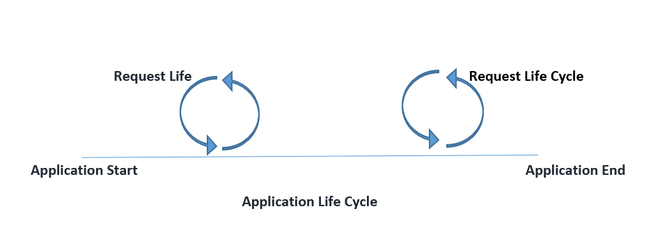
**Asp.NET MVC Important Interview Questions**

**ASP.NET MVC Life Cycle**

The life cycle is basically is set of certain stages which occur at a certain time.

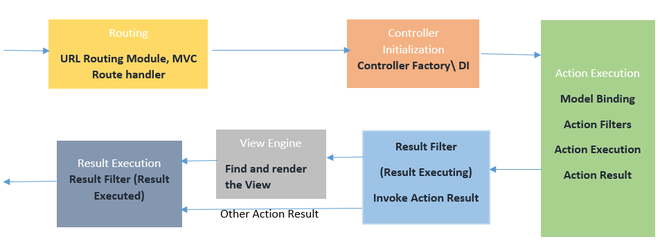


MVC actually defined in two life cycles, the application life cycle, and the request life cycle.

The **application life cycle**, in which the application process starts the running server until the time it stops and it tagged the two events in the startup file of your application. i.e. the application start and end events

This is separate from the **request life cycle**, which is the sequence of events or stages that executed every time an HTTP request is handled by the application

**MVC Request Life Cycle:**



*MVC Request Life Cycle*

* The Starting point for every MVC application begins with routing. After that, the received request figures out and finds how it should be handled with the help of the URL Routing Module. The routing module is responsible for matching the incoming URL to routes that we defined in our application.
* Every route has an associated route handler associated with them. If the request is matched to a route defined in our MVC application, the MVC Route Handler retrieves an instance of the MVC HttpHandler.
* The MVC Handler Start the process of initializing and executing a controller. The MVC framework handles converting route data into a specific controller that can handle the requests.
* This is achieved through the MVC components like the Controller Factory and Activators which are responsible for creating an instance of the Controller class.
* After the controller has been created, component called the action invoker finds and selects an appropriate Action method to invoke on our controller.
* Model binding happens before the method is called, which maps the data from our HTTP request to the parameters of our action methods. Also called action filter before and after action results are generated.
* Now after our action result has been prepared, the next stage triggers, which is Result Execution.
* If the result is a view type, the View Engine will be called and it will find the view and render it.
* If not a view type then the action result will execute on its own. This Result Execution is nothing but generates an actual response to that original HTTP request.

**1. What is ASP.NET MVC, and how does it differ from ASP.NET Web Forms?**

**Answer:** ASP.NET MVC is a web application framework that follows the Model-View-Controller architectural pattern. It provides more control over HTML and is suitable for building SEO-friendly, RESTful, and highly testable web applications. ASP.NET Web Forms, on the other hand, follows a more event-driven and component-based model for web development.

**2. What are the main components of ASP.NET MVC architecture?**

**Answer:** The main components are:

* **Model:** Represents the application's data and business logic.
* **View:** Represents the user interface and presentation logic.
* **Controller:** Handles user input, processes requests, and controls the flow of the application.

**3. Explain the request lifecycle in ASP.NET MVC.**

**Answer:** The request lifecycle involves the following steps:

* **Routing**: The URL is mapped to a controller action.
* **Controller Selection**: The controller is selected.
* **Action Selection**: The appropriate action method is chosen.
* **Model Binding**: Data from the request is bound to action method parameters.
* **Action Execution**: The action method is executed and result execution.
* **View Rendering**: If result is view type. The view is rendered, and HTML is generated.
* **Response Sent**: The HTML response is sent to the client.

