templateoptionen lassen sich live auf der Anwendungsseite mittels der <Entwicklerleiste> • Schnellbearbeitung editieren und in ihrer optischen Auswirkung beurteilen.

Regionen vom Typ `Image` können wie alle anderen Regionen auch an beliebiger Stelle auf der Anwendungsseite positioniert werden, zum Beispiel als eingeschachtelte Region einer zweiten Region oder im Bereich des Banners der Anwendung. Die Verwaltung von Bildern zur Aufwertung von APEX-Anwendungen wird durch die neuen

Möglichkeiten deutlich einfacher und leistungsfähiger. Zudem stellen die vielen Filter und Manipulationsmöglichkeiten einen Werkzeugkasten bereit, der die Erstellung unterschiedlicher Anwendungsbilder reduziert, weil das gleiche Bild durch diese Filter im Vorder- oder Hintergrund verwendet werden kann. Während dieser Bereich in früheren Versionen von APEX noch eine Spielwiese für CSS-Profis war, ist durch die neue Funktionalität die Verwaltung von Anwendungs-Icons und Hintergrundbildern jedem Entwickler möglich, so denn entsprechende Bilder zur Verfügung stehen.

6.4 Zwischen Anwendungsseiten navigieren

Die Navigation zwischen Anwendungsseiten ist eine fundamentale Anforderung für alle APEX-Anwendungen. In APEX stehen hierfür mehrere Verfahren zur Verfügung:

- **Schaltflächen**
  Sie haben dies bereits im Einsatz gesehen, wenn zum Beispiel in einem Bericht eine Schaltfläche die Neuanlage eines Mitarbeiters ermöglicht.
- **Links**
  Häufigstes Anwendungsbeispiel ist ein Bearbeitungslink in einem Bericht.
- **Listen**
  In APEX sind Listen dynamische oder statische Linksammlungen. Sie können, ebenso wie klassische Berichte, in unterschiedlichster optischer Erscheinungsform auftreten. Listen sind die Grundlage für Breadcrumbs und das Navigationsmenü.
- **Verzweigungen**
  Verzweigungen werden zu unterschiedlichen Zeitpunkten des Lebenszyklus einer Anwendungsseite berechnet. Sie können zum Beispiel vor dem Rendern auf die Login-Seite verzweigen oder nach der Verarbeitung eines Formulars auf eine Berichtsseite.

6.4.1 Allgemeine Eigenschaften der Navigation

Egal, auf welchem Weg die Navigation ausgelöst wird, können bei der Verzweigung innerhalb einer Anwendung oder beim Navigieren zwischen unterschiedlichen Anwendungen eine Reihe von Angaben gemacht werden. Diese Angaben dienen der Konfiguration der aufgerufenen Anwendungsseite. Die verfügbaren Optionen können wir uns auf <EMP_ADMIN> • `Mitarbeiterübersicht` in dem Spaltenattribut `APEX$Link` • LINK • ZIEL ansehen, indem wir die dortige Schaltfläche klicken.

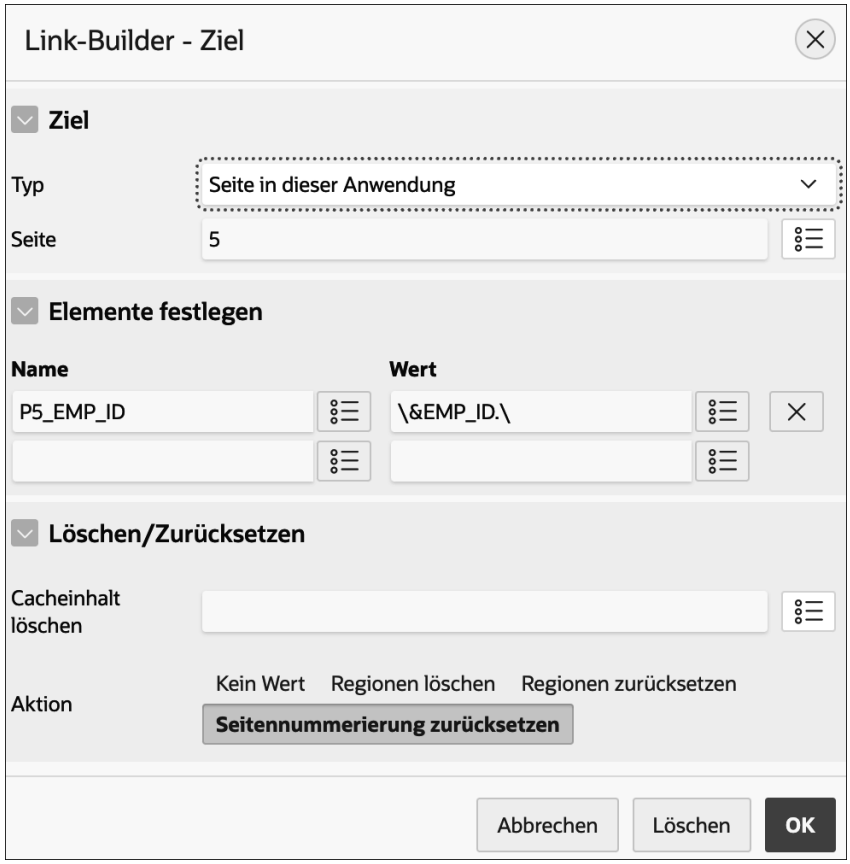


Abbildung 6.24 Attribute eines Links

Zunächst können wir im Attribut Ziel • Typ festlegen, ob wir zu einer Anwendungsseite innerhalb der aktuellen Anwendung, einer anderen APEX-Anwendung oder zu einem sonstigen URL verzweigen möchten. Interessant ist zunächst die Möglichkeit der Verzweigung zu einer Seite in der aktuellen Anwendung. Andere APEX-Anwendungen erweitern die Attribute lediglich um die Auswahl dieser anderen Anwendung, funktionieren ansonsten aber gleich.

