Standards

- Coding Standards
- Pull Request Checklist
- ADA Compliance Checklist

Coding Standards

Coding Rules

1. Thou shalt not commit to the master branch, pull requests only.
2. Thou shalt not use magic values.
3. Thou shalt not use static classes save for utilities.
4. Thou shalt not write recursive methods without an exit case.
5. Thou shalt spell things correctly.
6. Thou shalt modularize all thine code.
7. Thou shalt keep thine methods short.
8. Thou shalt name thine variables with a descriptive name.
9. Thou shalt name thine methods with a name that describes what the method does.
10. Thou shalt think of the future while coding the present.
11. Thou shalt make thy code self-documenting.

Please read: https://github.com/dotnet/corefx/blob/master/Documentation/coding-guidelines/coding-style.md

And: https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/

Javascript Styling Guidelines

1. With dependency injections, (think AngularJS controllers) the expected pattern is to have the item name declared at the top and then use new lines after each dependency declaration.

Clean Code Article

Clean Code.pdf

Pull Request Checklist

General Items

1. Code compiles
2. Code has been developer-tested
3. Code is tidy (indentation, line length, no commented-out code, no spelling mistakes, etc)
4. Code adheres to the Coding Styles and Standards Guidelines (Found in the documentation folder)
5. Exceptions have been used properly
6. New functionality is appropriately logged as needed
7. Unused using's have been removed
8. Eliminated warnings
9. Ensured NullReference Exceptions caught or handled appropriately
10. Leftover stubs and test routines have been removed
11. Hard-coded or development only things have been removed
12. Considered performance and scalability
13. Considered security, and potential exploits
14. All resources (HTTP connections, DB connection, files, etc) properly released in all code exits (i.e. normal and exception)
15. Corner cases and workarounds for known limitations of dependencies are well documented
16. No new code is a repeat of existing functionality that can be safely reused
17. Thread safety and possible deadlocks are considered and handled accordingly
18. Method bodies are kept as close to 4 lines or less if possible
19. Method names must be descriptive (reveal their intention aka what they do)
20. Methods do one thing only
21. Asynchronous Methods have the post-fix of "Async" appended to the end of their names.

Utopia Specific Checklist Items

1. All code added to UtopiaSharedClasses is safe to expose to customers (No sensitive information)
2. Libraries or projects that could become Libraries should be stored at the top level of the Utopia Repository
3. Public API endpoints have documentation comments if needed

General

1. .NET standard Coding Guidelines: https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/
2. Clean Code Book with corresponding guidelines are found in the Utopia Repository under Documentation.

Code Reviewer's Checklist

1. Types have been generalized where possible
2. Parameterized types have been used appropriately
3. Exceptions have been used appropriately
4. Repetitive code has been factored out
5. Frameworks have been used appropriately – methods have all been defined appropriately
6. Classes have been designed to only have one purpose
7. Views do not contain business logic
8. Common errors have been checked for
9. Potential threading issues have been eliminated where possible
10. Any security concerns have been addressed
11. Performance and Scalability was considered
12. The functionality fits the current design/architecture
13. The code does not use unjustifiable static methods/blocks
14. The code complies to coding standards
15. Logging used appropriately (proper logging level and details)
16. Exception use is in compliance with proper methodology
17. Code quality is excellent

Utopia Specific Checklist Items

1. All code added to UtopiaSharedClasses is safe to expose to customers (No sensitive information)
2. Libraries or projects that could become Libraries should be stored at the top level of the Utopia Repository
3. Public API endpoints have documentation comments if needed
4. Functionality should be exposed at the highest layer of the architecture as possible and only moved down when reuse becomes necessary. (i.e. CSV generation at the presentation layer, but could be moved into business logic if reuse becomes necessary)

ADA Compliance Checklist

This is not meant to be a comprehensive list but is hopefully a good starting place. Feel free to add.

To see the WCAG standards established by W3C go here

- Form fields should have a visible label, not just a placeholder
- Field formatting information should stay visible after beginning to edit
- Test page text with Text Only setting up to 200%
- Color contrast should meet WCAG standard
- Don’t rely on color to convey meaning, provide another way for the information to be understood
- All page content MUST be keyboard accessible
    - Confirm that the tabbing order makes sense for the user experience
    - Ensure that the user won't get stuck in an infinite tabbing loop
- Header tags should not be used for styling alone. They should add meaning to the page hierarchy. Add styling to get different text sizes and styling
- Use provided HTML semantic tags whenever possible. This makes it much easier for screen readers
- Avoid using small targets (i.e. be sure that the text associated with a checkbox selects that checkbox)
- Not everyone can use drag and drop or similar components. Be sure to provide an alternative way to modify the information
- Make sure all page actions are visible. (i.e. don’t rely on a mouse hover to show that an image is a carousel)
- Page titles should go from specific to general information. They should help the user know what the page is about, not just what the website is.
- Audio and visual content that conveys meaning must have a text alternative. This could be an alt tag description, closed captioning, a transcript, etc.
    - Media that has meaning that is already conveyed can have a blank alt tag
- Look at content proximity, both for auditory content and visually. Related information should be close together. Those who need to zoom in may not be able to see a modal or submit button that is on another part of the page
- Don’t use auto-advancing fields. They cause issues for accessibility
- Use plain language and provide full versions of acronyms

For a more comprehensive checklist from WebAIM go here