PART I

Access Web Application Design and Development

1

Introduction to Access Web Apps and Architecture

WHAT'S IN THIS CHAPTER?

- Defining an app
- Examining how apps are structured
- Identifying methods for distributing apps
- Listing deprecated components and new tools

With the release of Office 2013, Microsoft introduced major changes to the Office architecture, and Access is no exception. To make Office more accessible to an increasing number and variety of users, devices, and platforms, Microsoft continues to expand, provisioning Office "in the cloud" with an emphasis on making a seamless transition when running on different devices. One major frustration with traditional hosted environments has been that you had to accept everything out of the box; custom applications or code might not run in a hosted environment or might run only in a restricted mode. New to SharePoint 2013 and Office 2013, the *Cloud App Model* (subsequently referred to as *app* in this book) is Microsoft's answer to the longstanding problem of enabling people to write and create custom solutions that can run in the cloud without the usual complexities that come with a client installation. With the new model, we now can create a new app, which is essentially a web application that has been prepared and packaged in a specific way. Apps can take many different forms, but essentially they are all web pages containing custom code and content that is integrated into Office, typically using Office 365. Access 2013 can create an app that can be used in the cloud. In this chapter, you'll learn how the new app architecture can allow you to distribute, deploy, and integrate custom solutions.

First, you will quickly review what has changed in Access 2013, starting with features that have been deprecated. Then you'll move into the anatomy of an app, and wrap up with a brief look at how an Access web app functions.

DEPRECATED COMPONENTS

As with every new release, one of your first tasks is to review the possible impacts on existing Access solutions and tools if they are migrated to or integrated into environments using the new version. Features that depend on deprecated components will typically need to be modified in order for the application to work using Access 2013. The following discussion identifies the components that were deprecated for Access 2013 and offers suggestions to accommodate the changes.

With the vast quantity of legacy files being used for storing data, it can be helpful to know what features are deprecated or compatible with each new release. The `.accdb` file format was introduced with Access 2007 so the lists will start with that release. The following lists are for your convenience and to assist with troubleshooting and converting files.

The following features are no longer available, as of Access 2007:

- *Designing* Data Access Pages (DAPs)
- Microsoft Office XP Web Components
- Replication
- User-Level Security and Workgroup Administrator
- The UI for import and export of older file formats

The following features are no longer available as of Access 2010:

- *Opening* Data Access Pages (DAPs)
- Snapshot format for report output
- Calendar control (`mscal.ocx`)
- ISAM support, including Paradox, Lotus 1-2-3, and Jet 2.x or older
- Replication Conflict Viewer

The following features are no longer available as of Access 2013, and each is discussed further in this section:

- Access Data Projects (`.adp`)
- Jet Replication
- Menus and Toolbars
- Import/Export/Link to Jet 3.x and dBASE files

- PivotTables and PivotCharts
- Collect Data via E-mail
- SharePoint Workflow
- Source Code Control Extension
- Packaging Wizard
- Upsizing Wizard
- Creating Access Web Databases

Access Data Projects

Introduced in Access 2000, the Access Data Project (`.adp`) file format allowed for Access solutions to be built with SQL Server, bypassing the Jet engine entirely. The `.adp` file format also enabled you to create and edit SQL Server objects (for example, tables, views, and stored procedures) within Access. Unfortunately, the object designers were version-dependent; if the SQL Server instance to which the `.adp` connected was upgraded to a later version, Access Designer for those objects would no longer function and it was necessary to use T-SQL to edit those objects. Moreover, `.adp` uses OLEDB and ADO to connect with SQL Server, and SQL Server 2012 is the first version to start deprecating OLEDB as a first-class provider. Microsoft now recommends ODBC instead of OLEDB. Since the release of Access 2007, Microsoft recommends creating new Access solutions using the `.accdb` file format rather than the `.adp` file format. In Access 2010, there was no quick button to create a new `.adp` file format. While we believe that a SQL Server back end for an Access solution is a wonderful combination, the combination is best delivered via the linked table method made available in both `.mdb` and `.accdb` file formats. It's better to design the Access solution with server-client architecture from the start. In the second half of this book, we will look into this architecture more deeply. Furthermore, the new Access web apps, which we will discuss, are now based on SQL Server; so in a sense, Access web apps can be said to be a more modern replacement for the `.adp` file format. We will also look at how this is achieved in the first half of the book.

For those wanting to migrate an `.adp` file, we recommend that you start from scratch in an `.accdb` file format; import the forms, reports, and modules; and then create linked tables referencing the tables and views in SQL Server. Refitting the code will be necessary to get the forms and reports to work correctly in the `.accdb` file format. Keep in mind that you can continue to use ADO as the connection technology in `.accdb`, but because forms default to DAO, it will be necessary to write the code needed to bind a form to an ADO recordset. For reports, passthrough queries will be the recommended method for binding a report to a SQL Server object output. Modules require review and may require modifications to verify that the code will function correctly, especially for those depending on an `.adp`-specific context such as the `CurrentProject.Connection` object.

Jet Replication

Since the introduction of the `.accdb` file format in Access 2007, Microsoft has been recommending against creating new Access solutions based on Jet Replication. The new `.accdb` file format did not support Jet Replication. However, Access 2007 and Access 2010 did continue to support Access solutions that used the `.mdb` file format and Jet Replication. Access 2013 ends this support entirely. Jet Replication is a technology that allows for offline synchronization of data and was more relevant when Internet availability and speed were not as good as today. However, the effort required to manage a replicated Access database usually made it quite daunting and required considerable specialized knowledge. Furthermore, with the new `.accdb` file format and SharePoint integration, working offline using SharePoint lists is much simpler and requires none of the administrative work that Jet Replication required. Thus, if you have a solution that uses Jet Replication, we recommend that you look at SharePoint lists as replacements for providing offline synchronization of the data, and remove the replication from the Access file prior to upgrading to Access 2013.

Menus and Toolbars

Since Office 2007, Microsoft has moved away from menus and toolbars and has largely replaced them with ribbons. In prior versions of Access, it was possible to create custom menus and toolbars that could then be added to an Access solution. Access 2007 and 2010 continued to support the use of those custom menus and toolbars by displaying them in the ribbon's Add-Ins tab. Alternatively, by hiding the ribbon, you could get the original menu back. However, it was awkward and not without problems.

Access 2013 no longer supports displaying menus and toolbars from older versions, though you can continue to import the legacy menus and toolbars with the understanding that they can only be used via the ribbon's Add-Ins tab. You can continue to use shortcut menus as always. For those who rely on custom menus and toolbars, here are a few recommendations:

- Customize the ribbon.
- Create macros to create your shortcut menus and/or create the shortcut menus programmatically. Refer to *Ribbon X: Customizing the Office 2007 Ribbon* (Wiley Publishing, Inc., 2008) for the details the details on creating a shortcut menu.
- Create controls on your forms to provide the needed navigation in lieu of the ribbon.

