Self-Study

1. In mvc when comes the need of sending the id to controller for editing or deleting, There are two ways one is sending it through the action attribute of the form and the second way is to send it through a hidden input So In what cases could each one be the best practice ?

In ASP.NET MVC, sending an ID to a controller for editing or deleting can be done either through the action attribute of the form or via a hidden input field. Each approach has its best practices depending on the scenario:

Sending ID through the Form Action Attribute

- **Best Practice When:**
    - The ID is part of the resource's URL and directly identifies the resource being acted upon.
    - You want to follow RESTful URL conventions, e.g., `/Controller/Edit/5` or `/Controller/Delete/5`.
    - The form does not need to be dynamically changed after page load.
    - The URL itself should clearly reflect the resource identity for bookmarking, logging, or SEO.
- **Advantages:**
    - Cleaner and more semantic URLs.
    - Makes it easier for routing and link sharing.
    - The ID is visible in the URL, which can aid debugging and crawlers.

Sending ID through a Hidden Input Field

- **Best Practice When:**

    - The ID needs to be submitted with the form POST request as part of the form data.
    - The data is dynamic or can be changed client-side (e.g., with JavaScript) before submitting.
    - The form contains multiple data fields and the ID is just one part of the payload.
    - When using non-RESTful URL patterns where the resource identifier is not in the URL.
    - When the operation should be performed strictly by POST to improve security (less visible in URL).
- **Advantages:**
    - Keeps URL clean (ID not exposed in URL).
    - Easier to bind multiple form fields including the ID.
    - Supports scenarios where the ID is fixed per form session, but the form can dynamically change and needs the ID as hidden data.

Example:

- Form Action URL:

```
<form action="/Products/Edit/5" method="post">
```

- Good when the ID is integral to the route.
- Hidden Input Field:

```
<form action="/Products/Edit" method="post">
    <input type="hidden" name="Id" value="5" />
```

- Good when posting multiple fields including ID without exposing it in the URL.

Choosing between these depends on architecture goals, RESTfulness, security preferences, and complexity of the form data submitted.

2. What is a relative path and a full path ?

What is a Relative Path?

A relative path specifies the location of a file or directory relative to the current working directory or the position of another file. It does not start from the root of the file system but from a location considered the "current directory."

- It uses special notations such as:
    - `.` to represent the current directory
    - `..` to represent the parent directory
- Relative paths are typically shorter and more flexible in scenarios where files or directories may move as long as their relative positioning remains the same.

Example:
If you are in `/home/user/documents` and want to access a file in `/home/user/documents/reports/report1.txt`, the relative path would be:

```
reports/report1.txt
```

or

```
./reports/report1.txt
```

To access a file one directory up:

```
../pictures/image.png
```

What is a Full Path (Absolute Path)?

A full path or absolute path specifies the complete address to a file or directory starting from the root directory of the file system. It always points to the same location regardless of the current working directory.

- It starts from the root directory, such as:
    - On Unix/Linux systems, it starts with a forward slash `/`

    - On Windows, it starts with a drive letter followed by a colon, e.g. `C:\`
- Provides the exact location of the resource regardless of where the command or program is executed.

Example:

- Unix/Linux:

```
/home/user/documents/reports/report1.txt
```

- Windows:

```
C:\Users\User\Documents\reports\report1.txt
```

Key Differences:

- Flexibility: Relative paths are more flexible for project portability.
- Specificity: Full paths precisely locate a file, independent of context.
- Usage: Use relative paths for internal references within projects, use full paths for unambiguous file access.

These definitions help manage file access paths effectively in programming, scripting, and system navigation.