Das Attribut Ziel • Seite legt die Seite fest, zu der verzweigt werden soll. Möglich ist hier, entweder die Seiten-ID oder das Seitenalias einzugeben (was ich bevorzuge, da dadurch der URL sprechender wird). Im Bereich Elemente festlegen kann dann definiert werden, dass Seitenelemente der Zielseite mit Werten belegt werden, die auf der aufrufenden Seite bekannt sind. Das können Konstanten sein, aber auch Seitenelementwerte. Wichtig zu verstehen ist, dass die Werte nur beim Rendern der Seite belegt werden und nicht dynamisch aus der aktuellen Seite übernommen werden. Dies ist oftmals kein Problem, kann aber durchaus zu schwer zu verstehendem Fehl-

verhalten führen. In Abbildung 6.24 wird nur ein Seitenelement der Zielseite 5 auf den aktuellen Wert der Spalte EMP_ID gesetzt. Die verfügbaren Spalten (wir befinden uns ja in der Spalte eines Berichts) können über die Symbolschaltfläche neben der Spalte für den Attributwert angezeigt und ausgewählt werden. Möchten Sie mehrere Seitenelementwerte der Zielseite definieren, tragen Sie alle benötigten Namen und deren Werte in der Liste ein. Neben den berechneten Werten können als Wert beliebige konstante Zeichenketten übergeben werden.

Wie funktioniert nun die Übergabe unterschiedlicher Primärschlüsselwerte? Dadurch, dass die Spalte APEX$Link einen Link auf die Formularseite enthält, weiß APEX, dass beim Rendern der Seite ein konkreter Link pro Zeile des Berichts berechnet werden muss. Daher kann zu diesem Zeitpunkt der aktuelle Wert der Spalte EMP_ID ermittelt und im URL verwendet werden. Der Grund, warum dies nur serverseitig möglich ist und nicht dynamische Eingaben des Anwenders referenziert werden können, liegt im erhöhten Sicherheitsbedürfnis von Webanwendungen, denn ein Anwender könnte einen URL manipulieren und so eventuell Mitarbeiter editieren, für die er eigentlich keine Berechtigung hätte. Um dies zu verhindern, wird der URL durch eine Checksumme geschützt, die nur von der Logik innerhalb der APEX-Umgebung berechnet werden kann. Manipuliert ein Benutzer den URL, funktioniert er nicht mehr. Eine Manipulation des Links wäre aber erforderlich, wenn die aktuellen Elementwerte der Anwendungsseite berücksichtigt werden müssten, und das wäre nicht sicher realisierbar.

Das Attribut Löschen/Zurücksetzen • Cacheinhalt löschen ist ein wenig komplizierter zu erklären. Haben Sie ein Formular ausgefüllt und abgesendet, befinden sich diese Werte im Session Status. Öffnen Sie nun diese Seite erneut, könnte es sein, dass sich dort noch Werte aus dem letzten Aufruf der Formularseite befinden, die dann dargestellt würden. Dies wäre zum Beispiel der Fall, wenn Sie keine Primärschlüsselinformation an die Seite übergeben wie bei der Neuanlage eines Datensatzes, denn dann würde die Datenquelle des Formulars keine Daten abfragen. Wenn Sie dies sicher verhindern möchten, können Sie zwei Dinge tun: Zum einen können Sie in diesem Dialog die Seitennummer 5 eintragen, um anzufordern, dass der Cache dieser Seite bereinigt werden soll, zum anderen können Sie nach der Verarbeitung einer Formularseite den Cache dieser Seite durch einen Prozess leeren lassen. Natürlich geht auch eine Kombination beider Verfahren.

Von den restlichen Optionen, insbesondere auch denen im Bereich Erweitert, ist normalerweise nur die voreingestellte Option Löschen/Zurücksetzen • Aktion • Seitennummerierung zurücksetzen von Interesse. Diese Option löst folgendes Problem: Wenn die Seite, auf die Sie verzweigen, einen Bericht darstellt, könnte dieser beim letzten Aufruf über mehr Zeilen verfügt haben, als auf einer Seite darstellbar ist. Der Benutzer könnte daher durch den Bericht gewandert sein und sich Folgeseiten angesehen haben. Wird die Seite nun mit Parametern aufgerufen, die nicht genügend

Zeilen ermitteln, um zum Beispiel zur dritten Folgeseite zu navigieren, ist ein Fehler die Folge. Durch diese Aktion wird der Bericht auf der Zielseite auf die erste Seite zurückgestellt und so sichergestellt, dass auch Zeilen angezeigt werden können.

6.4.2 Schaltflächen

Schaltflächen können direkt für die Navigation eingesetzt werden, etwa wenn sie einen neuen Datensatz anlegen sollen. Oftmals lösen sie aber indirekt eine Navigation aus, weil sie die Verarbeitung der Anwendungsseite anfordern und diese nach der Abarbeitung eine Verzweigung berechnet. Ich möchte hier allerdings auch diese indirekte Navigation betrachten, um nicht in einem weiteren Kapitel noch einmal auf dieses Standardverhalten von Schaltflächen zurückkommen zu müssen.

Eine Schaltfläche hat die bereits beschriebenen Standardattribute für Links, die auch alle genauso bedient werden wie in Abschnitt 6.4.1 beschrieben. Zusätzlich stehen noch die Optionen zur Verfügung, dass Schaltflächen die Anwendungsseite senden können (was einen Submit veranlasst), oder sie können die konkrete Aktivität an eine dynamische Aktion delegieren.

Da die Standardattribute keine neuen Erkenntnisse oder Funktionalität zeigen, beschäftigen wir uns mit den beiden zusätzlichen Optionen. Möchten Sie sich diese Optionen an einem Beispiel ansehen, böte sich <EMP_ADMIN> • CREATE an.

Seite senden

Wird die Schaltfläche genutzt, um die Seite abzusenden, stehen einige weitere Funktionen zur Verfügung, die festlegen, was in diesem Fall genau geschehen soll. Wir können uns diese Funktionalität auf <EMP_EDIT> ansehen, wo Schaltflächen für die Anlage, Aktualisierung oder das Löschen von Daten verwendet werden. Ich zeige die Optionen am Beispiel der Schaltfläche DELETE in Abbildung 6.25.

