# 1

# Introducing the Project: TheBeerHouse

This chapter introduces the project that you're going to develop in this book. I'll explain the concept behind the sample website that is the subject of this book, but as you read along you should keep in mind that this is a general-purpose, data-driven, content-based style of website that can easily be modified to meet the needs of a myriad of real-world website requirements. Although you'll use many of the older features of ASP.NET, the clear focus of this book is directed at showing you how to leverage the powerful new features of ASP.NET MVC in a real-world, non-trivial website.

This book follows a "Problem–Design–Solution" approach in each chapter: The Problem section explains the business requirements for the module designed in that chapter, the Design section is used to develop your roadmap for meeting those requirements, and the Solution section is where you write the code to implement your design. This is unlike traditional computer books because the focus is not on teaching basic concepts, but rather showing you how to apply your knowledge to solve real-world business requirements. If you are new to ASP.NET, perhaps this is not the best book to start with, but if you're generally familiar with the basic concepts of web development and ASP.NET (any version of ASP.NET), you're ready to put that knowledge to use, and perhaps you want to learn about the new features in ASP.NET MVC, so fasten your seat belt!



## Problem

In State College, Pennsylvania, where I attended The Pennsylvania State University, more than half the population consists of students. With all these young people around, it goes without saying that there are a lot of pubs, bars, and places to spend the evenings and weekends with friends. Concerts, parties, shows, and other special events are commonplace. However, with all this competition, every pub must find something that the others don't have, something that's somehow appealing to its potential customers. Marketing plays a significant role, and your pub owner wants to be stronger in that area. He has always used traditional, printed marketing ads for his pub *TheBeerHouse* (a fictitious name), but he wants to expand into new media possibilities, starting with having his own exciting website. He thinks that this would be useful because, once customers become familiar with the site, they can go there to read about new specials and

events, and possibly receive a newsletter right in their e-mail inbox. They can also browse photos of past events, rate them, and share messages with other website visitors, creating virtual relationships that they can later continue face-to-face right in the pub! The general idea is appealing, especially considering that the target audience is well accustomed to using computers and browsing the web to find out information about news and events. A pub is typically a fun place full of life, and it's perhaps more appropriate for this type of project, rather than, say, a classic restaurant. However, even classic restaurants may like to consider this type of website as well.

# Design

The Design section of each chapter is devoted to discussing the problem and designing a solution. This usually means writing down a list of business requirements and desired features to implement, as well as the design of the necessary database objects for the data storage, and the structure of the classes to retrieve, manipulate, and present the data to the user. At the beginning of a project you start out by thinking about your client's needs, how you might meet those needs, and possibly even expand on them to give your client more functionality than the minimum needed, while still staying within your time limits and budgetary guidelines. As stated in the Problem section, your client in this scenario is a pub owner who wants to have a website to promote his pub, providing online information about upcoming events, reports about past events, and more. This initial idea can be expanded in many ways to create a site that has a lot more interesting things, which is good for its users who are also potential customers for the physical pub, and for the store owner. You can begin by writing down a list of features that a modern content-based site should have, and a few reasons why they are useful:

- **An appealing user interface.** Appearance is important, because it's the first thing users will notice — well before appreciating the site's functionality and services. But the graphics are not all that matters regarding the UI. The information on the site must be well organized and easily reachable. The site must be usable and provide a good (and possibly great) user experience, which means that users must find it easy to browse and interact with. Some attention should also be given to cross-browser compatibility; that is, ensuring that the site looks and behaves as desired from different platforms and browsers. This is especially true for sites like this one, where you can't know in advance which browser your customers will use, as you might know in the case of an intranet site for a corporation, for example.
- **A personalized user experience.** A successful content-based site owes its popularity to its users. Loyal users, who regularly visit the site, help write content, and participate in polls and special events, are those who guarantee that the site will keep growing. To build a vibrant community of active members, users must have some sort of *identity*, something that describes and distinguishes them among other members. Because of this, the site needs a registration feature as part of a larger authentication/authorization infrastructure. This will also be used to grant and restrict access to some areas of the site.
- **Dynamic content.** The site needs a constant supply of fresh content to stay alive and vibrant. If the content becomes stale, visitors will lose interest in the site and won't visit it anymore. A pub's site can't be very good unless it has regular updates about upcoming events, parties, and concerts. What's the point in visiting the site if it doesn't display photos that were shot at the last party? To facilitate a constant stream of new content, the site needs some mechanism that enables the editor to easily update it with dynamic content. Furthermore, the editor, who will be in charge of the content updates, will probably not be a technical person, so you must build some simple administration pages that make updates easy, even for nontechnical people.

