

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

The following is a collection of standards and practices to be observed while developing/enhancing/repairing Waukesha County (WC) web applications. It is expected that development be constrained to the following technologies:

XHTML	ASP/VB Script
JavaScript	ASP.NET/VB.NET
Cascading Style Sheets (CSS)	ADO/ADO.NET
XML	SQL Server

Standards and practices specific to these technologies are listed in sections of the same name below.

This document is organized in the following manner:

- Development Process
- Target Browsers and Industry Standards Compliance
- Naming and Folder Structure
- Comments
- General Coding Practices
- XHTML
- JavaScript
- Cascading Style Sheets (CSS)
- XML
- ASP/VB Script
- ASP.NET/VB.NET
- ADO/ADO.NET
- SQL Server

Practices to be strictly applied are prefixed with “REQ:” (required). Practices to be followed when convenient are prefixed with “Sug:” (suggested).

Development Process

1. REQ: Development code will be controlled by Visual Source Safe. Any change to any file must be saved in source control at the end of each day.
2. REQ: Development may occur on a local PC or on a development server.
3. REQ: When promoting to the test environment, the entire set of project/application files, less configuration file, is copied to the new environment. The same is true for promotions to production.
4. REQ: When the application or database is placed in a development environment, add the suffix `_DEV` to the name of the root folder or database, as `ROD_TaxListing_Dev`. In the test environment, use `_TST`, and in the production environ, use `_PRD`.
5. Sug: The test environment is to be used only for second-party testing.
6. See the Naming section for name requirements for the different environments.

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

Target Browsers and Industry Standards Compliance

Try to design to W3C standards for XHTML, CSS, and WCAG level A.

1. REQ: Design all web pages to work acceptably on our specified browser of choice (currently IE 7 and 6) and the most recent version of the other most popular browser (currently Firefox).
2. REQ: Specify document type definition on each web page: `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">` . Observe XHTML coding standards.
3. *Sug*: Design for W3C Web Content Accessibility Guidelines (WCAG) Level A compliance. Level A is W3C’s minimum accessibility compliance rating.
 - a. A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).
 - b. Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.
 - c. Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.
 - d. Documents shall be organized so they are readable without requiring an associated style sheet.
 - e. Redundant text links shall be provided for each active region of a server-side image map.
 - f. Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.
 - g. Row and column headers shall be identified for data tables.
 - h. Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.
 - i. Frames shall be titled with text that facilitates frame identification and navigation.
 - j. Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.
 - k. A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.

Naming and Folder Structure

One of the best aids to understanding an application is through the wording used for project, file, variable, and method names.

1. REQ: For project names and application root folder names, use the format like **ROD_TaxListing**, where an abbreviation is used to indicate the department/division of the sponsor of the project, and the rest of the name provides a unique functional description. Note the upper case for the dept/div, the single underscore, and the Pascal casing of the project description.

County Board	CB	Info Technology	IT or IS
Corporate Counsel	CC	Land Info Systems	LIS

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

Circuit Court System	CCS	Parks and Land Use	PLU
County Clerk	CL	Records Management	RM
Dept. of Admin	DOA	Register of Deeds	ROD
Dept. of Public Works	DPW	Waukesha County	WC
Health & Human Services	HHS	Waukesha Communications Center	WCC
Human Resources	HR		

2. REQ: Databases should be named with upper case characters, and underscore characters used to prevent blank spaces. The database name structure is “sponsor_name_environment”, like “ROD_TAX_LISTING_PRD”. Environment suffixes of “_DEV” and “_TST” should be used where appropriate.
3. *Sug*: Where a database corresponds to an application, the name of the database should parallel the application. Application: ROD_TaxListing_PRD, database: ROD_TAX_LISTING_PRD
4. REQ: No spaces in URL’s. Hence, no spaces in file or folder names.
5. REQ: Place all image, video, and sound files in a /media folder.
6. REQ: For files that serve as auxiliary files to main or GUI files, precede name with abbreviation of usage type, as incHeader. This serves to identify auxiliary files, and group them according to type in an alphabetically ordered listing. This is not necessary for files with different extension names, as .css and .js.

Include	inc	User Control	ctl
Class	cls		

7. REQ: For file names, use camel casing (winnerListing.asp) where the first letter of each word except the first is capitalized. Constants should be all uppercase with underscores between words, such as NUM_DAYS_IN_WEEK. For folders, properties, and methods, use Pascal casing (CalculateInvoiceTotal) where the first letter of each word is capitalized.
8. REQ: For variable names, precede name with abbreviation of variable type (Hungarian notation). This is especially useful to discern the scalar variables (strUserName) from form components (txtUserName).

