✅1. Explain MVC architecture in ASP.NET MVC

Answer:

- **Model**: Represents the application's data and business logic. It retrieves/stores model state in a database.
- **View**: UI. Represents the presentation layer (Razor/HTML files).
- **Controller**: Acts as a mediator between Model and View. Handles user input, manipulates the model, and returns a view.

Flow:
User sends a request → Route determines the controller → Controller processes and calls the Model → Model processes data → Controller returns View → HTML rendered to browser.

✅2. What is the request life cycle in ASP.NET MVC?

Answer:

1. **Routing**: Maps URL to a controller/action using `RouteConfig.cs`.
2. **MVC Handler**: Receives request and initializes controller.
3. **Controller Initialization**: Executes the target action.
4. **Model Binding**: Binds HTTP request data to parameters or model.
5. **Action Execution**: Controller processes logic.
6. **Result Execution**: ViewResult or JsonResult returned.
7. **View Engine**: Razor parses `.cshtml` and renders HTML.

✅3. Difference between ViewBag, ViewData, and TempData

Feature	ViewBag	ViewData	TempData
Type	Dynamic	Dictionary	Dictionary (TempDataDictionary)
Lifetime	Current request	Current request	Survives 1 redirect only
Use case	Simple UI data	Simple UI data	Pass data across redirects

---

✅4. What is Routing in MVC?

Answer:
Routing maps incoming URLs to controller actions. It avoids physical file mapping.

Example:

```
csharp
CopyEdit
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
);
```

Types:

- **Convention-based** (defined in `RouteConfig.cs`)
- **Attribute-based** (decorated on controller or action)

✅5. What are Action Filters?

Answer:
Action filters allow custom logic to execute **before or after** action methods.

Types:

- `Authorize` – for authentication
- `HandleError` – for exception handling
- `OutputCache` – for caching
- Custom filters using `ActionFilterAttribute`

Example:

```
csharp
CopyEdit
public class LogActionFilter : ActionFilterAttribute {
    public override void OnActionExecuting(ActionExecutingContext
filterContext) {
        // Logging logic
    }
}
```

✅6. What is Model Binding?

Answer:
Model binding automatically maps HTTP request data (form fields, query strings, etc.) to action method parameters or model properties.

```
csharp
CopyEdit
public ActionResult Register(User user) { ... }
```

Advantages:

- Reduces boilerplate
- Strong typing
- Validation friendly

✅7. How is validation done in ASP.NET MVC?

Answer:

- Server-side: using **Data Annotations** (`[Required]`, `[Range]`, `[StringLength]`, etc.)
- Client-side: uses **jQuery validation** + **unobtrusive JavaScript**

Example:

```
csharp
CopyEdit
public class Product {
    [Required]
    public string Name { get; set; }

    [Range(1, 1000)]
    public int Price { get; set; }
}
```

✅8. Partial View vs View Component

Feature	Partial View	View Component
Reusability	Yes	Yes
Logic support	No (only rendering)	Yes (can run C# logic)
Use case	Shared UI fragments	Complex reusable logic + UI

Partial View:

```
html
CopyEdit
@Html.Partial("_ProductList", Model.Products)
```

View Component:

```
csharp
CopyEdit
public class ProductSummaryViewComponent : ViewComponent {
    public IViewComponentResult Invoke() {
        return View("_ProductSummary", data);
```

```
    }
}
```

✅9. What is TempData?

Answer:
`TempData` is used to pass data between **two consecutive requests** (e.g., after redirect).

```
csharp
CopyEdit
TempData["Success"] = "Record saved!";
return RedirectToAction("Index");
```

Internally uses **session** but clears automatically after read.

✅10. How is Dependency Injection handled in ASP.NET Core MVC?

Answer:
In `.NET Core`, DI is built-in.

```
csharp
CopyEdit
// Register in Startup.cs
services.AddScoped<IProductService, ProductService>();

// Inject in Controller
public class ProductController : Controller {
    private readonly IProductService _service;
    public ProductController(IProductService service) {
        _service = service;
    }
}
```

Lifetimes:

- `AddSingleton`: one instance for app
- `AddScoped`: one per request
- `AddTransient`: new instance every time

✅11. How to handle exceptions globally?

Answer:

- **HandleErrorAttribute**:

```
csharp
CopyEdit
[HandleError(View = "CustomError")]
```

- **Global filters**:

```
csharp
CopyEdit
filters.Add(new HandleErrorAttribute());
```

- **Middleware (ASP.NET Core)**: Use `UseExceptionHandler` in `Startup.cs`

✅12. How is caching implemented in MVC?

Answer:
Output Caching (non-Core):

```
csharp
CopyEdit
[OutputCache(Duration = 60)]
public ActionResult Index() { ... }
```

ASP.NET Core Caching:

- In-memory caching
- Response caching
- Distributed caching (Redis)

```
csharp
CopyEdit
services.AddMemoryCache(); // Startup.cs
```

✅13. How to secure MVC application?

- Use `[Authorize]` for controller/action protection
- Apply **AntiForgeryToken**:

```
html
CopyEdit
@Html.AntiForgeryToken()
csharp
CopyEdit
[ValidateAntiForgeryToken]
public ActionResult SubmitForm(Model m) { ... }
```

- Encode output to avoid XSS: `@Html.Encode(model.Name)`

✅14. Entity Framework Integration

- Code-first or DB-first
- Use DbContext to interact with DB

```
csharp
CopyEdit
public class AppDbContext : DbContext {
    public DbSet<Product> Products { get; set; }
}
```

Migrations:

```
bash
CopyEdit
Add-Migration InitialCreate
Update-Database
```

✅15. What are Areas in MVC?

Answer:
Areas separate large applications into **functional sections** (e.g., Admin, User).

```
bash
CopyEdit
Add Area "Admin"
```

Creates its own controller, views, route config.