**4. What is the purpose of the Global.asax file in an ASP.NET MVC application?**

**Answer:** The Global.asax file is used to handle application-level events and configuration. You can use it to register routes, set up authentication, and handle application-level events like application start, session start, and more.

**5. Explain the role of a controller in ASP.NET MVC.**

**Answer:** A controller is responsible for handling user requests, processing data, and deciding which view to render. It acts as an intermediary between the model and the view, orchestrating the flow of the application.

**6. What are Action Filters in ASP.NET MVC?**

**Answer:** Action Filters are attributes that can be applied to controller actions to perform tasks like authentication, caching, exception handling, and more. They are used to add behavior to action methods.

**7. What is the purpose of ViewBag, ViewData, and TempData in ASP.NET MVC?**

**Answer:**

* **ViewBag:** It is a dynamic object used to pass data from a controller to a view. Data in ViewBag is not strongly typed.
* **ViewData:** Similar to ViewBag, it is used to pass data from a controller to a view, but it is a dictionary-based key-value pair collection.
* **TempData:** TempData is used to pass data from the current request to the next request. It is often used for redirect scenarios.

**8. What is the role of a ViewModel in ASP.NET MVC?**

**Answer:** A ViewModel is a pattern that helps to shape the data and behavior for a specific view. It allows you to consolidate data from multiple models and present it in a format that's suitable for a particular view, helping to avoid overloading views with business logic.

**9. Explain the concept of Routing in ASP.NET MVC.**

**Answer:** Routing is the process of mapping URLs to controller actions. In ASP.NET MVC, you define routes in the RouteConfig.cs file to specify how URLs should be interpreted. The routing engine matches the URL to a route and invokes the corresponding controller action.

**10. What is the importance of attribute-based routing in ASP.NET MVC?**

**Answer:** Attribute-based routing allows you to define routes directly on controller actions using attributes, such as **[Route]**. This gives you more control and makes the routing configuration more concise and readable.

**11. How can you secure an ASP.NET MVC application?**

**Answer:** You can secure an ASP.NET MVC application by implementing:

* Authentication: Use authentication mechanisms like forms authentication, Windows authentication, or third-party authentication providers.
* Authorization: Implement role-based or claim-based authorization to restrict access to specific resources.
* HTTPS: Enforce secure connections using HTTPS.
* Input Validation: Sanitize and validate user inputs to prevent security vulnerabilities.

**12. What is Dependency Injection, and how is it used in ASP.NET MVC?**

**Answer:** Dependency Injection is a design pattern that promotes loose coupling by injecting dependencies into classes instead of creating them internally. In ASP.NET MVC, you can use a container like Unity or Ninject to configure and inject dependencies into controllers and other components, making your code more testable and maintainable.

**13. How does model binding work in ASP.NET MVC?**

**Answer:** Model binding is the process of populating a model or action method parameters with data from the request. ASP.NET MVC automatically maps form fields, query parameters, and route data to action method parameters based on their names and types. This process simplifies the extraction of data from HTTP requests.

**14. What are the different types of ActionResult in ASP.NET MVC?**

**Answer:** ActionResult is used to return results from controller actions. Common types of ActionResult include:

* ViewResult: Returns an HTML view to be rendered in the browser.
* JsonResult: Returns data in JSON format.
* FileResult: Returns binary data as a file download.
* RedirectResult: Redirects the client to another URL.
* ContentResult: Returns raw content with a specified content type.

**15. Explain the concept of Areas in ASP.NET MVC.**

**Answer:** Areas are used to organize a large ASP.NET MVC application into smaller, more manageable sections. Each area can have its own controllers, views, and models, allowing for a structured approach to building complex applications. Areas are often used for modular development.

**16. What is the purpose of the Razor view engine in ASP.NET MVC?**

**Answer:** Razor is the default view engine in ASP.NET MVC, used to generate dynamic HTML content. It offers a compact syntax and enables developers to embed server-side code (C#) within views. This makes it easier to create dynamic, data-driven web pages.