Any of these recommendations will work and, if appropriate, may be mixed and matched. Only you can decide what is best for migrating your Access solutions with custom menus and toolbars. Figure 1-1 shows the Customize Ribbon pane in Access 2013.

Since Access 2010, creating custom ribbons has been much easier because there are two panes: Customize Ribbon pane and Customize Quick Access Toolbar pane; in the Access Options, which also supports importing and exporting of the customization. You can use the import/export functionality to quickly build the ribbon as you need it, and then either fine-tune the resulting XML or put it into the ribbon table so that at distribution your users get the ribbon just as you specified it.

FIGURE 1-1: Customize the Ribbon pane

Import/Export/Link to Jet 3.x and dBASE Files

In Access 2010, we saw deprecation for support of Jet 2.x (Access 2.0 `.mdb` file format), Lotus 1-2-3, and Paradox files. Those were removed as those formats were quite old and rarely used. Likewise, use of Jet 3.x files (Access 95 and 97 `.mdb` file format) and dBASE has dwindled and thus does not merit continued support. As was the case in Access 2010, if you have files of those types, we recommend that you upgrade them using a previous version of Access that still supports importing/exporting from those file formats. This will enable you to complete the upgrade to Access 2013. If you do not have a previous version of Access available to you, the alternative is to export the source file using the original program into another file format, such as text file or Excel spreadsheet, which Access 2013 still supports for import and/or linking operations.

> **NOTE** *For cases where you need to use an* `.mdb` *file based on Jet 3.x or earlier and do not need to use Access, an option to consider is to use Jet 4.0, which is usually already installed with the Windows operating system, to open the database. You could then use DAO automation and a private* `DBEngine` *object to upgrade the file or access the data.*

PivotTables and PivotCharts

In previous versions, Access forms could be configured to show data in PivotTable and/or PivotChart view, which was effective for displaying multi-dimensional data or for rendering aggregated data in a visual manner. Both relied on Microsoft Office Web Components, which has already been deprecated in previous versions of Office. Excel has since refined and enhanced its PivotTable/PivotChart capabilities, but those enhancements aren't available to Access directly.

To upgrade Access solutions that make use of PivotTables or PivotCharts, you have two possible approaches:

1. Add an ActiveX control referencing "Microsoft Office XX.0 PivotTable" and/or "Microsoft Office XX.0 Chart" where XX.0 may be either 10.0 or 11.0, and update all VBA references to PivotTable/PivotChart events and properties from the form to the added ActiveX control.
2. Instead of rendering the data in Access, automate Excel to build a PivotTable and/or PivotChart and use Office Data connection to enable Excel to query the data directly from the source table/queries.

Collect Data via E-mail

Introduced in Access 2007, this feature made it possible to build an e-mail template based on a table, send e-mail requests out to recipients, process their replies via Outlook, and convert the replies into rows in the table. While it seemed like a good idea when it was introduced, this feature required that the e-mail be formatted as HTML, or created as an InfoPath form. Also, it generally worked best with Microsoft Outlook. These limitations made it impractical for situations where you had no control over whether the e-mail would be viewed as HTML or text, or over which e-mail client would be used.

If your Access solution requires input from external users, we generally recommend that you make use of a service that can provide web pages for filling out and submitting data to you. One good example of such a service is `www.surveymonkey.com`. This service is more accessible in the sense that you need only send people a link to the survey URL. They can use their favorite web browser to fill in their responses, and you can extract the data out of a downloaded Excel spreadsheet.

Furthermore, there is no guarantee that you need only a single table to store the responses; normalization may require multiple tables for a single "survey." For this reason, you would still have to transform the responses, whether you were using the Collect Data via Email feature or a linked Excel spreadsheet.

SharePoint Workflow

In Access 2007, as part of enhanced integration with SharePoint, it was also possible for Access to initiate a SharePoint workflow that was associated with a SharePoint list. Figure 1-2 demonstrates how to start a new workflow in Access 2010.

However, adoption of workflows was not easy because the programmability was quite limited, required that you work in SharePoint or use associated tools such as SharePoint Designer rather than Access, and called for considerable knowledge of how SharePoint Workflow works.

Furthermore, there were no events to facilitate scripting actions when a workflow was started, progressed, and finished. For those reasons, the adoption of Workflow has been slow.

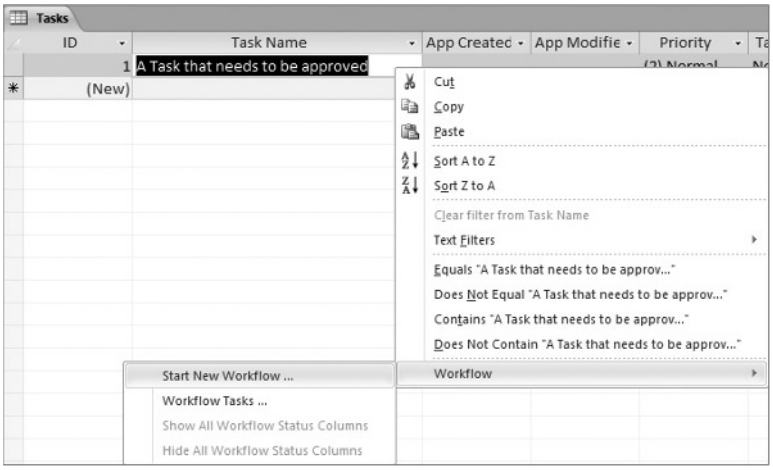

FIGURE 1-2: Starting a new workflow in Access 2010

For those who use workflows, better solutions can be discovered via two possible workarounds:

1. Use forms and VBA to perform the work that was done by the workflow. VBA is fully capable of performing automation (for example, generating a report, creating and sending an e-mail) and is more flexible than SharePoint Workflow, which is scripted using a flowchart.
2. Use table event macros to perform a custom action based on insert/update/deletion of data.

Source Code Control Extension

Access has long supported the Microsoft Source Code Control Interface (MSSCCI), which allowed Access to work with source code control client applications such as Visual SourceSafe, Team Foundation Server, and several other third-party software packages, providing control over the source code in your databases. The source control model has not changed much. It is based on a locking model where one must first obtain an exclusive lock on an Access object before editing it. Because Access Form objects and Report objects are technically binary objects, it is not possible for two or more developers to edit the same Form objects or Report objects concurrently. Only VBA modules can be edited concurrently. The source code control model has since evolved to support a merging model as well as decentralized repositories.

