معماری MVC (Model-View-Controller) یک الگوی طراحی نرم‌افزار است که به منظور جداسازی منطق برنامه از رابط کاربری و کنترلرها استفاده می‌شود. این معماری به سه جزء اصلی تقسیم می‌شود که هر کدام نقش خاصی در برنامه دارند:

اجزای معماریMVC

1. Model(مدل):

- وظیفه : مدیریت داده‌ها و منطق تجاری (Business Logic) برنامه.

- کاربرد : مدل نماینده داده‌های برنامه است و معمولاً شامل توابعی برای دسترسی، ذخیره و به‌روزرسانی داده‌ها در پایگاه داده یا منابع دیگر است. مدل فقط داده‌ها و قوانین مرتبط با داده‌ها را مدیریت می‌کند.

- مثال : در یک برنامه فروشگاه آنلاین، مدل می‌تواند شامل کلاس‌هایی برای محصولات، کاربران، سفارشات و تعاملات با پایگاه داده باشد.

2. View(نما:)

- وظیفه : نمایش داده‌ها به کاربر.

- کاربرد : ویو مسئولیت نمایش اطلاعات به کاربر را دارد و هیچ منطق تجاری در آن وجود ندارد. ویو فقط داده‌های مدل را دریافت کرده و به شکل قابل مشاهده برای کاربر نمایش می‌دهد.

- مثال : در یک برنامه فروشگاه آنلاین، ویو می‌تواند شامل صفحات HTML/CSS برای نمایش فهرست محصولات، سبد خرید و صفحات سفارش باشد.

3. Controller(کنترلر:)

- وظیفه : مدیریت تعاملات کاربر و به‌روزرسانی مدل و ویو.

- کاربرد : کنترلر درخواست‌های کاربر (مثل کلیک‌ها، ورودی‌های فرم و ...) را پردازش می‌کند، مدل مناسب را به‌روزرسانی کرده و ویو مناسب را انتخاب می‌کند تا داده‌های به‌روز شده را نمایش دهد.

- مثال : در یک برنامه فروشگاه آنلاین، کنترلر می‌تواند شامل کدهایی باشد که ورودی‌های کاربر را مدیریت می‌کنند، مثل اضافه کردن محصول به سبد خرید یا پردازش یک سفارش.

جریان کار در معماریMVC

1. درخواست کاربر : کاربر یک درخواست را از طریق رابط کاربری (View) انجام می‌دهد.

2. پردازش توسط کنترلر : کنترلر درخواست را دریافت کرده و آن را پردازش می‌کند. ممکن است نیاز به تغییر وضعیت مدل یا به‌روزرسانی داده‌ها داشته باشد.

3. به‌روزرسانی مدل : کنترلر مدل را به‌روزرسانی می‌کند یا داده‌های جدید را از مدل دریافت می‌کند.

4. نمایش به‌روزرسانی‌شده : کنترلر ویو مناسب را انتخاب کرده و مدل به‌روزرسانی‌شده را به ویو ارسال می‌کند.

5. نمایش داده‌ها به کاربر : ویو داده‌های به‌روزرسانی‌شده را به کاربر نمایش می‌دهد.

مزایای معماریMVC

1. جداسازی نگرانی‌ها : با تقسیم برنامه به سه بخش جداگانه، نگهداری و به‌روزرسانی هر بخش آسان‌تر می‌شود.

2. تسهیل توسعه و نگهداری : تغییرات در یکی از اجزا (مثلاً ویو) معمولاً نیازی به تغییرات در دیگر اجزا (مثلاً مدل یا کنترلر) ندارد.

3. تست‌پذیری بهتر : به دلیل جداسازی منطق برنامه، می‌توان هر بخش را به صورت جداگانه تست کرد.

4. قابلیت توسعه : افزودن ویژگی‌های جدید به برنامه با استفاده از معماری MVC ساده‌تر است، زیرا اجزا مستقل از هم هستند.