Die Attribute des Bereichs Verhalten werden immer angeboten, der Bereich Bestätigung hingegen nur, wenn der Schalter Verhalten • Bestätigung erforderlich angeschaltet wurde. Die Optionen sind (mittlerweile, früher war das anders) sehr intuitiv. Zunächst wird festgelegt, ob vor dem Absenden der Seite die Validierungen ausgeführt werden sollen, was bei CREATE und SAVE eingeschaltet ist, bei DELETE aber keinen Sinn ergibt (Sie könnten sonst Daten, die gegen die Validierungen verstoßen, nicht löschen). Ebenfalls wird bei DELETE nicht angewählt, dass eine Warnung bei nicht gespeicherten Änderungen ausgegeben werden soll. Dieser Mechanismus wird durch seinen Namen recht gut beschrieben und ergibt eigentlich nur bei einer CANCEL-Schaltfläche einen Sinn, da die anderen Schaltflächen die Daten ja verarbeiten.

Als Nächstes legt das Attribut Verhalten • Datenbankaktion fest, welche Aktivität den Daten auferlegt werden soll.

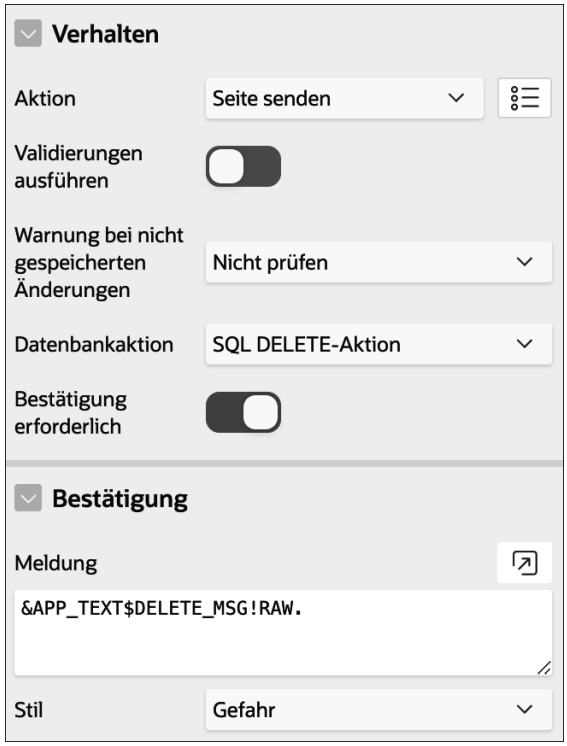


Abbildung 6.25 Attribute zum Senden der Seite

Wird schließlich der Schalter Bestätigung erforderlich gesetzt, hat dies zur Folge, dass vor dem Absenden der Seite ein Dialog angezeigt wird, der nachfragt, ob Sie wirklich wissen, was Sie tun. Sehr verwirrend ist, welche Meldung im Beispiel ausgegeben werden soll. Was bedeutet `&APP_TEXT$DELETE_MSG!RAW.`?

Die Zeichenfolge `&` und `.` (Punkt) schließt einen Ausdruck ein, der durch APEX als Zeichenkette ersetzt wird. Der Name des Ausdrucks zeigt, dass es sich bei der Ersetzungszeichenfolge um einen Anwendungstext mit dem Namen `DELETE_MSG` handelt. Diese Meldung ist von APEX vordefiniert, kann von Ihnen aber überschrieben werden, wenn Sie eine Meldung gleichen Namens in <Gemeinsame Komponenten> • Übersetzen • Textnachrichten anlegen. Dabei brauchen Sie meine Hilfe nicht, es werden nur der Meldungsname, die Sprache und der Meldungstext eingefügt. Beachten Sie lediglich, dass der Meldungsname nur `DELETE_MSG` lautet. Der abschließende Teil ist eine Syntax, um APEX mitzuteilen, ob der Text für bestimmte Zwecke formatiert werden soll, was durch die Angabe `RAW` abgeschaltet wird.

Abschließend existiert noch das Attribut Bestätigung • Stil, in dem Sie die Darstellung der Meldung an Ihre Bedürfnisse anpassen können.

Einen anderen Weg geht die Schaltfläche CANCEL, denn sie sendet die Seite nicht ab, sondern delegiert die Aktion an eine dynamische Aktion. Diese wiederum ist schnell erklärt, denn sie bricht das modale Dialogfeld ohne weiteres Federlesen ab.

6.4.3 Listen

Listen sind hierarchische Linksammlungen. Als Beispiel sehen wir uns die Navigationsmenü-Liste in <Gemeinsame Komponenten> • Listen • Navigationsmenü an. Klicken Sie auf den einzigen Eintrag des Berichts, und öffnen Sie anschließend den Eintrag Mitarbeiterbericht. Dieser Eintrag ist insofern typisch, als er einen übergeordneten Eintrag referenziert. Im Bereich Ziel sehen wir (derzeit noch?) eine optisch andere Version der Attribute mit etwas weniger Optionen.

Abbildung 6.26 Verwaltung des Links in einer Liste

Wir haben, wie in Abbildung 6.26 zu sehen ist, nur die Wahl zwischen einem Ziel in der aktuellen Anwendung oder einem URL. Das ist keine wirkliche Einschränkung, sondern eher unkomfortabel, denn ein Link auf eine Seite der aktuellen Anwendung ist letztlich auch nur ein URL. Wenn Sie den Bericht des Navigationsmenüs betrachten, sehen Sie das auch, die Einträge, die hier als Verweis auf eine Seite der aktuellen Anwendung dargestellt werden, sind im Bericht ein URL der Form f?p=&APP_ID.:3:&APP_SESSION.::&DEBUG.::::. Es liegt ein wenig außerhalb des Fokus dieses Abschnitts, alle

Details des URL zu erläutern, wir erkennen aber, dass die Ersetzungszeichenfolge `&APP_ID.` durch die aktuelle Anwendungs-ID und der Eintrag `&APP_SESSION.` durch die aktuelle Session-ID ersetzt werden wird.

Daher könnten wir hier auch eine alternative Form wählen, etwa `f?p=EMP:REPORT:&APP_SESSION.`. Auch dieses Format würde funktionieren und zeigt uns, wie wir auch andere Anwendungen erreichen können. Aber, wie gesagt, die konkrete Ausgestaltung und die Optionen dieser Notation sind ein wenig außerhalb des Fokus dieses Abschnitts. Zu den Low-Code-Möglichkeiten von Listen zählt zudem, dass wir die Einträge der Listen nicht nur manuell, sondern auch dynamisch über eine SQL-Abfrage ermitteln können. Daher komme ich auf diese Listen noch einmal zurück.