If you want to migrate to Access 2013 and continue to use source code control, then you should adopt an approach that involves manually exporting and importing Form and Report objects as text using the undocumented `SaveAsText` and `LoadFromText` methods. You would then perform the update/commit activities in the filesystem rather than within Access.

Alternatively, you could explore using plugins that do not rely on MSSCCI and automate the `SaveAsText`/`LoadFromText` process for you. OASIS-SVN, a commercial product is one example of such a plugin that can be used in Access 2013.

Packaging Wizard

Since Access 2007, it has been easier to create an installer to distribute easy installs of custom Access solutions to users. The Packaging Wizard can be used to create templates describing how to build a Windows installer that incorporates the source Access files, additional files needed for the application, and any registry entries that needed to be set. Furthermore, the Packaging Wizard can create a bootstrapper to determine whether the Access Runtime is installed on the target computer, and if not, to download it from the web.

In theory, this seemed like a godsend, but in practice, complex Access solutions were usually the ones that needed an installer. More often than not, a solution would require installation of additional components, something that the Packaging Wizard did not help with. Thus, it was still necessary to obtain a third-party packaging software application. Choices ranged from freeware, such as Inno Setup, to high-end InstallShield, to developer-centric Wix.

BOOTSTRAPPER AND SETUP.EXE

When installing software, people instinctively know to go for a `setup.exe` file. However, they may not realize they are actually running a bootstrapper. Windows Installer (aka `.msi` files) is not able to determine the prerequisites, download the needed files, verify privileges, and so forth. For this reason, a small executable is needed to "bootstrap" the install by performing those tasks before executing the `.msi` files to perform the actual installation.

Upsizing Wizard

In earlier versions of Access, Microsoft developed the Upsizing Wizard to make it easier to transform Access tables into SQL Server tables. This provided an easy-to-use GUI for selecting tables and specifying additional options that should be performed when the tables were upsized. However, the Upsizing Wizard has not been updated for several versions. In the interim, a new tool, called SQL Server Migration Assistant for Access (SSMA), was introduced and has matured into a powerful tool for not simply uploading, but also transforming several Access-specific options such as the Allow Zero Length property, into their SQL Server analogues. Furthermore, SSMA also makes intelligent recommendations on what changes may be needed, such as adding a `rowversion` column to tables that may benefit from having one.

ROWVERSION BY ANY OTHER NAME...

In earlier versions of SQL Server, the data type was called `timestamp` but that term is quite confusing because it does not reference an actual timestamp but rather identifies when a version of the row was updated relative to the creation of the database. Since SQL Server 2005, this data type has been called `rowversion`, which accurately describes its functionality. Unfortunately, even into SQL Server 2012, SQL Server Management Studio's designer and scripting continue to use the older term `timestamp` even though Microsoft has stated that it is deprecated and will be removed in future versions of SQL Server.

Because SSMA is a far more modern solution, there is no reason to continue to use the Upsizing Wizard. SSMA can be downloaded for free from the Microsoft download center. Alternatively, as you will discover later in this book, Access web apps make it very easy to import Access data into SQL Server tables without the configuration steps that either SSMA or the Uploading Wizard required of you.

Creating Access Web Databases

Access 2010 introduced us to Access web databases. Web databases are still supported in Access 2013 and can be opened and edited just like in 2010. However, to encourage people to use the new app architecture, Microsoft does not provide a way to create a new web database in Access 2013. As Figure 1-3 shows, you can work around this limitation by creating a new web database via a SharePoint site, but we feel that you'll find that the new format in Access 2013 has more to offer than Access 2010 web databases did. Note that the availability is contingent on whether the SharePoint farm has the Access 2010 Services running.

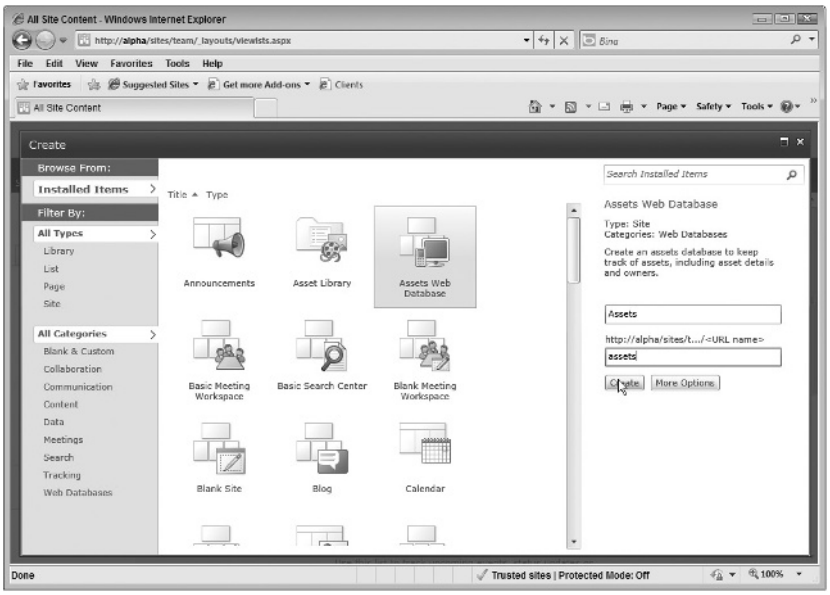

FIGURE 1-3: Creating a 2010-style web database in SharePoint 2013

NEW COMPONENTS ADDED

While the first part of this book will focus on Access web apps, we want to point out new additions to Access that are also relevant to Access solutions migrating from previous Access versions. Microsoft has emphasized strengthening the manageability of both Access databases and Excel spreadsheets. Because these are files, they have a tendency to proliferate. An organization can easily find itself awash in hundreds, if not thousands, of Excel and Access files, and not just different files but also multiple versions of the same file. Even if you don't suffer from file management challenges, the new 2013 tools may prove beneficial for managing your Access solutions. Two are most relevant to Access: the Database Compare tool and the Audit and Control Management Server.

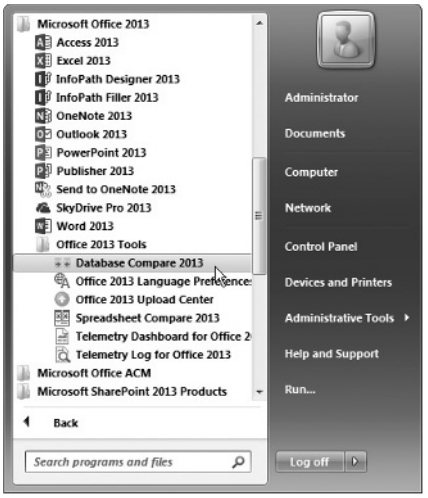

FIGURE 1-4: Database Compare

Database Compare

To help address the common problem of identifying what has changed in two similar Access desktop files, Microsoft has introduced Database Compare, as shown in Figure 1-4.