- **Site-to-user communication.** Once the site has new content ready to be read, the site's manager must have some way to inform its users about this. Not all users visit the site every day, so the site manager must be proactive and notify the customers about recent updates. If customers have registered on the site and provided their e-mail address, they might also have requested to receive a newsletter notifying them about recent changes and additions to the site. Of course, there are also other ways to syndicate news, such as exposing syndication feeds to which users can register and then control from their favorite RSS reader, and get automatic notifications about news without having to visit the site daily to get the information.
- **User-to-site communication.** A site like this can also be a good opportunity to get feedback from customers about a variety of issues: What do they like most in a pub? What brand of beer do they prefer? Do they want to listen to live music while drinking with friends, or perhaps they don't like all that noise? Establishing some kind of user-to-site communication is important, and if you get a good number of responses it can even lead to strategic decisions and changes that may improve the business.
- **User-to-user communication.** If the presence of some sort of user-to-site communication is important, user-to-user communication may be even more so, because that's the central point of creating a community of loyal users who come to the site frequently to chat, discuss the news posted on the site, ask others about upcoming events, and more. This translates into more traffic on the site, and a feeling of membership that will pay off in both the short and long run.
- **An e-commerce store.** Once the pub's physical store has a strong customer base, the pub's owner may decide to expand it so that it supports an online store. In fact, the pub already offers a catalog of products for beer enthusiasts, such as glasses, T-shirts, key chains, and more. If the site has a lot of traffic, it may be a good way to promote these products so people can place orders without even visiting the pub in person. And once customers see a product and like it, they can rate that product to tell other people how much they like it. The online store must be easy to manage by nontechnical people, because it might possibly be the pub's owner who adds and edits products, and manages the orders, so there must be a module with a simple and intuitive UI that automates as many operations as possible, and guides the customers through the ordering process.
- **Localized content.** As mentioned previously, the pub is typically visited by a lot of customers coming from many different countries, and the pub's owner expects the same to happen for the website. Because of this, the site must be partially or fully translated into multiple languages, making it easy for most users to understand it. Not only text must be translated; information such as dates and numbers should also be displayed according to the user's preferred locale settings, so that nobody will misunderstand an announcement about an upcoming party or event.

To recap everything in a few words, the TheBeerHouse site will have everything a modern content-based site will have, including dynamic articles and news, polls for user-to-site communication, forums for user-to-user communication, newsletters and RSS feeds to notify members about new content on the site, an e-commerce store for selling products online, home page personalization, and content localization. Although the sample project is built around a fictitious pub, you'll recognize in this list of requirements the common features of the majority of content- and commerce-based sites you find online now, and sites that you're likely to develop in the near future, or maybe even sites you're developing right now.

# Solution

The Solution section of each chapter contains the instructions and actual code for implementing all of the features and requirements outlined and designed in the previous sections. For this first chapter, however, I'll give you a more detailed description of exactly what the following chapters cover, so that you can get a good idea of what the final result will be like.

In Chapter 2 you learn about the general concept behind MVC and then learn about Microsoft's implementation of MVC called ASP.NET MVC. You learn how an ASP.NET MVC application is constructed and what the terms model, controller, action, filter, route, and view are used for and how they relate to give you a basic understanding of the framework before you dive into creating code in ASP.NET MVC.

In Chapter 3 you build the site's design, the graphics, and the layout that's shared among all pages of the site, through the use of master pages. This will create a flexible design that is easy to maintain and customize through the use of CSS.

In Chapter 4 you lay down the foundations for building a flexible, easily configurable, and instrumented site. First you design a model or data access layer (DAL) for use through the new language integrated query (LINQ) in .NET 3.5. Then you build a controller or business logic layer on the top of the model, with the required validation logic, transaction management, event logging, and caching as necessary. Finally, you look at the view or user interface (UI) as the presentation layer of the site, to create complex and feature-rich, data-driven pages.