**17. How do you handle exceptions in ASP.NET MVC?**

**Answer:** Exceptions can be handled in ASP.NET MVC using various techniques, including:

* Global Error Handling: Implementing error handling in the **Application\_Error** event in Global.asax.
* Exception Filters: Applying custom exception filters to controller actions.
* Custom Error Pages: Configuring custom error pages to provide a user-friendly error message.
* ELMAH (Error Logging Modules and Handlers): A library for logging and handling errors in web applications.

**18. What is the difference between ViewBag and TempData?**

**Answer:** ViewBag is used to pass data from a controller to a view for the current request, while TempData is used to pass data between different requests (from one action to another) or across redirects. TempData is stored temporarily in session.

**19. How do you enable attribute-based routing in ASP.NET MVC?**

**Answer:** Attribute-based routing can be enabled by adding the following line to the **RegisterRoutes** method in the **RouteConfig.cs** file:

routes.MapMvcAttributeRoutes();

This allows you to use attributes like **[Route]** on controller actions to define custom routes.

**20. Explain the concept of validation in ASP.NET MVC.**

**Answer:** Validation in ASP.NET MVC ensures that user input is valid and secure. You can use data annotations to define validation rules for model properties, and the framework provides validation helpers to perform server-side and client-side validation. Common validation libraries include DataAnnotations, FluentValidation, and jQuery Validation.

**21. What is the purpose of the Web API in ASP.NET MVC?**

**Answer:** ASP.NET Web API is a framework for building RESTful web services, primarily used for serving data to client applications, such as mobile apps and single-page applications. It uses HTTP methods and can return data in various formats, including JSON and XML.

**22. What is the purpose of the ValidateAntiForgeryToken attribute in ASP.NET MVC, and how does it work?**

**Answer:** The **ValidateAntiForgeryToken** attribute is used to protect against Cross-Site Request Forgery (CSRF) attacks. It ensures that a form submission comes from a legitimate source by including a hidden anti-forgery token in the form. In the controller action, this attribute checks if the submitted token matches the one generated for the request. If they don't match, the request is considered invalid.

**23. What is scaffolding in ASP.NET MVC, and how does it simplify development?**

**Answer:** Scaffolding is a code generation framework in ASP.NET MVC that automates the creation of common elements, such as controllers, views, and data access code. It simplifies development by generating boilerplate code based on data models, reducing the amount of repetitive work required when building standard CRUD (Create, Read, Update, Delete) operations.

**24. Explain the concept of partial views in ASP.NET MVC.**

**Answer:** Partial view is a small view that contains content that can be shared among multiple views. Partial views are reusable components in ASP.NET MVC used to render a portion of a page rather than the entire page. They can be used to encapsulate and render specific parts of the user interface and can be included in multiple views, promoting code reusability.

HTML helper has two methods for rendering the partial view: Partial and RenderPartial.

<p>

@Html.Partial("PartialViewExample")

</p>

<p>

@{

Html.RenderPartial("PartialViewExample");

}

</p>

**25. How does ASP.NET MVC handle state management compared to ASP.NET Web Forms?**

**Answer:** ASP.NET MVC follows a stateless approach, meaning it does not maintain the state of pages across multiple requests. Developers have more control over managing state, which can be done through mechanisms like TempData, ViewState, or by using client-side technologies like cookies and session.

**26. What is the role of the JsonResult and ContentResult in returning data from a controller action?**

**Answer:** **JsonResult** is used to return data in JSON format from a controller action. It is commonly used when building AJAX-based applications. **ContentResult** allows you to return raw content with a specified content type, which can be useful for returning non-HTML data such as XML, plain text, or binary content.

**27. Explain the concept of HTML Helpers in ASP.NET MVC.**

**Answer:** HTML Helpers are methods that generate HTML markup in views. They simplify the process of generating HTML elements and can be used to create form controls, links, and other HTML elements. ASP.NET MVC provides built-in HTML Helpers, and you can create custom ones to suit your specific needs.

