1. Architektura składa się z 3 części: Widoku, Kontrolerów i Modelu.
   Widok - pokazuje dane z bazy poprzez generowanie UI lub interfejsu użytkownika. Widok jest tworzony dzięki danym uzyskanym od bazy przy pomocy kontrolerów.
   Kontrolery - jest pośrednikiem w komunikacji między widokami a modelami. Przekazuje modelowi informację o potrzebnych danych, które następnie przesyła do widoków z wyjaśnieniem jak ich użyć.
   Model - odpowiedzialna za utrzymywanie danych, połączonych z bazą. Odpowiada za wykonywanie wszystkich zadań związanymi z operacjami na danych i logikę danych. Model odpowiada na zapytania kontrolerów, które nie komunikują się bezpośrednio z bazą danych. Zatem model jest pośrednikiem w tej komunikacji.

2. Każda część ma swój folder w projekcie. Nazwy plików i folderów są napisane za pomocą notacji Pascal Case. Generalnie konwencja kontrolera jest taka, że routing może łatwo znaleźć odpowiedni kontroler bez dodatkowej konfiguracji.
   Model: Tutaj nie ma standardowej konwencji. Powodem tego jest to, że platforma ASP.NET MVC nigdy nie dotyka bezpośrednio modeli. Modele są dostępne tylko z logiki w kontrolerze, więc framework nie musi o nich wiedzieć (importuje się je za pomocą odpowiedniej komendy). Jednak często dzieli się je w struktury podobne do Widoku.
   Kontrolery: Dla każdego kontrolera odnoszące się do innego widoku przypada jeden plik z kontrolerem. Podobna sytuacja dot. kontrolerów akcji.
   Widok: w widoku znajdziemy foldery odpowiadające za widok konkretnych stron lub części stron. Folder Shared zawiera segmenty wspólne dla wielu stron, jak sam CSS, czy układ strony. Pozostałe foldery zawierają dwa rodzaje plików: Index.cshtml i *.cshtml. Domyślnie widoki powinny znajdować się w folderze o nazwie kontrolera i mieć taką samą nazwę jak akcja, która je wywołuje, dzięki temu metoda wywołania View() w akcji może działać bez określania widoku.

3. Słownik ViewData może być używany do przekazywania danych z kontrolera do widoku. Drugą metodą jest przekazywania danych z kontrolera do widoku za pomocą modelu widoku.

4. Kontroler obsługuje segmenty adresów URL i wartości ciągu zapytania, których wartości przekazuje do modelu. Model może używać tych wartości do wykonywania zapytań do bazy danych. Np: /[Controller]/[ActionName]/[Parameters].

5. Atrybut HttpPost określa, jak metodę można wywoływać, ale tylko w przypadku żądań POST. Można stosować atrybut [HttpGet] do metody, ale nie jest to konieczne, ponieważ [HttpGet] jest wartością domyślną.

6. Atrybut ValidateAntiForgeryToken służy do zapobiegania fałszowaniu żądania i jest sparowany z tokenem zabezpieczającym przed fałszerstwem wygenerowanym w pliku widoku za pomocą pomocnika tagów formularza. Generuje on ukryty token zabezpieczający przed fałszerstwem, który musi być zgodny z wcześniej opisanym tokenem.

7. Można deklaratywnie określić reguły walidacji w jednym miejscu (w klasie modelu), a reguły zostaną wymuszone wszędzie w aplikacji.