Integer	int	Guid	guid	Textbox	txt
String	str	Array	ary	Label	lbl
Boolean	blnl	Dataset	ds	Button	btn
DateTime	dt	Table	tbl	Listbox	lst
Byte	byt			Panel	pnl
Long	lng	Object	obj or o		
Double	dbl				

9. REQ: A name should tell “who” or “what” rather than “how.” For example, you could use “aryStudents” instead of “array1”. Don’t be reluctant to use long names, like “SortStudentArrayForAdmissionTickets()”.
10. REQ: Do not use vague/ambiguous/generic words. Searches for instances of words like “user” or “edit” yields many useless hits. Instead, use names like ‘EditStudentRecord’ that will more likely be unique among all names.

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

11. REQ: Use the verb-noun method for naming routines that perform some operation, such as CalculateInvoiceTotal(). When naming functions, include a description of the value being returned, such as GetCurrentStudentName(). Boolean variables and Boolean functions should contain “Is” which implies Yes/No or True/False values, such as blnFileIsFound.
12. REQ: Use single-letter variable names, such as i, or j, for short-loop indexes only.
13. REQ: Include notation that indicates the scope of the variable, such as prefixing a g for global variables and m for module-level variables. For example, gstrUserName.
14. *Sug*: In object-oriented languages, it is redundant to include class names in the name of class properties, such as Book.BookTitle. Instead, use Book.Title.
15. *Sug*: Use customary opposite pairs in variable names, such as min/max, begin/end, and open/close. Also, append computation qualifiers (Avg, Sum, Min, Max, Index) to the end of a variable name where appropriate.

Comments

Sufficient comments should be included in source code to allow any developer to understand the logic of the code without referring to external documentation. Strive to be clear and concise.

1. REQ: At the top of each file, include a description of the purpose of the file and a modification log. At the top of each method, include a description of the purpose of the method. For variable declarations, include an in-line comment describing the purpose of the variable if not made clear by the name of the variable.
2. REQ: Use comments for loops and logic branches. These are key logic areas.
3. REQ: Separate comments from comment delimiters with one space—easier to see where developer cannot see color-coding. Do not surrounding a block comment with a typographical frame—use white space instead.

General Coding Practices

1. REQ: Source code should reflect a harmonized style, as if a single developer wrote the code in one session. Harmonized style and repeated techniques are easiest to read.
2. REQ: Limit source code line length to 80 characters. When a code statement is broken across several lines, indent the subsequent lines. Makes for easier screen viewing and cleaner hardcopies.
3. REQ: Sections of code should be vertically delimited by blank lines. For example, methods should have two or three blank lines separating them from surrounding code. Loop structures should have one blank line separating them from surrounding code.
4. REQ: Indent code structures in two-space increments. Where braces are used to demarcate code structures, align the opening brace with the closing.

<pre> For Each item In collection A = B If A = TRUE Then C = D End If Next </pre>	<pre> for (i = 0; i < 100; i++) { A = B if (A == true) { C = D } } </pre>
---	--

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

5. REQ: Use a space after each comma in comma-delimited lists, such as array values and arguments, when doing so does not alter the intent of the code.
6. REQ: Use spaces before and after most operators. An exception is in HTML, where attributes (class="tableHead") are easier to read with no spaces around the operator.
7. REQ: Declare local variables just prior to where they are used, not necessarily at the top of the code as some advise. Localized declaration provides a better indication of where and how the variable is used, and makes it more likely the variable will be removed when its associated code is removed.
8. *Sug*: Keep the scope of variables as small as possible. Instantiate object instances as late as possible, and *explicitly* destroy object instances as soon as possible.
9. *Sug*: Use variables and routines for one and only one purpose.
10. REQ: Refer to database fields by field name; do not reference fields by their ordinal placement. Reference by placement is not as readable, and is prone to break.
11. *Sug*: Use Select Case or Switch statements in lieu of repetitive checking of a common variable using If...Then statements.
12. *Sug*: Critical program errors should be reported to an error log or the Windows Application Event viewer.
13. *Sug*: Remove all unused code prior to deployment. Rely on back-up copies of the source code to store code that "someone might use some day". At the least, move unused code to the end of the file so that it can be out-of sight/out-of mind.
14. REQ: If a password will be stored into a data table, and stronger encryption techniques are not specified, it will be encrypted using the MD5 hash methodology.

XHTML

1. REQ: Observe XHTML rules. Use lower case for all tags. Use lower case for attribute and attribute values. Use double-quotes for attribute values where possible; otherwise, use single-quotes. Always provide closing HTML tags. For self-closing tags, place the slash at the end of the tag (
).

JavaScript

1. *Sug*: Minimize the usage of JavaScript as much as possible.

Cascading Style Sheets (CSS)

1. *Sug*: NO styling should occur in-line, all styling should occur in the linked .CSS file.

XML

1. *Sug*: XML File formats should be used for any transfer of data from one system to another.