**28. How can you handle form submissions in ASP.NET MVC, and what is the role of the [HttpPost] attribute?**

**Answer:** Form submissions are typically handled using controller actions. The **[HttpPost]** attribute is applied to an action to specify that it should respond to HTTP POST requests. It is used to differentiate between HTTP GET and POST requests and ensures that the action is only invoked for POST submissions.

**29. What is the purpose of the Web.config file in an ASP.NET MVC application?**

**Answer:** The **Web.config** file is used to store configuration settings for an ASP.NET MVC application. It includes settings related to authentication, authorization, custom error pages, database connections, and other application-specific configurations.

**30. How can you optimize the performance of an ASP.NET MVC application?**

**Answer:** Performance optimization in ASP.NET MVC can be achieved through techniques like:

* Caching: Implement output caching, data caching, and client-side caching.
* Minification and Bundling: Combine and minify CSS and JavaScript files.
* Database Optimization: Optimize database queries and use appropriate indexing.
* Asynchronous Programming: Use async/await to prevent blocking operations.
* Load Balancing: Distribute requests across multiple servers using load balancing.
* Content Delivery Networks (CDNs): Use CDNs for serving static content.
* Proper resource management: Dispose of resources properly, use connection pooling, and reduce unnecessary object creation.

**31. What are Areas in ASP.NET MVC, and why are they useful?**

**Answer:** Areas in ASP.NET MVC allow you to organize your application into smaller, self-contained modules. Each area can have its own set of controllers, views, models, and routes. This is particularly useful for breaking down a large application into manageable sections, often following a modular approach to development.

**32. How can you enable and configure authentication in an ASP.NET MVC application?**

**Answer:** To enable authentication in an ASP.NET MVC application, you can use tools like Identity Framework for user management. You need to configure authentication settings in the **Web.config** file, implement the necessary controllers and views for registration and login, and use authentication attributes like **[Authorize]** to restrict access to specific actions or controllers.

**33. Explain the role of ActionResults in ASP.NET MVC.**

**Answer:** ActionResults are the return types of controller action methods in ASP.NET MVC. They represent the result of an action and determine what should be sent back to the client. Common ActionResult types include ViewResult (returning HTML views), JsonResult (returning JSON data), FileResult (returning files), and ContentResult (returning plain content).

**34. How do you handle errors and exceptions in ASP.NET MVC?**

**Answer:** Error handling in ASP.NET MVC can be done by using try-catch blocks in action methods, or by implementing error handling filters like **HandleErrorAttribute**. You can also configure custom error pages in the **Web.config** or use third-party libraries like ELMAH for detailed error logging and notification.

**35. What is the purpose of the TempData dictionary, and when is it typically used?**

**Answer:** TempData is a dictionary used to store data that needs to be preserved across redirects. It's often used to pass data from one action to another within the same session, making it useful for scenarios like passing messages or data between different parts of a multi-step process.

**36. Explain the concept of Output Caching in ASP.NET MVC.**

**Answer:** Output caching is a performance optimization technique in ASP.NET MVC that stores a rendered view in memory and serves it directly to subsequent requests, bypassing the execution of the controller action and view generation. It helps reduce server load and improve application response times.

**37. What are the advantages of using the Entity Framework in ASP.NET MVC for data access?**

**Answer:** Entity Framework is an Object-Relational Mapping (ORM) framework that simplifies data access in ASP.NET MVC applications. Advantages include automatic code generation, a strongly-typed query language (LINQ), easy database schema changes, and a consistent way to work with various database systems.

**38. What is the purpose of URL routing in ASP.NET MVC, and how do you configure routes?**

**Answer:** URL routing in ASP.NET MVC maps URLs to controller actions. You can configure routes in the **RouteConfig** class by defining patterns using placeholders like **{controller}** and **{action}**. The routing engine matches incoming URLs to routes and invokes the appropriate controller action based on the URL pattern.