In Chapter 5 you integrate ASP.NET's membership infrastructure into the site, to create user registration forms and supporting logic to authenticate/authorize users. You also use the `Profile` module, which allows you to declaratively define user-level properties that are automatically persisted to a durable medium, quite different from the well-known traditional `Session` state variables that only last as long as the user browses the site on one occasion. You build a complete management console to enable administrators to see the list of members, disable members that behave badly on the site, and view and edit each user's profile.

In Chapter 6 you build a sort of Content Management System, a module that enables administrators to completely manage the site's articles, news, and blog posts from an intuitive UI, accessible also by non-technical users. The module will integrate with the built-in membership system to secure the module and track the authors of the articles, and will have a syndication service that publishes a syndication feed of recent content for a specific category, or for every category, and will support ratings and comments, among many other features. The result will be quite powerful, enabling the editor to prepare richly formatted content in advance, and schedule it for automatic publication and retirement, so that the site's content updates are as simple as possible, and require the least effort and time. At the end of the chapter you will have an excellent understanding of how to create an MVC application and how all the working parts come together to give you a functional site.

In Chapter 7 you implement a solution for creating and managing multiple polls on the site. It will feature an administration console for managing the polls through a web browser, a way that enables you to change the polls that show on all the pages, as well as a history page for viewing archived polls.

In Chapter 8 you enrich the site with a complete module for sending out newsletters to members who registered for them in their profile page. The module will enable you to send out the e-mail newsletters from a background thread, instead of the main thread that processes the page request, so that the

page won't risk timeouts, and more important, so that the editor will not be left with a blank page for minutes at a time. You use AJAX (Asynchronous JavaScript and XML Programming) to implement partial-page updates that provide real-time feedback about the newsletter being sent in the background. Finally, end users will be able to look at past newsletters listed on an archive page.

In Chapter 9 you create a forums system from scratch, which supports multiple subforums with optional moderation, lists threads and replies through custom pagination with different sorting options, crowd-sourcing mechanisms like those seen on Digg, Reddit, and Stackoverflow, and other features typical of the most recent forum-like software. You also provide complete administration features (deleting, editing, approving, and closing threads and posts).

In Chapter 10 you add a working e-commerce store with most of the essential features, including a complete catalog and order management system, a persistent shopping cart, integrated online payment via credit cards, product ratings, product stock availability, rich formatting of a product's descriptions, including text and images, configurable shipping methods and order statuses, and much more. You implement all this in relatively few pages, because it will leverage the good foundations built in previous chapters, and of course the ASP.NET built-in membership and profile systems, as well as other new features provided by ASP.NET MVC.

In Chapter 11 you make the site's home page fully localizable to an additional language and will support the user's preferred locale settings when displaying dates and numbers. All this can now be done easily with ASP.NET, thanks to its automatic resource generation, implicit and explicit localization expressions, and strongly typed and dynamically compiled global resources.

Finally, in Chapter 12 you look at the different ways to deploy an ASP.NET MVC site, either on a local IIS server, to a remote production site, or to an inexpensive shared hosting server. The new ASP.NET compilation model enables you to use a simple `XCOPY` deployment that includes everything. You also learn how to deploy the local SQL Server Express database to a remote full-featured SQL Server 2008 instance, and how you can create an installer package for distributing the application to automate as many installation tasks as possible.

# Summary

In this first chapter you were given an overview of an aggressive plan to develop a highly functional content-based website that shows you how to use ASP.NET MVC to its full capability. I gave you a broad idea about what we're going to discuss, design, and implement throughout the rest of the book. In each chapter, you learn something new about ASP.NET MVC, and at the end of the book you will also have created a real-world site with most of the features required by modern content-centric sites and e-commerce stores. Furthermore, the site you develop in this book may provide a good deal more functionality than any site you've designed in the past, and the relatively small development effort will enable you to do more than you thought possible in a small amount of time. Microsoft has stated that one of its key goals in the ASP.NET MVC release is to make a developer's job easier and provide more flexibility and control over the HTML content sent to the browser from the server. You can do this by separating your concerns ahead of time into the model, view, and controller, reducing the amount of effort required to implement common functionality. This gives the developers more time to focus on business needs and enables them to offer more advanced functionality to empower users and site administrators, while keeping the site maintainable, testable, and scalable. This book will help you judge whether Microsoft has met this goal. Let the adventure begin!