Ich gehe davon aus, dass sich die Listen in einem bald folgenden Release bezüglich der Möglichkeiten der Definition von Links den anderen Stellen angleichen werden.

6.4.4 Verzweigungen

Verzweigungen stellen Prozesse dar, die zu verschiedenen Zeitpunkten im Lebenszyklus einer Anwendungsseite in der Datenbank berechnet werden können. In Bezug auf die Definition eines Links bieten Verzweigungen keine Besonderheiten oder Erweiterungen zum allgemeinen Link, typisch ist lediglich, dass Verzweigung im Regelfall durch eine serverseitige Bedingung eingeschränkt werden und so kontrollieren, unter welchen Umständen welche Verzweigung ausgeführt wird.

Eine Information könnte von Belang sein: Falls eine Anwendungsseite erfordert, dass eine Anmeldung erfolgt ist, und noch keine gültige Session vorliegt, so hatten wir gesagt, dass in diesen Fällen automatisch zur `Login`-Seite verzweigt wird. Dies wird zwar durch eine Verzweigung erreicht, die in der Datenbank berechnet wird, allerdings wird diese nicht direkt eingerichtet, sondern ergibt sich durch die Festlegung im Seitenattribut SICHERHEIT • AUTHENTIFIZIERUNG • `Seite benötigt Authentifizierung`.

Es stellt sich die Frage, wo der Link hierfür definiert ist. Tatsächlich ist er ein wenig versteckt, und zwar in <ANWENDUNGSEIGENSCHAFTEN> • ATTRIBUTE • `Anmelde-URL` bzw. `Home-URL`. Diese beiden URLs zeigen einmal auf die Startseite und dann auf die Seite, die angesprochen werden soll, falls eine Anmeldung erforderlich ist.

6.5 APEX-Komponenten konditional anzeigen

Anwendungsseiten bestehen aus Regionen, Seitenelementen und Schaltflächen, wie wir gesehen haben. Daher bestimmen die verfügbaren Komponenten auf der Seite die Funktionalität der Anwendungsseite. Oft ist es jedoch so, dass die Funktionalität von den Berechtigungen des angemeldeten Anwenders abhängig sein sollte. So könnte es zum Beispiel einem »normalen« Anwender untersagt sein, Adressdaten

von Mitarbeitern zu ändern, während dies einem Supervisor oder Administrator möglich sein soll. Nun könnte man die Seite kopieren und alle Funktionen, die dem Bearbeiten der Adresse dienen, aus der Seite entfernen. Dann jedoch haben wir das eigentliche Problem nur verschoben, denn nun muss ja geklärt werden, warum ein Anwender die eine und eine andere Anwenderin die andere Seite aufrufen soll.

Was wir benötigen, ist ein Mechanismus, der Berechtigungen kontrollieren kann. Und dafür bietet APEX mehrere Mechanismen an:

- **Serverseitige Bedingungen**
  Wir haben diese Bedingungen bereits verwendet, und auch die Assistenten verwenden sie, zum Beispiel beim Ein- oder Ausblenden von Schaltflächen bei der Neuanlage oder dem Editieren von Daten.
- **Dynamische Aktionen**
  Mittels einer dynamischen Aktion können Seitenelemente ausgeblendet werden, wie wir das gemacht hatten, als wir das Eingabefeld für die Kommission eines Mitarbeiters von dessen Beruf abhängig gemacht haben.
- **Autorisierungsschemata**
  Dies ist ein Weg, den wir bislang noch nicht verwendet haben. Ein Autorisierungsschema zentralisiert die Prüfung, ob ein Anwender eine Berechtigung besitzt oder nicht, an einem zentralen Ort und macht sie anschließend deklarativ überall in APEX verfügbar.
- **Erstelloptionen**
  Auch die Erstelloptionen haben wir bislang nicht besprochen. Es ist ein Mechanismus, der eher grundlegend kontrolliert, ob eine Anwendungsseite zur Anwendung gehören soll oder nicht. Zweck ist die Verwaltung mehrerer Anwendungsversionen und das An- und Abschalten von Funktionalität.

Da wir serverseitige Bedingungen und dynamische Aktionen bereits kennengelernt haben, möchte ich zunächst die Autorisierungsschemata vorstellen, um anschließend zu diskutieren, wann welcher Mechanismus zum Einsatz kommen sollte.

6.5.1 Erstellung von Autorisierungsschemata

Ich halte Autorisierungsschemata für ein zentrales Feature einer APEX-Anwendung. Es ist ganz einfach zu erstellen und zu verwenden und besitzt gleichzeitig eine enorme Leistungsfähigkeit. Was also verbirgt sich hinter dem Konzept?

Ein Autorisierungsschema ist eine gemeinsame Komponente, die an einer zentralen Stelle prüft, ob der angemeldete Anwender über eine Berechtigung verfügt. Wie das Autorisierungsschema dies prüft, ist dabei völlig frei definierbar. Einige Wege sind durch APEX deklarativ unterstützt, andere benötigen weitergehende Programmie-

rung. Da wir uns im Bereich NO_CODE befinden, sehen wir uns Möglichkeiten an, die ohne Programmierung auskommen.

[!]

Merke

Ich hatte Sie in Kapitel 5, »Eine einfache Datenbankanwendung erstellen«, gebeten, das Feature Zugriffskontrolle zu installieren. Falls Sie dies nicht gemacht hatten, können Sie sich den letzten Stand der Anwendung installieren, dort ist das Feature vorhanden. Alternativ bespreche ich in Kapitel 7, wie man Features auch nachträglich installieren kann. Was ich im Folgenden besprechen möchte, ist jedoch nicht an die Installation dieses Features gebunden, durch das Feature sind lediglich bereits Daten angelegt worden, die wir nun verwenden. Sie können die Schritte auch nachstellen, indem Sie an den entsprechenden Stellen manuell die benötigten Informationen anlegen.

