|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier Naming Convention** | | | |
| Explanation:  Developers need to follow a consistent naming standard throughout a project. | | | |
| **Platform Applicable** | .Net Core, C# | **Version Applicable** | Any |
| **Classification** | Best Practice | **Impact** | High |
| General Guidelines: | | | |
| Guidelines for C# Variables Naming Convention: | | | |
| Example: | | | |
| Reference Url:  <https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/identifier-names> | | | |
|  | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Code Comments** | | | |
| Explanation:  Code comments are a way for developers to add text to source code that is ignored by the compiler or interpreter. Developers can use code comments to explain high-level code intent, provide documentation or describe a bug fix. | | | |
| **Platform Applicable** | C#, JavaScript, HTML | **Version Applicable** | Any |
| **Classification** | Guideline | **Impact** | High |
| General Guidelines: | | | |
| Guidelines for writing comments:  Comments should not duplicate the code.  Comments should provide explanations for nonobvious code.  Comments should clarify, not confuse.  Comments should be brief.  Add comments when fixing bugs.  Use comments to mark incomplete implementations. | | | |
| Example (C#): | | | |
| Example (JavaScript): | | | |
| Example (HTML): | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Manage Data Concurrency** | | | |
| Explanation:  Data concurrency is the ability to allow multiple users to affect multiple transaction within a database. Simply, data concurrency allows multiple users to access data all at the same time. When more than one user manages the same record at the same time, data is changed by one of the users makes the record retrieved concurrently by another user inconsistent. | | | |
| **Platform Applicable** | Entity Framework | **Version Applicable** | Any |
| **Classification** | Guideline | **Impact** | High |
| General Guidelines: During the data entry, certain measures need to be taken by the developer to rectify this situation. In EF Core, Optimistic Concurrency can be utilized which assumes conflicts are rare and makes the data modification to fail if the data has been changed since it was retrieved. Concurrency failure can be reported to the application and appropriate decision can be made inside the code. | | | |
| Guidelines to handle concurrency conflict:  Model:   1. Add a property of type byte[] to manage the Version of each record on the table. 2. Add Timestamp Attribute to the property 3. Assign timestamp data type to TypeName parameter of Column attribute   Page:   1. Add hidden field to bind Version Field in data entry page   Implementation:   1. Catch DbUpdateConcurrencyException during Save Operation. 2. Return Save Failed with Concurrency Error indicator in the Save Method   Error State:   1. Show Error Notification on the Entry Page 2. Stay on the Entry Page | | | |
| Example:  try  {  await \_context.SaveChangesAsync();  }  catch (DbUpdateConcurrencyException ex)  {  saveResult = 5;  }  catch  {  saveResult = 3;  }  return saveResult; | | | |
| Additional Reference: Refer to Notification Guidelines for Explanation and Examples on Notifications for Save Operations | | | |
| Reference Url:  <https://learn.microsoft.com/en-us/ef/core/saving/concurrency?tabs=data-annotations> | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **File Scoped Namespace** | | | |
| Explanation:  File scoped namespaces use a less verbose format for the typical case of files containing only one namespace. File scoped namespace offers an alternative way to organize code within a file without the need for explicit namespace declarations. Instead, the namespace can be defined at the file level, meaning that all code elements within the file will be automatically enclosed within that namespace. | | | |
| **Platform Applicable** | C# | **Version Applicable** | 10.0+ |
| **Classification** | Standard | **Impact** | Low |
| Benefits of File-Scoped Namespace Declaration:   1. Enhanced Readability and Conciseness: By declaring the namespace at the file level, the code becomes more concise and easier to read. Developers no longer need to repeat the namespace declaration for each class within the file, reducing visual clutter and making the codebase more straightforward. 2. Reduced Code Maintenance: With fewer lines of code to manage, developers can significantly reduce the likelihood of errors caused by mismatched or forgotten namespace declarations. This, in turn, simplifies code maintenance and refactoring tasks. 3. Improved Code Navigation: File-Scoped Namespace Declaration streamlines code navigation, making it easier to locate specific code elements within a file. Developers can quickly identify the namespace associated with a particular class without scrolling through the entire file. 4. Encourages Logical Grouping: The new feature encourages developers to organize code files based on logical groupings rather than cramming multiple classes into a single file. This practice aligns with the Single Responsibility Principle, contributing to better code architecture. | | | |
| Usage Considerations:  While File-Scoped Namespace Declaration offers several advantages, developers should consider some important points before adopting this feature:   1. Compatibility: File-Scoped Namespace Declaration is available starting from C# 10. Ensure that your project is using the appropriate language version to use this feature. 2. The transition from Legacy Code: If you are working on a legacy codebase, adopting a File-Scoped Namespace Declaration might require refactoring existing code to eliminate explicit namespace declarations in each file. 3. Collaboration and Code Style: Ensure that the entire development team agrees on using File-Scoped Namespace Declaration to maintain code consistency across the project. | | | |
| Usage:  The semantics are that using the namespace X.Y.Z; form is equivalent to writing namespace X.Y.Z { ... } where the remainder of the file following the file-scoped namespace is in the ... section of a standard namespace declaration. | | | |
| Convert existing namespaces to File Scoped:  Go to Tools -> Options -> Text Editor -> Code Cleanup -> Configure Code Cleanup. Then add the 'Apply namespace preferences'. Then go to Analyze -> Code Cleanup (or just search for 'Code cleanup') and run the Code Cleanup to automatically change the namespaces to file-scoped. | | | |
| Reference Url:  <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/proposals/csharp-10.0/file-scoped-namespaces> | | | |