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

ASP/VB Script

1. REQ: Make use of a single, global configuration module (config.asp) to declare and define global parameters, or parameters that are likely to be modified over the life of the code.
2. REQ: Use Option Explicit to minimize errors resulting from typographical errors.
3. REQ: Any created objects should be removed from memory when no longer used, for example “Set objName = nothing”.
4. REQ: Use script delimiters (<% %>) around blocks of script rather than around each line of script. Using script delimiters around each line or interspersing HTML fragments with server-side scripting increases the frequency of context switching on the server side, which hampers performance and readability.

ASP.NET/VB.NET

1. REQ: Set project OPTION EXPLICIT and OPTION STRICT
2. REQ: When writing classes, set fields/attributes for Private use only, then use Public or Protected get/set properties to expose fields needed by external code. This provides an opportunity to validate value changes, and indicates to other developers the scope of use of the field value.
3. REQ: The format for .vb files (class and code-behind) is demonstrated below

<ol style="list-style-type: none"> a. Description of file purpose. b. blank line c. Modification log d. blank line e. Compilation options (Strict and Explicit are required) f. blank Line g. Import declarations h. blank Line i. Class definition header j. blank line k. Field declarations l. blank line m. Property definitions with blank line between them. n. 2-3 blank lines o. Method definitions with 2-3 blank lines between them. p. blank line q. Class definition footer 	<pre> ' This is a sample class file that demonstrates required standar ' elements as well as desired formatting to be used by Waukesha ' developers. ' ' Modification log ' 10/5/07 jvm Built as new for standards document. Option Strict On Option Explicit On Imports System.text.RegularExpressions Imports System.Configuration.ConfigurationSettings Public Class clsSample Private m_intDayCount As Integer Private m_strChosenDay As String Public Property ChosenDay() As String Get ChosenDay = m_strChosenDay End Get Set(ByVal Value As String) m_strChosenDay = Value End Set End Property Public Sub New() m_intDayCount = 0 m_strChosenDay = "Monday" End Sub Public Sub TipToeThroughTheTulips() ' description of what this is all about, and any tricks used End Sub Public Function TeaForTwo() As String ' description of what this is all about, and any tricks used End Function End Class </pre>
--	---

4. REQ: Only one class definition should be included in each file. For class definition files, the file name should parallel the name of the class. For the class definition above, the file name should be clsSample.vb.

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

5. REQ: Disable Debug Mode in web.config before deploying an ASP.NET application to test or production.
6. *Sug*: Use early binding techniques whenever possible.
7. *Sug*: Do not use Exceptions for common input validation or messaging. Exception handling is resource intensive.
8. REQ: When project namespace becomes an issue, for example, when a project needs a strong name to be placed in the Global Assembly Cache (GAC) or when a dll is being generated to use in other applications, use the name **us.wi.Waukesha.co.doa.appname**. For example, us.wi.Waukesha.co.doa.ROD_TaxListing where the name of the application is ROD_TaxListing.

ADO/ADO.NET

1. REQ: To realize the benefit of database connection pooling, use a single database service account for all of the application's users.

SQL Server

1. REQ: Each non-log or non-backup table should include the following fields:
 - PKId. Primary Key Identifier (guid or auto number).
 - updateBy. Username of user that modified the record, set "unknown" as default value.
 - updateDate. Date-Time record was modified, set getdate() as default.
 to implement auditing. Additionally, tables that have records that will be programmatically modified must have the field
 - optLockCnt. Version counter, 0 by default and incremented on each update. to guard against dirty-data updates.
2. REQ: Applications will always need to service at least the updateBy fields to provide username information (unless Windows Integrated Authentication is used—then a trigger will do).
3. *Sug*: For thorough auditing, each table can have an insert/modify/delete trigger that copies each version of the primary table's records to a companion table (named primarytablename_AUDIT). Code like this can service the optLockCnt and updateDate fields of the primary table as well as perform the auditing function:

```

CREATE TRIGGER AppUser_auditAll
ON dbo.AppUser
AFTER INSERT, UPDATE, DELETE
AS

/*
Universal audit trigger for SQL Server. Logs changes of primary table
to companion table, services optLockCnt and updateDate fields. Assumes
1. Table has primary key field named 'PKId'
2. Table has audit field named OptLockCnt
3. Table has audit field named updateDate
4. Table has audit field named updateBy
5. Table has companion audit table of same name w/ _AUDIT suffix.
```

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

6. Audit table has same structure as primary table, with additional column named dbUserAcct (varchar(64)) added to end of column list.
7. Audit table allows NULL for all fields except PKid, updateDate, updateBy

AFTER Trigger tables (INSERTED and DELETED) cannot handle text, ntext, img data type columns. Audit bypasses these tables to get rows from primary table.