**39. What is the difference between the ViewBag and ViewData in ASP.NET MVC?**

**Answer:** Both ViewBag and ViewData are used to pass data from a controller to a view, but the key difference is that ViewBag uses dynamic typing, while ViewData uses a dictionary. ViewBag is more convenient to use, but ViewData is a bit more flexible when working with different data types.

**40. How can you implement authentication and authorization in ASP.NET MVC applications?**

**Answer:** You can implement authentication and authorization by configuring authentication methods (e.g., Forms Authentication, Windows Authentication, or Identity Framework), defining roles and permissions, using the **[Authorize]** attribute on controller actions, and managing user access through role-based or claim-based authorization.

**41. What are the different types of filters in ASP.NET MVC, and how are they used?**

**Answer:** ASP.NET MVC provides various filters such as Action Filters, Authorization Filters, Result Filters, and Exception Filters. Filters allow you to add or modify behavior in the MVC pipeline. For example, Action Filters can be used to add pre-processing or post-processing logic to action methods, while Authorization Filters control access to actions.

**42. Explain the concept of Dependency Injection (DI) and how it is used in ASP.NET MVC.**

**Answer:** Dependency Injection is a design pattern that helps in achieving loose coupling by injecting dependencies into classes rather than creating them internally. In ASP.NET MVC, you can use DI containers like Unity, Autofac, or Ninject to inject dependencies into controllers, services, and other components. This promotes modularity, testability, and maintainability.

**43. How can you implement client-side validation in ASP.NET MVC?**

**Answer:** To implement client-side validation, you can use JavaScript libraries like jQuery Validation. You annotate your model properties with validation attributes, and ASP.NET MVC will generate JavaScript code for client-side validation based on these attributes. This helps ensure data integrity before a form is submitted.

**44. What is the purpose of the BundleConfig class in ASP.NET MVC, and how does it help with optimizing web applications?**

**Answer:** The BundleConfig class is used to bundle and minify CSS and JavaScript files in ASP.NET MVC. By grouping multiple CSS or JS files into a single bundle, you reduce the number of HTTP requests, which can significantly improve page load times and overall application performance.

**45. Explain the concept of RESTful web services and how they can be implemented in ASP.NET MVC.**

**Answer:** REST (Representational State Transfer) is an architectural style for designing networked applications. In ASP.NET MVC, you can create RESTful web services by using ASP.NET Web API, which is a framework for building HTTP-based APIs. Web API allows you to expose resources and actions through HTTP methods, making it suitable for building RESTful services.

**46. What is the role of the @Html.Action method in ASP.NET MVC views?**

**Answer:** The **@Html.Action** method is used to invoke a child action from a view. It allows you to render the output of another controller action within a view, helping you modularize and reuse components across views.

**47. How can you improve the security of an ASP.NET MVC application to protect against common web vulnerabilities?**

**Answer:** To enhance security, you should:

* Validate and sanitize user inputs.
* Implement proper authentication and authorization.
* Protect against Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) attacks.
* Use parameterized queries and stored procedures to prevent SQL injection.
* Secure sensitive data and use encryption.
* Keep the framework and libraries up to date to patch security vulnerabilities.

**48. What is the purpose of the ActionName attribute in ASP.NET MVC?**

**Answer:** The **ActionName** attribute is used to specify an alternative name for an action method, allowing you to have more descriptive or SEO-friendly (Search Engine Optimization) URLs while still keeping the method names in your code. This attribute is useful when you want to define custom route patterns.

**49. How do you implement unit testing in ASP.NET MVC applications?**

**Answer:** Unit testing in ASP.NET MVC can be performed using testing frameworks like MSTest, NUnit, or xUnit. You can write tests for your controllers, models, and services to ensure that they behave as expected. Dependency Injection makes it easier to isolate components for testing.