Der Workspace ist von mir so angelegt worden, dass bereits drei Benutzer vorhanden sind: BUCH_ADMIN, BUCH_ENTWICKLER und BUCH_ANWENDER. Alle haben das gleiche Standardpasswort und können daher verwendet werden, um die Berechtigungen auszuprobieren. Diesen Benutzern werden wir nun Rollen zuweisen. Doch: Wo finden wir diese Rollen?

Navigieren Sie zu <Gemeinsame Komponenten> • Sicherheit • Anwendungszugriffskontrolle.

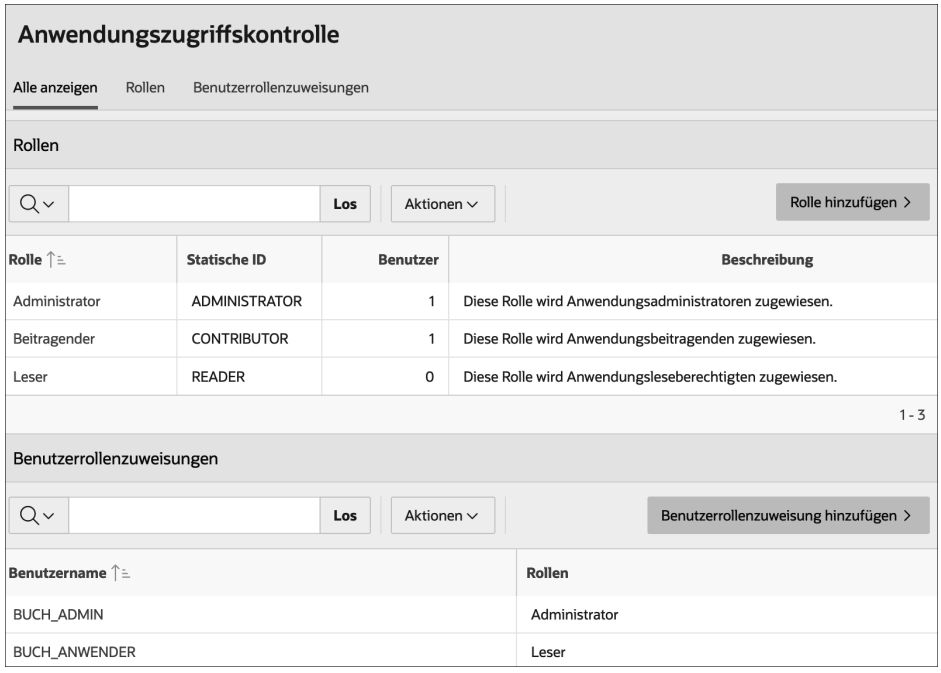


Abbildung 6.27 Anwendungsrollen des Features Zugriffskontrolle

Das Feature `Zugriffskontrolle` hat hier bereits drei Rollen angelegt. Haben Sie das Feature nicht installiert, können Sie die benötigten Rollen manuell hinzufügen. Die statische ID einer Rolle dient als technische ID und sollte daher nicht geändert werden, ansonsten sind keine weiteren Funktionen mit diesen Rollen verknüpft.

Diese Rollen müssen nun Anwendern zugeordnet werden. Dies können wir erreichen, indem wir zunächst im Regionsselektor oberhalb des Berichts alle Regionen anzeigen lassen oder direkt auf den Eintrag Benutzerrollenzuweisung klicken. Anschließend editieren wir die Zuweisung der Rollen durch Klick auf die Schaltfläche Benutzerrollenzuweisung hinzufügen.

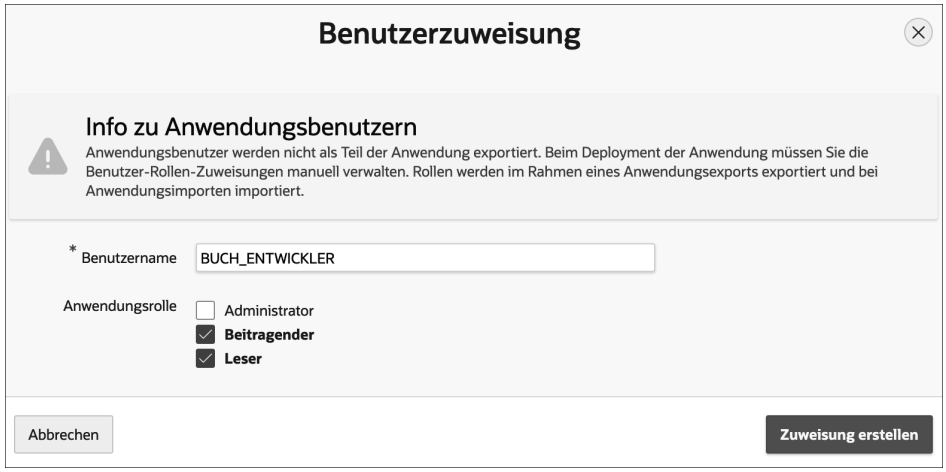


Abbildung 6.28 Zuweisung einer Rolle an einen Benutzer

Nun haben die Benutzer also unterschiedliche Berechtigungen. Bitte weisen Sie dem Benutzer `BUCH_ADMIN` die Rolle `Administrator` und dem Benutzer `BUCH_ANWENDER` die Rolle `Leser` zu. Der Benutzer `BUCH_ENTWICKLER` erhält die Rollen wie in Abbildung 6.28 gezeigt. Wie Sie dem Infotext in Abbildung 6.28 entnehmen können, werden die Zuweisungen von Rollenberechtigungen nicht exportiert, weil die Anwender nicht Teil der Anwendung, sondern des Workspaces sind. Daher kann ich diesen Teil nicht mit einem Export vorbereiten.

Wir haben also derzeit Anwendungsbenutzer, Rollen und eine Zuweisung der Rollen zu den Anwendern. Nun fehlt noch eine Komponente, die uns sagt, ob ein Anwender eine Rolle zugewiesen bekommen hat oder nicht. Dies ist das Autorisierungsschema. Das Feature Zugriffskontrolle hat auch hier bereits einige Daten angelegt, die wir ebenfalls editieren oder erweitern können. Navigieren Sie zu <Gemeinsame Komponenten> • Sicherheit • Autorisierungsschemas (siehe Abbildung 6.29).

