In ASP.NET MVC, handling HTTP requests is fundamental to processing client requests and delivering responses.

**1. Request**

The Request object in ASP.NET MVC is an instance of HttpRequestBase that provides access to all HTTP request data, such as the URL, headers, query string, and form data. It is used to retrieve data that is sent by the client in the request, including GET and POST parameters.

Example: Accessing Request Parameters

```
public ActionResult Index()
{
    string queryParam = Request.QueryString["name"]; // Access query string parameter "name"
    string formParam = Request.Form["username"];     // Access form parameter "username"
    ViewBag.QueryParam = queryParam;
    ViewBag.FormParam = formParam;
    return View();
}
```

In this example:

Request.QueryString["name"] retrieves the value of the name query string parameter from the URL.

Request.Form["username"] retrieves the value of the username form field from a POST request.

**2. HttpRequestBase**

HttpRequestBase is a class that represents the HTTP request sent to the server. It is used as the base class for HttpRequest and provides access to the incoming request data like headers, cookies, and the body of the request. Most of the time, you interact with HttpRequest directly in MVC, but HttpRequestBase is important when working with unit tests and mock requests.

Key properties of HttpRequestBase:

QueryString: Retrieves data from the query string (for GET requests).

Form: Retrieves data from the body of a POST request (form data).

Cookies: Retrieves cookies sent with the request.

Headers: Retrieves the HTTP headers sent by the client.

Example: Using HttpRequestBase

```
public ActionResult GetRequestInfo()
{
   string userAgent = Request.UserAgent; // Retrieve the user-agent from headers
   string referrer = Request.UrlReferrer?.ToString(); // Retrieve the referring URL
   ViewBag.UserAgent = userAgent;
   ViewBag.Referrer = referrer;
   return View();
}
```

Here, Request.UserAgent gives the user agent string, which contains details about the browser, and Request.UrlReferrer gives the URL of the referring page.

**3. [HttpPost] and [HttpGet] Annotations**

The [HttpPost] and [HttpGet] attributes are used in ASP.NET MVC to specify the type of HTTP request a controller action should handle. These annotations are used to decorate controller methods and define which HTTP method (GET or POST) will trigger each method.

[HttpGet]: The HttpGet attribute is used to indicate that the action method will handle GET requests. GET requests are typically used for retrieving data or rendering a form.

[HttpPost]: The HttpPost attribute is used to indicate that the action method will handle POST requests. POST requests are used to submit data, such as when a user submits a form.

Example: Using [HttpGet] and [HttpPost]

```
// GET: Display the form
[HttpGet]
public ActionResult Create()
{
   return View();
}
// POST: Process form submission
[HttpPost]
public ActionResult Create(FormCollection form)
{
   string name = form["Name"];
   string email = form["Email"];

  // Process the form data (e.g., save to the database)
   return RedirectToAction("Index");   }
```

In this example:

The Create action decorated with [HttpGet] handles the initial request to load the form.

The Create action decorated with [HttpPost] handles the form submission, processing the form data when the user submits it.

Another Example:

```
// GET request to display the form
[HttpGet]
public ActionResult Register()
{
    return View();
}

// POST request to process form data
[HttpPost]
public ActionResult Register(string name, string email)
{
    // Validate and process the data (e.g., save to database)
    ViewBag.Message = "Registration successful!";
    return View();
}
```

**Register.cshtml (View):**

```
@using (Html.BeginForm())
```

```
{
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" />

    <label for="email">Email:</label>
    <input type="text" id="email" name="email" />

    <input type="submit" value="Register" />
}
```

In this example:

The GET method displays the registration form when the user navigates to the Register page.

The POST method processes the form data when the user submits the form, validating the input and displaying a success message.

**4. FormCollection**

FormCollection is an object in MVC that contains the form data sent in a POST request. It acts as a dictionary where each form field is represented as a key-value pair. You can use it to retrieve form data submitted by the client in a POST request.

Key-value pair: The form field names are used as keys, and the values are the data entered by the user in the form.

Example: Using FormCollection

```
[HttpPost]
```

```
public ActionResult SubmitForm(FormCollection form)
{
    string username = form["Username"];
    string password = form["Password"];

    // Process the form data (e.g., authenticate user)
    return View();
}
```

Here, FormCollection form captures the form data. You access individual form fields by using their names (form["Username"] and form["Password"]). This is commonly used when the form is submitted via POST.

**Summary of Key Concepts:**

Request: A representation of the HTTP request made by the client, providing access to query strings, form data, headers, etc.

HttpRequestBase: The base class for HTTP requests, used for handling incoming requests and retrieving request data.

[HttpGet]: Used to decorate methods that handle GET requests (usually for displaying data or forms).

[HttpPost]: Used to decorate methods that handle POST requests (usually for form submissions or data creation).

FormCollection: A collection that represents the data submitted in a form, accessed as key-value pairs.