**50. What is the role of the RoutePrefix attribute in ASP.NET Web API, and how does it differ from routing in ASP.NET MVC?**

**Answer:** The **RoutePrefix** attribute is used in ASP.NET Web API to define a common prefix for a group of controller actions. It helps to simplify route configuration in Web API. While ASP.NET MVC uses route attributes to define routing for actions, Web API uses route attributes like **RoutePrefix** and **Route** to define the URL structure for API controllers.

**51. What is the role of the TempData dictionary in ASP.NET MVC, and how does it differ from ViewBag and ViewData?**

**Answer:** **TempData** is used to store data between requests. It is similar to **ViewData** and **ViewBag**, but with one key difference: **TempData** is intended for one-time use between requests, while **ViewData** and **ViewBag** are used for passing data from controllers to views within the same request.

**52. What are the advantages of using Razor Pages in ASP.NET Core compared to traditional ASP.NET MVC?**

**Answer:** Razor Pages is a page-based model in ASP.NET Core that simplifies the development of web applications. Advantages include a simpler file structure, a more focused and self-contained approach to handling requests, and improved separation of concerns, making it easier to build smaller, modular components.

**53. How can you implement authentication using third-party identity providers like Google, Facebook, or Microsoft in ASP.NET MVC applications?**

**Answer:** You can implement external authentication using OAuth or OpenID Connect protocols. ASP.NET MVC provides middleware for integrating with various identity providers. You typically configure these providers and handle the callback to authenticate users with their external accounts.

**54. What are attribute routes, and how are they different from conventional routing in ASP.NET MVC?**

**Answer:** Attribute routing is a way to define routing directly on controller actions and is available in ASP.NET MVC 5 and later. It allows you to specify the URL pattern for an action using attributes like **[Route]**. This offers more control and makes the routing configuration more concise compared to conventional routing, where routes are defined in the RouteConfig file.

**55. Explain the concept of Areas in ASP.NET Core, and how do they differ from Areas in previous versions of ASP.NET MVC?**

**Answer:** Areas in ASP.NET Core are similar in concept to those in earlier versions of ASP.NET MVC. They allow you to organize your application into separate modules with their own controllers, views, and models. The key difference is that in ASP.NET Core, Areas are not a built-in feature, but they can still be implemented using namespaces and folder structures.

**56. What is the purpose of AJAX in ASP.NET MVC, and how is it typically implemented?**

**Answer:** AJAX (Asynchronous JavaScript and XML) is used in ASP.NET MVC to enable dynamic, client-side updates without requiring a full page reload. AJAX is implemented by making asynchronous requests to the server using JavaScript and handling the response to update specific parts of the page. jQuery is often used to simplify AJAX requests.

**57. Explain the role of ModelState in ASP.NET MVC and how it is used for server-side validation.**

**Answer:** **ModelState** is a dictionary that contains the current state of model binding and validation. It is used to store error messages and validation results for model properties. In server-side validation, you can check **ModelState.IsValid** to determine whether there are any validation errors on the submitted data.

**58. How can you implement custom error handling in ASP.NET MVC to provide user-friendly error pages and log detailed error information?**

**Answer:** To implement custom error handling, you can use the **customErrors** element in the **Web.config** file to redirect users to error pages. Additionally, you can use the **Application\_Error** event in Global.asax to log detailed error information. Third-party libraries like ELMAH can be used for advanced error logging and notification.

**59. What are the main advantages of using asynchronous programming with ASP.NET MVC, and how do you use async/await in controller actions?**

**Answer:** Asynchronous programming in ASP.NET MVC improves application responsiveness and scalability. By using the **async** and **await** keywords, you can make non-blocking calls to external resources or I/O operations, allowing your application to efficiently handle more requests and reduce resource consumption.