It is common to copy Access solutions as part of distribution; however, opening an Access file immediately changes its last modified timestamp. This can lead to confusion when trying to identify the latest version. To alleviate this problem, we can now turn to the Database Compare tool, which allows us to compare two Access files. In Figure 1-5, you can choose which type of objects to compare and the desired level of granularity.

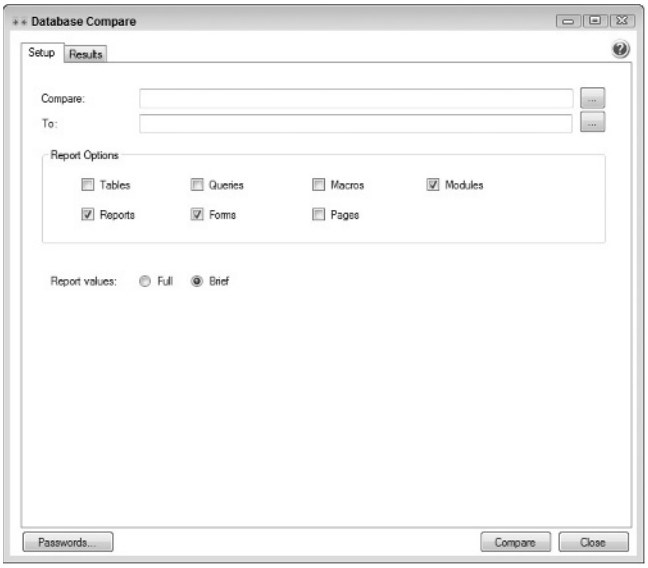

FIGURE 1-5: Selection screen

Once the comparer has finished the analysis, you get a report listing all differences. With this, you can quickly zero in on the changes and ensure that all desired updates are included in your final selection.

Audit and Control Management Server

For organizations with large numbers of Access and/or Excel solutions, Audit and Control Management (ACM) Server can be very useful in tracking the usage of Access and Excel solutions organization-wide, and Figure 1-6 provides an idea of what it can do.

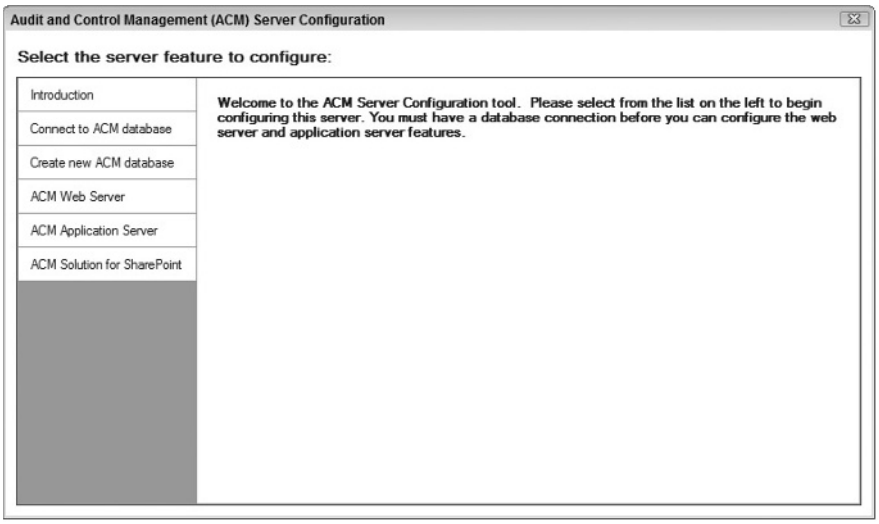

FIGURE 1-6: Audit and Control Management Server

ACM Server will provide data on when the file was last used and how often it was used, which can be useful in supporting decisions about organizing the files and whether that activity requires additional support. ACM Server is available on certain Office 365 plans and Office 2013 Professional Plus.

WHAT IS AN APP?

Having reviewed the deprecated features and new components added to Access 2013, you are now ready to explore the underlying foundation that enables you to use Access in a web browser. Understanding the foundation will be beneficial in assessing and identifying appropriate solutions for deploying and distributing Access web apps.

As noted at the start of the chapter, a longstanding problem with a hosted environment is that customization is inherently limited or even outright banned. This is a regrettable, but logical, conclusion when we consider that the overriding requirement is maintaining a stable hosted environment that won't go down when someone introduces a custom solution that doesn't work as expected.

But this is not the only problem the app architecture is intended to solve. With mobile devices becoming more widespread, it can be a daunting task to create an application that can be consumed on a wide variety of hardware, operating systems, and software.

Many mobile devices come with their own specialized repository, commonly referred to as an *app store*, for distributing, downloading, or purchasing applications. The advantages of such repositories are numerous. For one, consumers can have some level of assurance that the application listed in the repository has at least been reviewed and deemed safe for use. For another, the process of buying and acquiring an application is greatly simplified in this environment — there is a single process to check out and pay for the applications. This eliminates the uncertainty inherent in entering into a transaction with an unknown merchant. Furthermore, distribution and installation are now as simple as downloading and deploying.

Another problem that app architecture attempts to solve is to make the melding of disparate business needs easier. In a given organization, it is likely that you use several different software packages, and other types of solutions in different languages, that may perform one business function very well. But suppose you need to integrate those business processes? What if you had QuickBooks for accounting, but you need to be able to get the relevant transaction information out of a SQL Server database for a web order? The usual approach has been to find the appropriate bridge, such as ODBC for QuickBooks, or maybe custom scripting, or one of many other possible solutions. The common factor in these solutions is that they are one-time glue, are not easy to reuse for other similar problems, and don't necessarily scale well. It can be quite frustrating when an organization is constantly refining its processes but does not want to dedicate resources for significant software engineering required to achieve specific tasks.

You are probably very much aware that this is why Access works so well on a worker's desktop; it makes it very easy to perform common automation. This is what Microsoft expects to achieve with Access web apps in a web browser.

To help unify this experience for Office applications, Microsoft will use the app architecture, along with the Office Store, to facilitate cross-platform distribution and in the process simplify development greatly.

There are two major classes of apps:

1. Apps for SharePoint
2. Apps for Office

Let's consider Apps for Office first. They are akin to Add-Ins, with which you are probably familiar. However, their major advantage is that Apps for Office are written in JavaScript and render their output as HTML5, meaning they will work on any platform, including Office Web Apps. Apps for Office are, by definition, client-side scripting and can be used to extend or enrich a document. For example, a Word document could have an app to render a page from Wikipedia with additional content regarding a word used in the document.

