1. **MVC:**
   1. **Model:** Set of classes that describes the data we are working with and the business rules for how the data can be changed and manipulated
   2. **View:** Applications UI
   3. **Controller:** handles communication from the user, overall application flow and application-specific logic
2. **MVC features history:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MVC1** | **MVC2** | **MVC3** | **MVC4** | **MVC5** | **MVC6** |
|  | UI helpers | Razor view engine | ASP.NET Web API | One ASP.NET |  |
|  | Scaffolding (automatic code generation) | Data annotations | Enhancements to default project templates | New web project experience |  |
|  | Attribute based model validation | Improved model validation | Mobile project template using jQuery mobile | ASP.NET identity |  |
|  | Strongly typed HTML helpers | Dependency resolution | Display modes | Bootstrap templates |  |
|  | Asynchronous controllers | Global action filters | Task support for asynchronous controllers | Attribute routing |  |
|  | Rendering subsections of a page | Unobstructive javascript | Bundling and minification | ASP.NET scaffolding |  |
|  |  | jQuery validation |  | Authentication filters |  |
|  |  | JSON binding |  | Filter overrides |  |
|  |  | NuGet |  |  |  |

1. **Important MVC features:**
   1. **One ASP.NET:** This eliminates the choice of choosing MVC or web forms before starting a project. You can add MVC to your project at any time.
   2. **ASP.NET identity system:** This facilitates membership login to MVC application. MVC5 has simplified this identity system so that it suits with all ASP.NET web applications (web forms, web pages, MVC applications, web API, SignalR etc)
   3. **Attribute routing:** To specify routes by placing annotations on your controller classes or action methods.
   4. **ASP.NET Scaffolding:** used to generate boilerplate code based on your model classes.
   5. **Authentication filters:** To control access by role or other custom logic. Difference between authentication and authorization
   6. **Filter override:** Filter overrides mean that you can exclude a controller or action from executing a global filter.
2. **Installing MVC 5 and creating applications:**
   1. **Requirements:**
      1. .NET 4.5
   2. Project templates
      1. Empty
      2. Web forms
      3. MVC
      4. Web API
      5. Single page application
      6. Facebook
      7. Azure mobile service
   3. Folder and core references
      1. Web forms
      2. MVC
      3. Web API
   4. Enabling unit testing
   5. Configuring authentication
      1. No authentication 🡪 for public websites
      2. Individual user accounts
      3. Organizational accounts 🡪 Used for accounts that authenticate via some form of Active Directory.
      4. Windows authentication 🡪 For intranet applications
3. **MVC Application structure:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Directory** | **Description** |
| 1 | /Controllers |  |
| 2 | /Models |  |
| 3 | /Views |  |
| 4 | /Scripts | To store javascript files |
| 5 | /fonts | To store bootstrap fonts |
| 6 | /Content | CSS, images and other project files |
| 7 | /App\_Data | Data files we read or write |
| 8 | /App\_Start | Configuration files |

The above folder structure is not a mandatory or required one. It’s just a convention. Big projects split code among different projects to provide more flexibility.

1. **ASP.NET MVC and conventions:**
   1. **Controllers** ends with Controller suffix and are stored in Controllers directory
   2. **Views:** single views directory for all views of your application with controller as the folder name. **Ex:** ProductController has a corresponding view in /Views/Product directory.

**Chapter 2: Controllers**

1. **Controllers:** Controllers in MVC pattern are responsible for responding to user input, often making changes to the model in response to user input. In this way, Controllers are concerned with the flow of the application, working with data coming in, and providing data going out to the relevant view.