Asp.net and Design Patterns

**Introduction:**

* A problem can have more than one solution.
* Problems with only one solution are well-known and not a big deal.
* Problems with no solution don't need discussion.
* The challenge is when a problem has multiple solutions.
* Four individuals conducted research on problems with multiple solutions.

**Research:**

* They documented approximately 27 recurring problems.
* Explored all possible solutions for each problem.
* Identified the best solution for each problem.
* These problems occur rarely but may happen sporadically.
* The focus was on recurring problems faced by every programmer.

**Patterns:**

* Patterns are the best solutions to recurring problems.
* A pattern forms when two or more people follow the same solution.
* Objectively, a pattern is the best solution to a recurring problem.

**Types of Patterns:**

* Two types of software patterns: architectural patterns and design patterns.
* Architectural patterns provide high-level solutions to architectural problems.
* Design patterns are low-level solutions for design problems.
* Data patterns exist but are separate from software engineering.

**Importance of Design Patterns:**

* Design patterns are essential for programmers.
* They help understand and apply solutions in daily work.
* Unfortunately, design patterns are not typically covered in the curriculum.

**Summary:**

* Software patterns are solutions to recurring problems.
* Two main types: architectural and design patterns.
* Design patterns are important for programmers.
* They provide solutions for common design issues.

**Patterns in ASP.NET:**

* ASP.NET frameworks often use various patterns, sometimes without explicitly naming them.
* Microsoft may introduce complexity in its frameworks due to best practices, not to confuse developers.

**MVC (Model-View-Controller):**

* MVC is an architectural pattern for designing software.
* It divides software into three layers: presentation, business, and data.
* MVC provides clear demarcation of responsibilities.
* Different types of controllers, including view controller, business controller, data access controllers, etc., exist in MVC.

**Importance of Design Patterns:**

* As a software developer, it's crucial to master design patterns.
* They help address common design issues in coding.
* Reliable sources for learning patterns include Microsoft, IBM, and Oracle websites.

**Creational Patterns:**

* Singleton
* Factory Method
* Abstract Factory
* Builder
* Prototype

**Behavioral Patterns:**

* Command
* Chain of Responsibilities
* Memento
* Mediator
* Template Method
* State
* Strategy
* Visitor
* Iterator
* Interpreter

**Additional Pattern:**

* Dependency Injection (DI) / Inversion of Control (IOC)
  + Dependency Injection and Inversion of Control are the same concept.

**Dependency Injection (DI):**

* DI means that external resources or dependencies are provided to a component rather than the component creating them itself.
* It allows someone else, typically the container, to inject necessary resources at runtime.
* DI is a way of achieving Inversion of Control (IoC).

**Inversion of Control (IoC):**

* IoC means that external control is applied to a component.
* In the context of web applications like ASP.NET:
  + The web server (e.g., IIS) serves as the container.
  + The ASP.NET container controls the execution of your code.
  + Dependencies and resources are injected by the container.
* IoC is essential in web applications because they run within containers that restrict permissions and provide web-related services.
* Your code runs within the constraints set by the container, and you must adhere to these limitations.

**ASP.NET Container:**

* ASP.NET applications are class libraries and cannot run independently.
* They are hosted inside a web container (e.g., IIS), which controls their execution.
* The container handles HTTP requests, IP addresses, ports, and the HTTP protocol.
* ASP.NET containers use Dependency Injection extensively for managing resources.
* To perform actions in a web application, you need to work within the constraints of the container.
* Code running inside the container is subject to the permissions and services provided by the container.
* Your code may need to adapt to the limitations and services offered by the container.

In summary, Dependency Injection and Inversion of Control are crucial concepts in web applications like ASP.NET. These concepts allow external control and resource management within a container, which imposes specific restrictions and provides web-related services. Understanding and working within the constraints of the container is essential for web application development.

**HTML Elements and Markup Language:**

* HTML stands for Hypertext Markup Language.
* HTML elements are enclosed in angle brackets and may have attributes.
* Elements have names and attributes contain properties of the element.
* There are about 110 commonly used HTML elements.

**Learning HTML Resources:**

* A recommended resource for learning HTML is Mozilla Developer Network (MDN) at [developer.mozilla.org](https://developer.mozilla.org/).
* Mozilla provides comprehensive HTML documentation and resources.