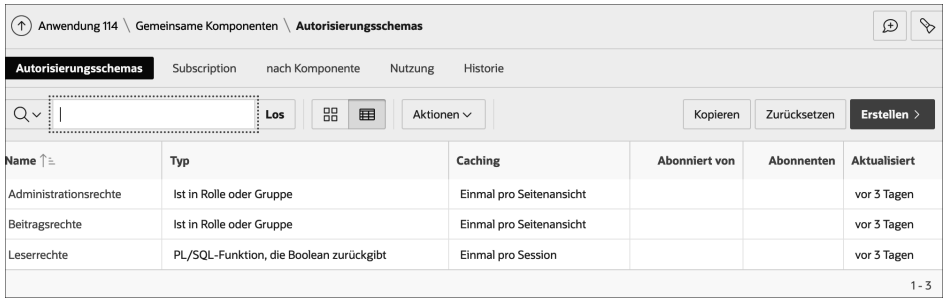


Abbildung 6.29 Autorisierungsschemata des Features Zugriffskontrolle

Durch die Vorarbeit der Rollen und die Zuweisung an Workspace-Benutzer haben wir nun einen Weg eingerichtet, um ohne Programmierung ein Autorisierungsschema zu verwenden. Sehen Sie sich ein Beispiel an. Falls Sie das Feature Zugriffskontrolle nicht installiert haben, erstellen Sie einfach ein neues Autorisierungsschema und übernehmen die Einstellungen wie in Abbildung 6.30.

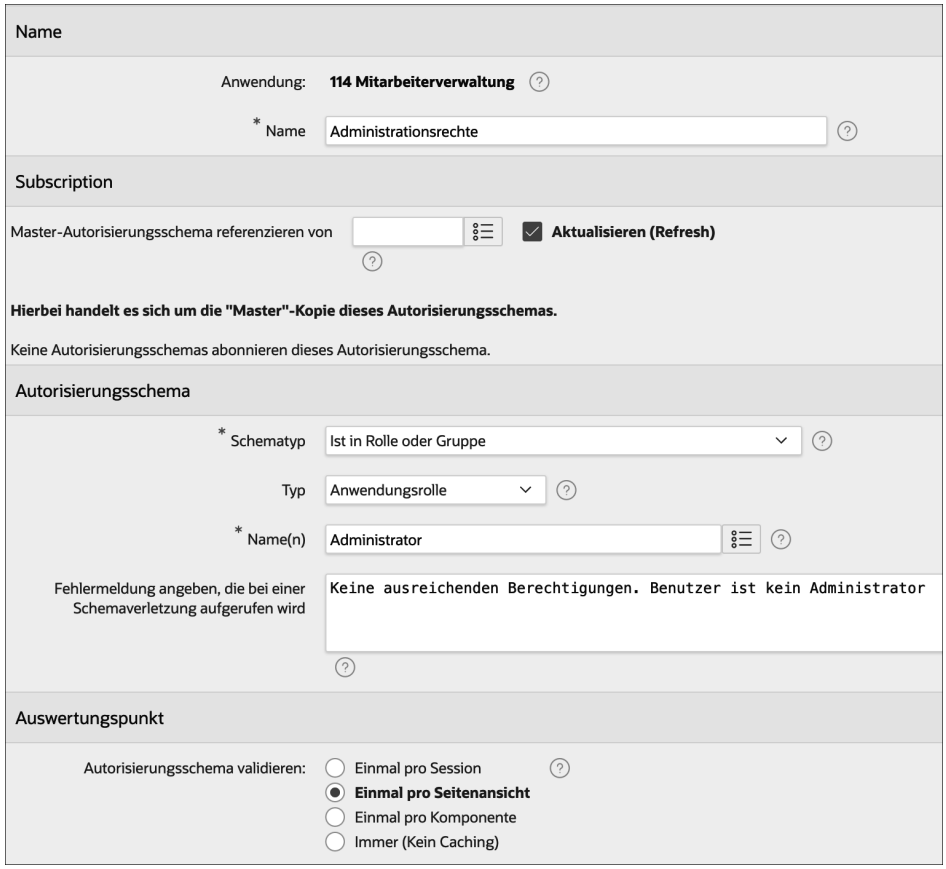


Abbildung 6.30 Autorisierungsschema für die Administratorprüfung

Da die Prüfung bereits im Attribut Autorisierungsschema • Schematyp • `Ist in Rolle oder Gruppe` vordefiniert ist, benötigen wir keine Programmierung. Zudem können wir den Eintrag Autorisierungsschema • Name(n) über die Symbolschaltfläche rechts neben dem Eingabefeld nachschlagen.

6.5.2 Verwendung von Autorisierungsschemata

Nun haben wir ein Autorisierungsschema, nun fehlt noch die Verwendung. Wie Autorisierungsschemata verwendet werden, halte ich für eine der hervorstechenden Eigenschaften von APEX, denn es ist möglich, beinahe jede Komponente deklarativ an ein Autorisierungsschema zu binden. Sehen wir uns das zum Beispiel für eine Anwendungsseite an. Die modale Formularseite zum Bearbeiten eines Mitarbeiters soll nur Administratoren möglich sein. Um dies zu erreichen, gehen wir zu <EMP_EDIT> • Sicherheit • Autorisierungsschema und wählen den Eintrag `Administrationsrechte`.

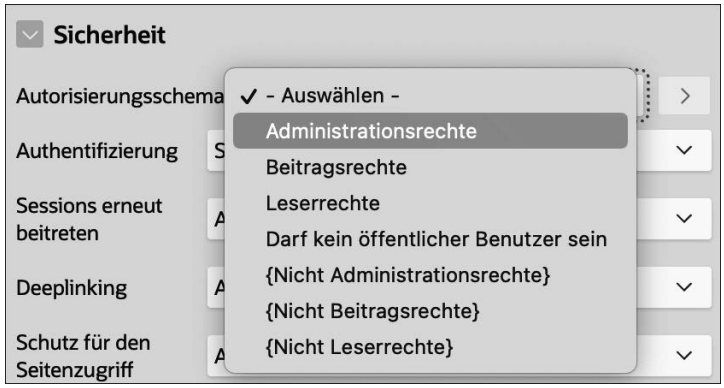


Abbildung 6.31 Schutz einer Anwendungsseite durch ein Autorisierungsschema

Wenn wir uns nun als Benutzer BUCH_ADMIN anmelden, funktioniert der Aufruf ohne Probleme. Falls wir uns jedoch ab- und als Benutzer BUCH_ENTWICKLER anmelden und auf diese Seite zugreifen möchten, geht das schief, wie Abbildung 6.32 zeigt.