Because Apps for Office already come with a means of distribution and deployment, the process is greatly simplified in comparison to the steps required to distribute an Add-In for Office. Installing an Add-In would potentially require elevated privileges and additional components such as Visual Studio Tools for Office runtime and other dependencies. In contrast, an App for Office is simple to

install — whether it's free or not — and distribution and installation is all done via Office Store and/or App Catalog. Furthermore, Apps for Office enable some object model access so they can be used to perform some simple automation tasks.

In the preceding introduction to the current environment for Access and the new app architecture, you should have gained some insight into why apps are potentially exciting. However, we should be clear on what you are going to be doing. Although Access is an Office product, it does not generate Apps for Office; it actually generates Apps for SharePoint, which we will now examine in detail.

Apps for SharePoint have all the functionality we've discussed regarding Apps for Office and, in addition, can have server-side code written and executed as part of the process. Because apps integrate into SharePoint, we have a rich environment that enables us to focus on solving business problems, rather than getting all the plumbing done as you would need to do when patching together disparate software. An app can be designed to work with SharePoint, other apps, and even Apps for Office.

HOW IS AN APP HOSTED?

There are three different models for an App for SharePoint. In the first two, the app has server-side code. For simpler cases where there is no custom code, the app can be hosted entirely within SharePoint. Like Apps for Office, HTML5 and JavaScript are used with the SharePoint client libraries to provide the functionality in the app. SharePoint provides a safe sandbox for those apps, called "app web." Suppose we have a SharePoint farm with the address `http://contoso.com`, or, perhaps a hosted O365 site with the address `http://contoso.sharepoint.com`, and we install an App for SharePoint named "SuperCalc." The URL for web hosting the app may then end up as: `http://app-XXXXXXXXXXXXX.appcatalog.contoso.com/` or `http://app-XXXXXXXXXXXXX.contoso.sharepoint.com` where the Xs are strings of hexadecimal characters randomly generated when the app is provisioned. We'll look into what this means for security later in the chapter.

> **NOTE** *Note that for some environments, SharePoint administrators can customize the location of the app and therefore provide custom prefixes and a different domain to store the apps, so the URL may be formed differently than shown in the book. However, in all cases, each app will get its own unique URL no matter what SharePoint environment it has been provisioned to.*

When an App for SharePoint contains custom code, you can choose between auto-hosted or provider-hosted models. When an app is "auto-hosted," SharePoint will deploy the needed components using Windows Azure Web Sites and/or SQL Azure.

With a provider-hosted model, your organization provides a dedicated server that the app can reach to execute custom code outside the SharePoint environment, in similar fashion to how we would use a remote web service. Of course, you can mix the models, such as by using ASP.NET hosted on your server to execute custom action based on inputs from a SQL Azure database (an auto-hosted component).

In the context of Access web apps, however, there is no custom code, at least not in the usual sense of writing lines of instructions in your preferred language. All custom code you will author for an Access web app is in the form of macros. Even so, an Access web app technically does generate custom code, usually by converting macros into T-SQL objects (for example, stored procedures or views). It hosts the data in a SQL Server database. Because of its dependency on a separate SQL Server database, an Access web app falls in the category of an auto-hosted App for SharePoint.

For those needing more customizations beyond what macros can provide, Access web apps also support the web browser control introduced in Access 2010. The web browser control enables consumption of additional apps or other web components, which you'll learn about in later chapters. For those who choose to run SharePoint on-premises or subscribe to dedicated hosting services where you have access to the SQL Server used to store Access web apps, you may choose to perform additional customizations in T-SQL directly. Obviously, such customizations would be unsupported by Microsoft, and they would not be practical for a hosted environment. We briefly look at those options later in the book as well.

HOW IS AN APP DISTRIBUTED?

As we alluded to in earlier discussions, app architecture helps simplify distribution by building the channels of distribution into the architecture so you don't have to worry about figuring out how to distribute an app and deal with a plethora of installer problems such as overcoming antivirus programs, getting all dependencies right, and numerous other complications.

App Marketplace for Publicly Available Solutions

Let's start with public distribution. Learning from the success of application repositories for mobile devices, Microsoft will be providing an Office Store, hosted at `http://office.microsoft.com/store/`. Instead of having to host your own website, manipulating search engines to get good rankings, and marketing your app, you can place your app in the Office Store. That greatly simplifies the effort to get exposure for your app. Instead of scouring the entire Internet for potential buyers who may or may not have Office 2013, you can focus your marketing on people who already have Office 2013 and later and thus can take advantage of app architecture in your solution. Furthermore, it is possible to set up licensing arrangements, which helps enormously in simplifying the transaction made between your potential buyers (if you choose to sell an app) and yourself. Thus, the Office Store provides you with new opportunities to monetize, or at least popularize, your solutions with much less effort.

App Catalogs for Internal-Facing Solutions

You are no doubt familiar with the fact that many available Access solutions are not designed to be public-facing; they're rarely meant to be used by the public, but rather by employees or other "insiders" in an organization. For organizations that wish to manage internal-facing apps, SharePoint can provide an internal repository called an App Catalog. A SharePoint administrator can deploy App Catalogs, which then become repositories for installing apps to any site within the

SharePoint farm. This is useful in ensuring that you don't clutter the farm with apps that may not be applicable to everyone but may be needed by more than one site.

App Catalogs can also function as a mechanism for requesting apps from an Office Store. In cases where a given app requires payment, it is more likely that the person wanting the app isn't the same person signing the checks for the organization. Or perhaps the app is free but the administrators would prefer to review and evaluate whether the app is appropriate for the internal App Catalog. In this case, the person submits a request, which queues the app for review by administrators. Once review is completed, the app approved, and the licensing fee, if any, is paid, the app from the Marketplace becomes available on the App Catalog for internal consumption. Furthermore, the App Catalog can also function as a licensing server for apps that have a licensing model based on per-seat or per-site, enabling the organization to optimize assignment of the app.

On the other hand, creating an app doesn't always require presence of an App Catalog or going to the Office Store. When you want to use the app for a single site out of an entire farm (or entire Internet), you can just create an app directly on the site. SharePoint Document Libraries, SharePoint Custom Lists, Access web apps, and many more are examples of apps that do not need to go through an App Catalog or Office Store because you can create them directly on your SharePoint site, as demonstrated in Figure 1-7 and Figure 1-8.