**60. How do you implement session management in ASP.NET MVC, and what are the advantages and disadvantages of using sessions?**

**Answer:** Session management in ASP.NET MVC is typically achieved using the **Session** object. Sessions can be used to store user-specific data, but they come with disadvantages such as increased server resource usage and potential scalability issues. It's essential to use sessions judiciously and consider alternatives like cookies or caching.

**61. What is session and its type?**

In web development, a session is a way to store and manage user-specific data between HTTP requests. It allows you to maintain state information for a particular user as they navigate through a web application. Sessions are typically used to remember user data and provide a personalized experience. There are different types of sessions in web development, including:

1. **In-Memory Session:**
   * **Type:** This type of session data is stored in the server's memory.
   * **Advantages:** Fast access to session data; suitable for small-scale applications.
   * **Disadvantages:** Limited scalability; session data is lost if the server is restarted.
2. **Cookie-Based Session:**
   * **Type:** Session data is stored on the client side as cookies.
   * **Advantages:** Scalable; no server-side storage required; data persists between server restarts.
   * **Disadvantages:** Limited data storage capacity due to cookie size limits; potential security concerns.
3. **Database-Backed Session:**
   * **Type:** Session data is stored in a relational database (e.g., SQL Server).
   * **Advantages:** Scalable and reliable; data persists between server restarts; suitable for larger applications.
   * **Disadvantages:** Slower access to session data compared to in-memory sessions; may require additional configuration and resources.
4. **Cache-Based Session:**
   * **Type:** Session data is stored in a caching system (e.g., Redis, Memcached).
   * **Advantages:** Scalable and fast access to session data; suitable for distributed applications.
   * **Disadvantages:** Requires a caching system setup and management.

The choice of session type depends on the specific requirements and constraints of your web application. In-memory sessions are simple and efficient but may not be suitable for large-scale applications. Cookie-based sessions are lightweight but have limitations on data storage. Database-backed and cache-based sessions provide more scalability and data persistence but require additional infrastructure.

ASP.NET, for example, offers session state management options, and you can configure it to use different session types based on your application's needs. It's important to consider factors like scalability, data size, security, and data persistence when choosing the right session type for your web application.

**62. What is cookie and its type?**

A cookie is a small piece of data that a web server sends to a user's web browser. The browser stores this data and sends it back to the server with subsequent requests. Cookies are used for various purposes in web development, such as tracking user sessions, personalizing user experiences, and storing user preferences.

While there are several types of cookies, you can primarily categorize them into two main types:

1. **Session Cookies:** These are temporary cookies that exist only for the duration of a user's browsing session. They are stored in the browser's memory and are deleted when the user closes the browser. Session cookies are commonly used for tasks like maintaining user sessions, such as remembering a user's login status.
2. **Persistent Cookies:** These cookies have an expiration date and are stored on the user's device even after the browser is closed. They persist until they expire or are manually deleted. Persistent cookies are often used for long-term purposes, such as remembering user preferences or providing personalized content over multiple sessions.

**Q:** **What are web services and why should we use them?**  
**Web services are a standardized way for developing interoperable applications i.e enabling an application to invoke a method of another application.** These applications can be on the same computer or different computers. Web services use open standards and protocols like **HTTP**, **XML**and **SOAP.**Since these are open and well known protocols, applications built on any platform can interoperate with web services. For example, a JAVA application can interoperate with a web service built using .NET. Similarly a web service built using JAVA can be consumed by a .NET application.  
  
**Hyper Text Transfer Protocol (HTTP)**is the protocl widely used by web services to send and receive messages.  
The messaging protocol is **SOAP.**SOAP stands for Simple Object Access Protocol. SOAP messages are in XML format.

##### What is a captcha form?

The word captcha is actually an acronym that stands for ‘completely automated public Turing test to tell computers and humans apart’. Quite a mouthful isn’t it?

What this means, in a nutshell, is that a captcha test is a tool that helps to distinguish a human user from a computer user online.

Captcha tests are often added to websites to stop them from receiving spam through the likes of contact forms.