Nun gilt es nicht als höflich, den Anwendern solche Fehlermeldungen zu präsentieren, doch wollte ich zeigen, dass die Bindung der Seite an ein Autorisierungsschema nun ausreicht, um einen Schutz der Seite sicherzustellen. Doch die wahre Kraft liegt darin, dass beinahe überall Autorisierungsschemata eingesetzt werden können. Das würde ich gern an einem Beispiel demonstrieren. Lassen Sie uns zunächst die Fehlermeldung aus Abbildung 6.32 vermeiden, indem wir den Bearbeitungslink des Mitarbeiterberichts nur anzeigen, wenn der angemeldete Benutzer ein Administrator ist. Gehen Sie hierfür zu <EMP_ADMIN> • Mitarbeiterübersicht • Spalten • `APEX$LINK`.

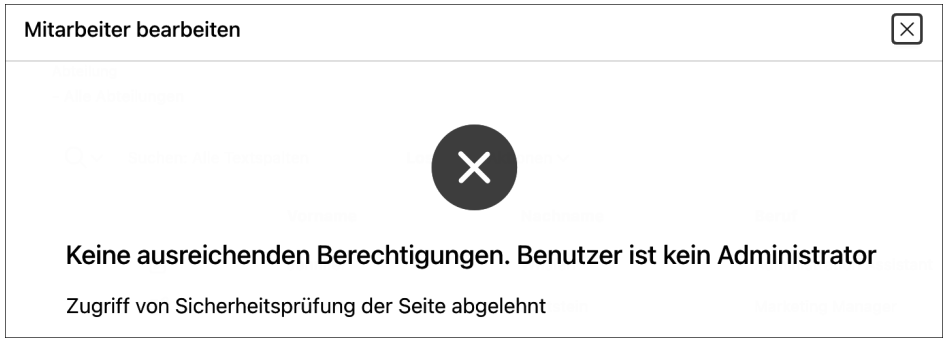


Abbildung 6.32 Fehlermeldung beim Aufruf der Seite

Diese Spalte verfügt wiederum über das Attribut SICHERHEIT • AUTORISIERUNGSSCHEMA, das Sie nun entsprechend einstellen können. Testen Sie das Ergebnis: Der Mitarbeiterbericht enthält nun keinen Link zur Bearbeitung eines Mitarbeiters mehr. Stellen Sie die gleiche Bedingung auch für die Schaltfläche `CREATE` auf derselben Seite ein und testen Sie.

Als zweites Beispiel werden wir die Berechtigung, die Seite `Mitarbeiter verwalten` aufzurufen, an die Berechtigung `Beitragsrechte` binden. Wir werden also die Seite an dieses Autorisierungsschema binden und müssen zudem den Eintrag im Navigationsmenü entsprechend schützen. Den ersten Punkt finden Sie selbst, der Eintrag im Navigationsmenü verbirgt sich in <GEMEINSAME KOMPONENTEN> • NAVIGATION • NAVIGATIONSMENÜ. Öffnen Sie die Liste und anschließend den Eintrag MITARBEITER VERWALTEN. Hier finden Sie das Autorisierungsschema im Attribut AUTORISIERUNG • AUTORISIERUNGSSCHEMA.

Wie Sie sehen, ist es sehr einfach, Anwendungselemente an ein Rollenkonzept zu binden. Die gesamte Funktionalität hat keinerlei Programmierung erfordert und erlaubt dennoch eine leistungsfähige und feingranulare Steuerung des Zugriffs.

[!]

Merke

Es scheint ausreichend, den Schutz einer Seite nur dadurch sicherzustellen, dass zum Beispiel der Bearbeitungslink und die Schaltfläche zum Erstellen eines neuen Mitarbeiters durch ein Autorisierungsschema geschützt werden. Das ist aber nicht richtig. Täten Sie nur dies, könnte ein Anwender durch eine einfache Manipulation des URL dennoch auf die Anwendungsseite verzweigen und Änderungen vornehmen. Der Link und die Schaltfläche sind ja nur Navigationshilfen, sie verhindern nicht, dass auf anderem Weg diese Seite aufgerufen werden kann.

6.5.3 Erstelloptionen

Eine Erstelloption ist lediglich ein Schalter mit einem Namen, der anzeigt, ob eine Komponente aktuell angezeigt werden soll oder nicht. Insofern ist die Erstelloption vergleichbar mit einer Autorisierungsprüfung, allerdings ist diese nicht von Anwendungslogik abhängig, sondern fest eingestellt. Diese Erstelloptionen können anschließend an beinahe beliebige Komponenten gebunden werden.

Da Sie auch eigene Erstelloptionen erstellen können, könnten Sie zum Beispiel eine Option `Katalogbestellung` erstellen und an alle Komponenten binden, die in Ihrer Anwendung für die Bestellung eines Kataloges verwendet werden. Solange Sie noch nicht fertig sind, könnten Sie die Erstelloption abschalten und somit den Aufruf der Komponenten unterbinden. Passt alles, schalten Sie die Erstellkomponente an, und alle referenzierten Komponenten werden ausgeführt.

Das APEX-Team gibt neben diesem Szenario auch noch das Gegenteil als Beispiel: Sie möchten Funktionalität entfernen. Dann können Sie diese erst einmal durch eine Erstelloption deaktivieren. Stellt sich heraus, dass wirklich niemand mehr die Funktionalität benötigt, löschen Sie die verbundenen Komponenten.

Durch eine Erstelloption werden mehrere Ziele erreicht:

- Sie können Funktionalität gruppieren und sehen, welche Komponenten dafür benötigt werden.
- Sie können Funktionalität an- und abschalten.
- Sie können Funktionalität löschen, denn es existiert eine entsprechende Option in den Erstelloptionen, um alle verbundenen Komponenten zu löschen.