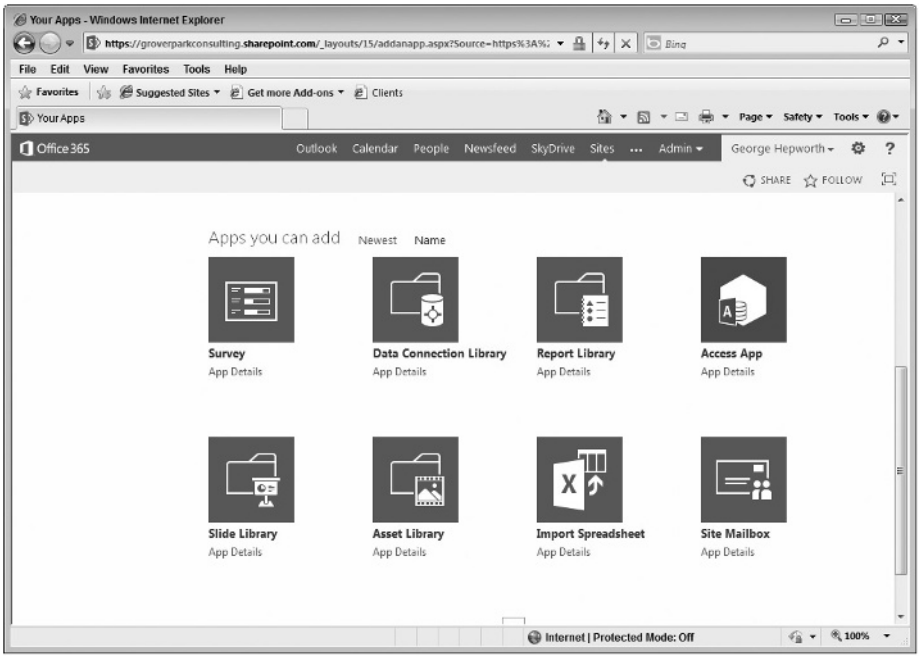

FIGURE 1-7: Creating an app via the SharePoint Site Contents page

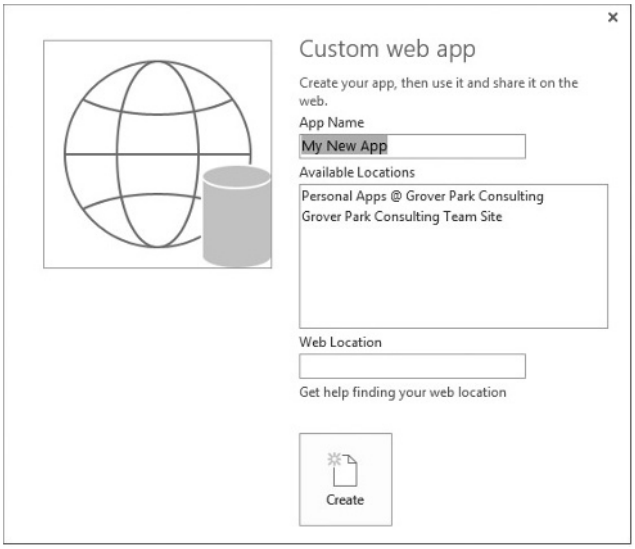

FIGURE 1-8: Creating an Access web app in Access 2013

By default, when you create a new app in Access 2013, you are actually adding a new app without going through an App Catalog or Office Store. This makes it very easy for you to deploy a new app. However, the downside of apps created in this manner is that there is no lifecycle management. If you want to be able to back up, perform upgrades, or monitor usage in various sites, then it may be more desirable for you to add an Access web app to a private developer site for development and then add the subsequent App Package to an App Catalog or the Marketplace. We will explore the issues surrounding lifecycle management in Chapter 14.

HOW ARE SECURITY AND TRUST MANAGED IN APPS?

One major enhancement SharePoint 2010 introduced to client libraries was that authentication was totally abstracted away. As long as you had a reference to a context object variable in the code, the authentication was already taken care of. Before you even get to run any SharePoint component, you have to log in, which establishes the security context. Because of this, there is no need to explicitly initiate an authentication process when invoking a web service within SharePoint. This continues into SharePoint 2013 and the new app architecture. SharePoint itself is very versatile in supporting different methods for authenticating users into the farm. A full discussion of different authentication protocols supported by SharePoint is beyond the scope of this book. However, it is sufficient to know that you do not necessarily need to plan on rolling out your own security for your Access web apps. In fact, you will need to resist the temptation to carry over the habit of rolling out your own access control from traditional Access development and get used to relying on SharePoint to provide the security context. In later chapters, we examine how you can effectively leverage the user context in SharePoint to provide access control and thus secure your Access web app.

However, our story does not end with merely authenticating the users. You also need to authenticate the app itself. Because an App for SharePoint may require additional privileges so it can interact with the rest of a SharePoint farm, it may be necessary to make an explicit trust to grant the app sufficient permissions to access other parts of SharePoint. Access web apps are no exception as you can see in Figure 1-9.

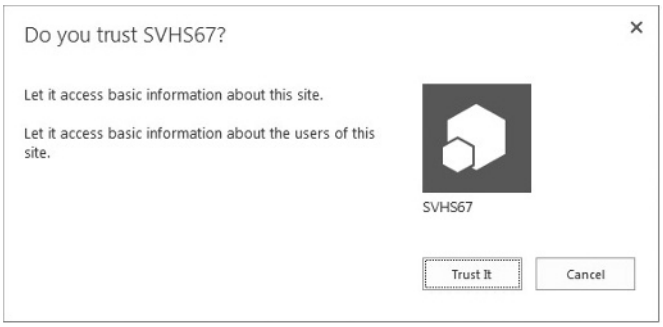

FIGURE 1-9: Trusting an Access web app

Typically, an Access web app needs to be able to access information about the SharePoint site to which it has been published, and to get information about the users of the site, as shown in Figure 1-9. Thus, installing an Access web app will require explicit trust. This step adds a layer of protection in that you will always know what an app needs to function, and you can decide if it is a reasonable request, especially when dealing with third-party apps. This provides a bit more transparency in contrast to traditional installer software, which is a black box in contrast. With a traditional installer, it was hard to know exactly what was installed. With the extra transparency, it becomes easier for administrators to make informed decisions, which is beneficial for you as Access developers interested in distributing an app.

In the preceding sections, you looked at different hosting models. We should make it explicit that there is a reason the hosting model is largely governed by what custom code is involved in an app. For a complex app, where reviewing the custom code can be difficult, providing a dedicated server apart from the SharePoint farm guarantees that custom code does not execute on the hosted SharePoint farm. An explicit trust decision is required when adding a provider-hosted app. Another enhancement introduced into apps is support for OAuth. OAuth is an open standard for authentication that allows servers to trust a user based on credentials provided by a specified third-party. OAuth also enables cross-domain authentication, which expands the reach of an app. If you need to invoke an external web service in a different domain without raising red flags, this can provide you with the means to build a secure app that is easy to trust without having to package everything in one monolithic domain. We hope this illustrates how app architecture makes manageability of custom solutions much easier.