**Well-Formed HTML Documents:**

* Well-formed HTML documents adhere to certain rules:
  + Open elements should be closed.
  + Parent elements should not close before child elements.
* HTML documents consist of a header, body, and optional attachments.

**HTML Header:**

* The header in an HTML document is exclusively for metadata.
* It does not contain content and is used for browser-to-browser and web server-to-web server communication.

**Web Browsers and Standards:**

* There are various web browsers, including IE, Microsoft Edge, Safari, Firefox, Google Chrome, and others.
* Some browsers are based on the Chromium project, which defines standards for building browsers.
* Chromium project sets standards for web browsers, and many browsers follow these standards.
* Each web server may have its own standards, but browsers may not always support them.

**HTTP Header Size:**

* The maximum size of HTTP headers used to be 1024 bytes but has now increased to 8190 bytes.
* Different web browsers may have varying support for HTTP header sizes, so it's essential to consider browser limitations.

**HTML Elements and Structure:**

* HTML elements can be categorized into two main types: text-only elements and mixed elements.
* Text-only elements contain only text content and are enclosed in opening and closing tags.
  + <p>This is a text-only element.</p>
* Mixed elements can contain both child elements and text content.
  + <div>This is a mixed element with <a href="#">a link</a>.</div>

**Document Object Model (DOM):**

* The DOM represents the structure of an HTML document as a tree.
* It allows manipulation of HTML elements using JavaScript.
* Elements can be accessed, modified, or removed programmatically.

**HTTP Methods (GET and POST):**

* HTTP methods define how data is sent between a browser and a server.
* **GET** is the default method, typically used for fetching data from the server. It appends data to the URL.
* **POST** is used to send data to the server in the request body.

**HTML Form and POST Request:**

* HTML forms are used to collect user input.
* Use the **method** attribute to specify the HTTP method (**GET** or **POST**).

<form method="POST" action="/submit">   
    <!-- Form inputs here -->   
    <input type="text" name="username" required>   
    <input type="password" name="password">   
    <input type="submit" value="Submit">   
</form> 

**Code Behind in ASP.NET:**

* In ASP.NET, code-behind is C# code that runs on the server-side.
* It processes user requests, performs actions, and generates responses.
* Each ASPX page has a corresponding code-behind file.
* Example of a simple ASPX page:

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="MyPage.aspx.cs" Inherits="MyWebApp.MyPage" %>   
   
<html>   
<!-- HTML content here -->   
</html> 

**ASP.NET Model Page:**

* ASP.NET pages are typically referred to as model pages.
* The model page is responsible for handling requests and generating responses.
* It can have methods like **OnGet** and **OnPost** to handle GET and POST requests, respectively.

**Layout in ASP.NET:**

* Layout plays a crucial role in organizing web pages in ASP.NET.
* The layout is often defined in a shared layout file (usually a .cshtml file), which contains the common structure for all pages.
* The layout typically includes elements like headers and footers, which are shared across multiple pages.

**CSS Styling:**

* CSS (Cascading Style Sheets) is used to style the appearance of web pages.
* In ASP.NET, CSS files are often linked to the layout or individual pages.
* CSS classes and selectors are used to define styling rules, such as centering text or specifying sizes.

**ASP.NET Project Structure:**

* ASP.NET projects are organized with a clear separation of concerns.
* Views (usually .cshtml files) handle the presentation layer.
* Code-behind files (usually .cs files) contain C# code that handles server-side logic.

**UI Focus in ASP.NET:**

* The lecture mentions that the primary focus of this program is not on the user interface (UI) but rather on the code-behind files.
* While UI design is important, this program will emphasize backend development.

**ASP.NET MVC (Model-View-Controller):**

* ASP.NET MVC is a design pattern that separates the application into three main components: Model, View, and Controller.
* The lecture indicates that the next session will involve creating a new ASP.NET MVC project.

**Key Takeaways:**

* Students should be comfortable creating ASP.NET projects.
* Understanding the concept of Views (cshtml) and code-behind files (cs).
* Familiarity with basic elements like labels, text boxes, and buttons in ASP.NET forms.