Die Erstelloption binden Sie an eine Komponente, indem Sie in den Komponentenattributen im Bereich KONFIGURATION • ERSTELLOPTION die entsprechende Option auswählen. Ähnlich der Autorisierung einer Komponente ist auch dieses Attribut in beinahe allen Komponenten vorhanden.

6.5.4 Vergleich der Strategien

Nachdem wir nun die Autorisierungsschemata kennengelernt haben, stellt sich die Frage, welche Strategie verwendet werden sollte, um APEX-Anwendungen flexibel auf die Berechtigungen des Anwenders reagieren zu lassen. Die Antwort ist, dass alle drei Strategien erforderlich sind, es gibt aber eine klare Rangfolge, die Sie einhalten sollten.

Die erste Ebene stellen die Autorisierungsschemata dar. Der große Vorteil besteht in der zentralen Definition, die sie auf allen Anwendungsseiten und auch innerhalb von gemeinsamen Komponenten deklarativ nutzbar macht. Die deklarative Unterstüt-

zung ist ein weiteres großes Plus. Daher sollten Sie, wann immer möglich, dieser Strategie den Vorzug geben.

Das Problem der Autorisierungsschemata ist allerdings, dass Sie sich nicht flexibel auf die konkreten Bedingungen einer Anwendungsseite einstellen können. Sie können also zum Beispiel nicht auf den aktuellen Inhalt eines Seitenelements reagieren. Dies ist jedoch oftmals erforderlich, zum Beispiel um zu entscheiden, ob eine Schaltfläche angezeigt werden soll oder nicht. Wenn Sie sich den modalen Dialog in Erinnerung rufen, war hier eine Entscheidung erforderlich, ob die Schaltfläche `CREATE` oder die Schaltflächen `SAVE` und `DELETE` angezeigt werden sollen. Grundlage der Entscheidung war, ob das Seitenelement `P5_EMP_ID` einen Primärschlüssel enthält oder nicht. Dies könnte vielleicht sogar durch ein Autorisierungsschema geprüft werden, konterkariert aber die Idee, denn dadurch ist das Autorisierungsschema nur noch auf Seite 5 und nur für diese konkrete Prüfung einsetzbar.

Das ist nicht sinnvoll, daher werden solche Prüfungen durch die zweite Ebene, die serverseitigen Prüfungen, durchgeführt. Diese erheben nicht den Anspruch, auf allen Seiten wiederverwendbar zu sein, sondern sind für ein konkretes Element auf der Anwendungsseite zu formulieren. Diese mangelnde Wiederverwendbarkeit ist aber kein Problem, da die Bedingung ja auch konkret für die Seite geprüft werden muss. Eine große Anzahl deklarativ vorhandener Prüfungen erleichtert zudem die schnelle Erstellung und Verwendung.

Nachteil der serverseitigen Prüfung ist, dass diese nur einmal, beim Rendern der Anwendungsseite, ausgeführt wird. Im Fall der Prüfung für die Schaltflächen ist das vollständig ausreichend, denn entweder wird ein Mitarbeiter bearbeitet oder eben neu angelegt, das ist bekannt, bevor die Seite gerendert wird. Im Gegenteil ist die Tatsache, dass diese Prüfung hier innerhalb der Datenbank vorgenommen wird, ein sehr großer Vorteil, denn eine Schaltfläche, die aufgrund einer serverseitigen Bedingung nicht dargestellt werden soll, wird auf der entstehenden Seite nicht einfach ausgeblendet, sondern erst gar nicht eingefügt, es gibt diese Schaltfläche auf der Seite nicht.

Es gibt aber Situationen, in denen wir flexibel auf die Benutzereingaben reagieren möchten, wie wir das am Beispiel des Eingabefeldes `P5_EMP_COMMISSION_PCT` gesehen hatten. Dieses Feld ist gebunden an gewisse Werte der Auswahlliste `P5_EMP_JOB_ID`, und dieses Auswahlfeld kann durch den Anwender geändert werden. Daher muss das Eingabefeld `P5_EMP_COMMISSION_PCT` auf der Anwendungsseite vorhanden sein, selbst wenn der gewählte Mitarbeiter zurzeit einen Beruf ausübt, der keine Kommissionsberechtigung umfasst – denn wenn wir den Beruf entsprechend ändern, muss etwas auf der Seite vorhanden sein, was wir in diesem Fall einblenden können.

Sie könnten diese Prüfung auch serverseitig machen, dann aber zum Preis, dass die Anwendungsseite komplett neu gerendert werden muss. Die meisten Entwickler hal-

ten dies nicht für akzeptabel (auch ich gehöre dazu), sodass für diese Einsatzszenarien eine dynamische Aktion die einzig sinnvolle Alternative darstellt.

Eine eigene Rolle spielen die Erstelloptionen. Richtig genutzt, können sie ein mächtiges Mittel sein, um an einer zentralen Stelle Funktionalität an- und abschalten zu können. Ob sich dies jedoch in Ihrer konkreten Anwendung so sauber separieren lässt und ob Sie die Disziplin aufbringen, alle entsprechenden Funktionsbereiche durch Erstelloptionen abzusichern, ist für mich eine andere Frage. Sie haben ihre Bedeutung in Zusammenhang mit den Features, wo sie sehr gut passen, weil ein Feature eine abgeschottete Funktionalität für sich ist. Ob es sich auch für andere Funktionalitäten eignet, müssen Sie im Laufe der Zeit selbst erspüren.

Wichtig für die richtige Wahl der Mittel ist es, zu versuchen, mit einem Autorisierungsschema zu starten und dies nur dann durch eine serverseitige Prüfung zu ersetzen, wenn es nicht anders geht. Die serverseitige Prüfung wird wiederum nur dann durch eine dynamische Aktion ersetzt, wenn es anders nicht geht oder nicht tolerabel wäre. Die Reihenfolge der Entscheidungen ist hierbei die wichtigste Information. Versuchen Sie nicht, hier kreativ zu werden. Das Problem besteht darin, dass Kollegen (oder auch Sie selbst nach einiger Zeit) Probleme haben werden, den Fluss der Anwendung zu verstehen, wenn Entscheidungen nicht dort getroffen werden, wo man sie erwarten würde. Die vorgestellten Mittel sind für den hier beschriebenen Einsatz gedacht und angelegt und sollten daher auch so genutzt werden.