Setting Up a SharePoint Site for Your Apps

As you just learned, Access web apps are not like Access desktop files, which are primarily under the developer's control. An app is not a file that you create and save on your local hard drive or network share. It is a web application created on the SharePoint platform and stored within the SQL Server database. Consequently, there is no way to create a new Access web app without access to a SharePoint 2013 server. Furthermore, SharePoint 2013 must be running the Access Services 2013 service application to enable creation of new Access web apps.

So, before we move on to discuss the specifics of creating web apps, let's take a quick look at some of the prerequisites when using the SharePoint 2013 server.

For small businesses, buying a SharePoint license is rarely practical. A much better solution is to consider a hosted environment. Microsoft offers Office 365, which is SharePoint hosted by Microsoft and can support Access web app solutions. There may be third-party hosting solutions available as well.

> **NOTE** *All Office 365 plans support Access, which means you can obtain a license to run Access 2013 and use your traditional Access solutions (i.e., client databases) with Access 2013 from Office 365. But if you want to host Access web apps on Office 365, then you must select a plan such as:*
>
> - *Office 365 Small Business Premium*
> - *Office 365 ProPlus*
> - *Office 365 Enterprise*
>
> *There may be other plans available. For up to date listings, refer to the Technet Article, "SharePoint Online Service Description"* at `http://technet.microsoft.com/en-us/library/jj819267.aspx`.

The move to a hosted environment, especially for application development tasks, can be a stumbling block for experienced Access developers who have frequently found themselves working outside the scope of the IT department. Frequently, Access solutions arise out of the necessity to meet a departmental need in the absence of IT department support or because of the inability of IT to respond to and act on business needs in a timely manner. Access web apps require involvement of an IT team, whether directly as an internal department or as support from a hosting provider in order to administer the SharePoint server. This is a profound change in the role played by many Access developers, so you should understand why this is happening.

Traditionally, Excel workbooks and Access databases have been problematic for IT departments. With Excel and Access files, there are many questions that IT departments cannot readily answer:

- Which version(s) of a given document is current?
- How many people use those documents?
- How are those documents secured?

Without answers to these questions, it's practically impossible to effectively manage the documents and assess what is required to provide adequate support. All too often, IT departments cannot budget sufficient funds to projects that need it the most, cannot prioritize projects properly, and so on. Microsoft is trying to solve manageability problems like these with enhancements in Office 2013 Apps.

Now, if you're thinking, "But those are enterprise problems; my clients are small businesses, and if they're lucky enough to have an IT department, it's probably a one-person department," you certainly are not alone. When you assess what a typical Access solution requires in terms of IT resources, it's rare that it's something that can be correctly implemented by non-IT personnel. For example, the best practice for sharing an Access desktop file is to split it, place the back end on a network share, and deploy a copy of the front end to each user's desktop. This requires some planning and configuration. If, like most Access developers, you're getting new clients today who are still sharing unsplit Access files among their users, that should indicate the problem is not trivial. Furthermore, it doesn't end at splitting; once the Access database is split, you must deal with the problems of distributing updates, making changes safely, maintaining backups, and more.

More important, users are no longer confined to a physical office. More and more people are working outside an office for various reasons. In the past, the solution to the problem of the traveling salesman was to use Jet Replication. However, replication certainly was not something that could be configured with a click of a button. It required even more careful planning and considerable expertise to deploy successfully.

Cross-platform usage is still another consideration. More people expect to be able to use an application on different platforms, so you can no longer assume you can count on a Windows-only solution. To address that requirement, Access web apps are based on HTML5 and JavaScript, which are accessible to a wide variety of hardware and operating systems, and thus you can build an application that can be used on any device — from a smartphone to a standard desktop computer.

Microsoft's answer for those small business owners who can't justify having an IT department, or even a full time IT person, is Office 365. Small business owners can shift the responsibility for securing, backing up, and managing their IT resources from themselves to Microsoft by signing up for a subscription service on Office 365. The Office 365 service is not limited to Access web apps; it also enables managing of Excel and Word documents. This helps make Office documents accessible to any authenticated and authorized user on any device. In addition, complex installation that would come with an on-premise SharePoint Server is abstracted away. Office 365 is meant to completely change how a small business manages its IT resources. Although the focus in this book is on Access web apps, the scope of Office 365 itself is not limited to Access solutions.

Setting Up an Office 365 Trial Account

For those who want to use Office 365 to host their Access web apps, the following section may be useful in getting a new Office 365 plan set up quickly. While the details of customizing and configuring Office 365 are beyond the scope of this book, these instructions will be useful when you want to follow the examples in the book to build your own Access web app. Those who already have access to an on-premises SharePoint server, an Office 365 plan, or another hosted provider can skip to the next section. Note that the specific steps provided may change as Microsoft frequently updates their site.

1. Using your preferred web browser, go to `http://office.microsoft.com/`.
2. Click Try it for free as shown in Figure 1-10.
3. Click the Try now link, as shown in Figure 1-11.
4. Enter the required information on the page.

