Kentico provides three basic development models. You can choose the model that best suits your needs:

- **Portal Engine** – allows you to build websites in a **browser-based interface** using components called web parts. Only requires programming in Visual Studio when creating custom components.
- **ASPX Templates** – can be chosen by ASP.NET developers who prefer to create websites using standard ASP.NET architecture and standard development tools, such as **Visual Studio**. This model requires you to be familiar with ASP.NET web form development and have at least basic programming knowledge of C# or VB.NET.
- **MVC** – allows developers to create websites using the Model-View-Controller architectural pattern (based on the ASP.NET MVC framework). Working with this model requires knowledge of programming and ASP.NET MVC. Note that not all features are supported for MVC development; see Supported and unsupported Kentico features on MVC sites for more details.

If required, the *Portal Engine* and *ASPX Templates* models can be combined on a single website. For example, you can place pages using ASPX templates onto a portal engine website, and even insert custom ASPX pages implementing your own applications. On the other hand, you can create ASPX page templates with areas that can be edited through the portal engine.

	Portal Engine	ASPX Templates	MVC
How you work	You build the website and design pages using a browser-based interface. No programming knowledge is required for common tasks.	You build ASPX pages (web forms) that are used to display content from Kentico . At least basic programming knowledge of ASP.NET and either C# or VB.NET is required.	You implement models, controllers and views for rendering pages rendering content retrieved from the Kentico database. Requires knowledge of MVC architecture, ASP.NET and C# or VB.NET.
How you assemble pages	You use built-in or custom web parts that you place into customizable page layouts (HTML code with placeholder zones for web parts).	You use built-in or custom ASP.NET server controls and place them onto the ASPX pages. These are standard web forms that are part of the web project, so you can also work with code behind files. It is also possible to place web parts on the page templates if the required server control is not available.	The appearance of the content you display is defined completely through MVC views, which are composed of HTML and inline code. The content structure in Kentico is represented by content only pages, which do not have page templates as they serve as content repositories only.
Master pages and visual inheritance	Subpages can nest within any ancestor pages. You can break the nesting on any level.	Page templates may inherit content from a master page, which works just like a standard ASP.NET master page (.master file). Pages do not inherit content from their parents in the website content hierarchy.	Visual representation is handled by the MVC application completely.
Custom code integration and extensibility	You can create your own user controls (ASCX files) or web parts (ASCX files with a portal engine interface) if you need to integrate custom functionality. Any custom controls or code can be added to the web parts placed on the website.	You build standard ASPX pages with code behind files, which means you can place any custom controls and code onto the page.	You prepare the content of the MVC views and the functionality of the controller and model classes in Visual Studio, so have full control over customization.

Advantages	• Easier and faster way to build websites. • ASP.NET programming knowledge is not required for common tasks. • You can build the whole website very quickly, using only a web browser.	• Standard ASP.NET architecture. • You can use your favorite development tools, such as Visual Studio.	• Model-View-Controller architecture. • The option of using the Razor view engine • Development via standard tools (Visual Studio).
Disadvantages	• Proprietary architecture and development process.	• Requires ASP.NET programming knowledge. • The design of the web pages cannot be fully managed via the browserbased administration interface.	• Development tasks require knowledge of ASP.NET MVC and programming. • The design of the web pages cannot be managed via the browserbased administration interface.