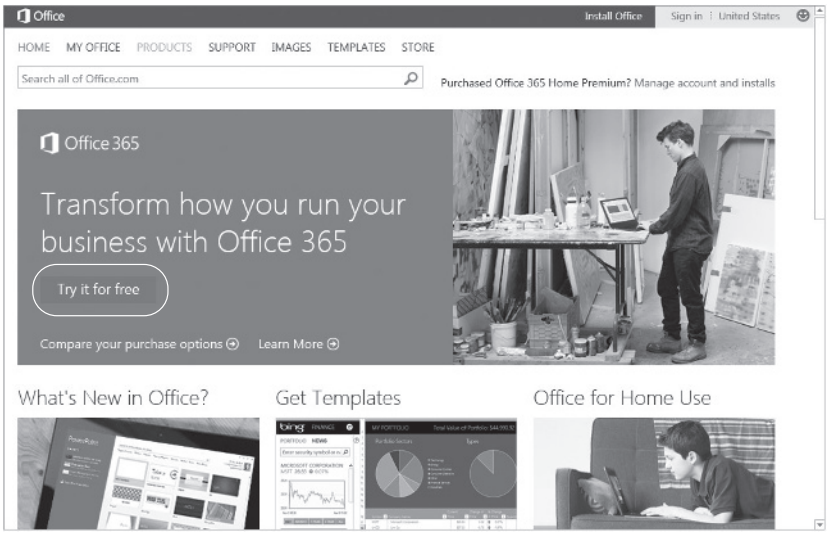

FIGURE 1-10: Office homepage

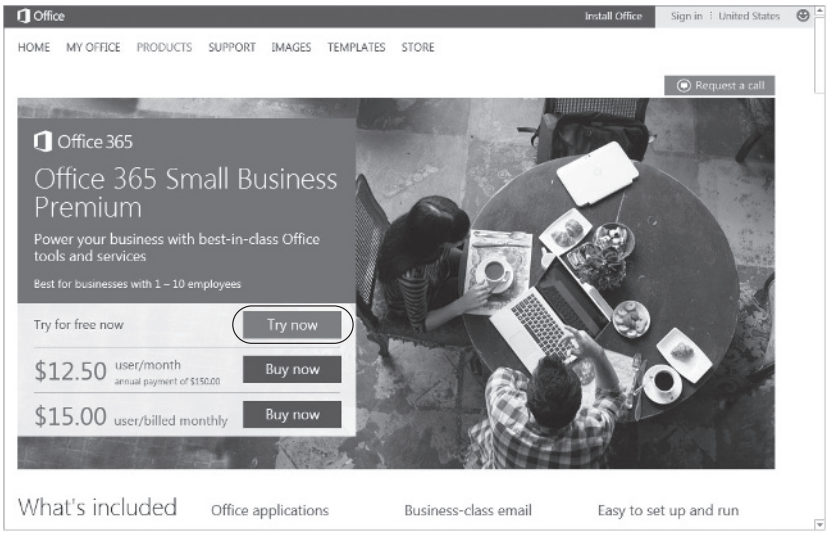

FIGURE 1-11: Trial page

> **NOTE** *You need to be aware of the difference in how e-mail addresses and site addresses are handled on Office 365. When you choose an organization ID such as "RedBarn," your e-mail address to sign into Office 365 will be* `yourname@redbarn.onmicrosoft.com`.
>
> *However, your Office 365 site, where your Access web apps would be discovered, would be hosted at* `www.redbarn.sharepoint.com`. *Don't be confused by this difference. Of course, it is possible to customize your e-mail address and site URL, but that is beyond the scope of this book.*
>
> *Furthermore, if you are using Windows 8, and your Windows login is linked to a Microsoft account such as a Hotmail account, the steps may differ because your Microsoft account already has you linked.*

5. Click create my account.
6. You'll be signed in and asked to provide a mobile number and e-mail address to which a forgotten login or password can be sent. Enter a mobile phone number (it must accept text messages) and the e-mail if not already filled in, and click Save and continue.
7. Click software.
8. Click Set up your software link. Ensure Office is selected so you see a page as depicted in Figure 1-12.

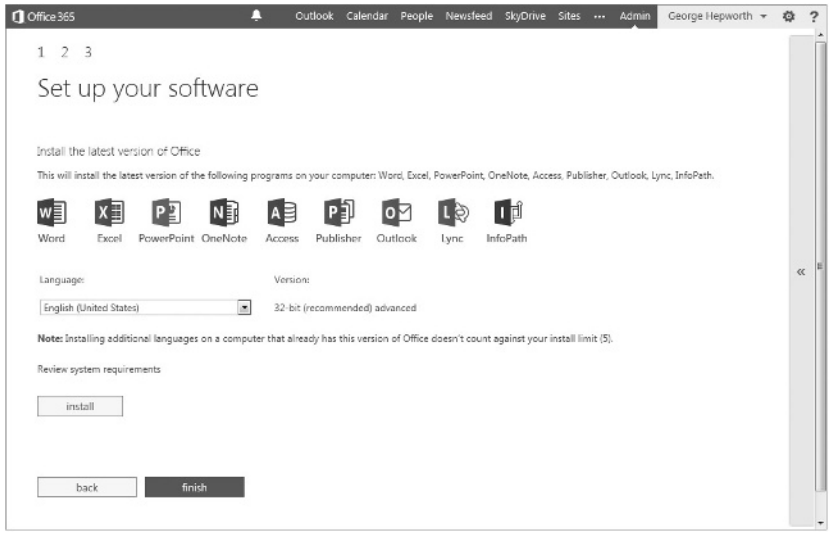

FIGURE 1-12: Office download page

9. You may need to wait until Office is ready. When you see an install button, click it.

Office installer will start streaming applications to you. Wait until you see Access 2013 listed in your Start menu. Then you're ready to create new Access web apps.

In the next chapter, where you learn about creating web apps and tables, you'll rely on the information you just learned here.

SOLVING BUSINESS PROBLEMS

Because we believe a sample application is a good way to introduce you to creating Access web apps, we'll have you build one to solve a real-world business problem and use it to illustrate the components of the web app as we discuss them. The following is a summary of the purpose of the database that you will build throughout the book. The complete app, `MaidToOrder.app` is available for download.

THE MAID TO ORDER WORK SCHEDULE DATABASE

Maid To Order is a ficitional private company that provides cleaning and maintenance services to commercial and private clients throughout its geographic region. Teams of professionally trained Maid To Order cleaners visit client sites to provide cleaning services contracted by the client. You will build a database application that supports the following features:

- Maid To Order managers will use the Work Schedule database to manage staff assignments and client requests.
- Maid To Order cleaning staff will use the Work Schedule database to stay informed of their assignments and report on completion of their assignments.
- Maid To Order customers will use the Work Schedule database to request and schedule cleaning services.
- The Maid To Order Work Schedule Database will support these requirements through both an in-house database and a web app via browser or mobile devices.

At appropriate points in the discussion, you'll learn the details of the database requirements and specifications. You will then learn how to complete the steps required to build the application.

SUMMARY

This chapter began with a review of the changes that Access 2013 introduced, starting with features that were deprecated and other features that were added. It also looked at the new tools included with Office 2013 to help effectively manage large numbers of Access solutions. You then turned to the anatomy of an app and looked at how it helps solve several problems encountered in traditional client and web solutions. You considered the channels of distribution available to apps, from the Marketplace for public distribution to App Catalogs for organizations.

The chapter concluded with an overview of security and trust for apps. With this information, you should now have a solid overview of the foundation Access web apps are built on, which will be very relevant in our subsequent discussions on building an Access web app. In the next five chapters, you will become familiar with basic operations in the new architecture, such as creating tables and queries that are different from client counterparts and identifying the difference. In subsequent chapters, you will look at building a complete solution. In the next chapter, you will learn how to create tables in and for Access web environments and how that differs from traditional table creation for client databases.