*/

```

DECLARE @errMsg varchar
SET @errMsg = ''
DECLARE @icount int
SET @icount = (SELECT count(*) FROM INSERTED)
DECLARE @dcount int
SET @dcount = (SELECT count(*) FROM DELETED)

IF @icount > 0 AND @dcount = 0      -- an insert operation
BEGIN
    INSERT INTO AppUser_AUDIT
        SELECT * , suser_sname()
        FROM AppUser
        WHERE PKid IN (SELECT PKid FROM INSERTED)
    IF @@ERROR <> 0 OR @@ROWCOUNT = 0
        SET @errMsg = 'Failed insert - audit insert.'
END

IF @icount > 0 AND @dcount > 0    -- an update operation
BEGIN
    SELECT DELETED.optLockCnt, INSERTED.optLockCnt
    FROM DELETED INNER JOIN INSERTED
    ON DELETED.PKId = INSERTED.PKId
    WHERE (DELETED.optLockCnt) <> INSERTED.optLockCnt
    if @@ROWCOUNT <> 0
        SET @errMsg = 'Failed update - dirty data.'
    ELSE
    BEGIN
        UPDATE AppUser
        SET optLockCnt = optLockCnt + 1, updateDate = getdate()
        WHERE PKid IN (SELECT PKid FROM INSERTED)
        if @@ERROR <> 0 OR @@ROWCOUNT = 0
            SET @errMsg = 'Failed update - lock handler.'
        ELSE
        BEGIN
            INSERT INTO AppUser_AUDIT
                SELECT * , suser_sname()
                FROM AppUser
                WHERE PKid IN (SELECT PKid FROM INSERTED)
            if @@ERROR <> 0 OR @@ROWCOUNT = 0
                SET @errMsg = 'Failed update - audit insert.'
        END
    END
END
END

```

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

```

IF @icount = 0 AND @dcount > 0      -- a delete operation.
--Does not capture delete username.  This must be captured by an application
--or stored proc that performs update prior to delete. Also, does not attempt
--to log data.  See prior update for data state prior to delete.
BEGIN
    INSERT INTO AppUser_AUDIT
    (PKId, optLockCnt, updateDate, updateBy, dbUserAcct)
    SELECT PKID, optLockCnt, getdate(), updateBy, suser_sname()
    FROM DELETED

    IF @@ERROR <> 0 OR @@ROWCOUNT = 0
        SET @errMsg = 'Failed delete - audit insert.'
END
IF @errMsg <> ''
BEGIN
    ROLLBACK
    RAISERROR(@errMsg, 16,1)
END

```

4. REQ: If a password will be stored into a data table, and stronger encryption techniques are not specified, it will be encrypted using the MD5 hash methodology.
5. *Sug*: Use stored procedures in lieu of views or ad hoc queries to leverage the performance gains they provide. Stored procs can also be nicely documented.
6. REQ: Put each major SQL clause on a separate line so statements are easier to read and edit:

```

SELECT FirstName, LastName
FROM Customer
INNER JOIN Order
ON Customer.PKId = Order.FK_Customer
WHERE State = 'WA'

```

7. *Sug*: Use upper case for clause statements, lowercase for SQL functions, and Pascal case (each word begins with a capital letter, no spaces) for table names and column names.
8. *Sug*: When naming tables, express the name in the singular form. For example, use Employee instead of Employees.
9. *Sug*: When naming columns of tables, do not repeat the table name; for example, avoid having a field called EmployeeLastName in a table called Employee.
10. REQ: Do not prefix stored procedures with sp_, because this prefix is reserved for identifying system-stored procedures. Use the prefix “sp”, as in spGetData.
11. *Sug*: Minimize the scope and duration of transactions.
12. *Sug*: Use RETURN statements in stored procedures to help the calling program know whether the procedure worked properly.
13. *Sug*: Avoid SELECT *. Be explicit in which columns to retrieve and retrieve only the columns that are required.
14. *Sug*: When using LIKE, avoid using a wildcard character at the start of the match value because SQL Server will not be able to use the indexes.

Waukesha County Government – Information Technology Division Web Application Development Standards

Date: June 14, 2010

Version: 1.0

15. *Sug:* Use WITH RECOMPILE in CREATE PROC when a wide variety of arguments are passed, because the plan stored for the procedure might not be optimal for a given set of parameters.
16. *Sug:* After each data modification statement inside a transaction, check for an error by testing the global variable @@ERROR.
17. *Sug:* Use uncorrelated subqueries instead of correlated subqueries. Uncorrelated subqueries are those where the inner SELECT statement does not rely on the outer SELECT statement for information. In uncorrelated subqueries, the inner query is run once instead of being run for each row returned by the outer query.
18. *Sug:* Use TRUNCATE TABLE instead of DELETE to clear all rows in a table. TRUNCATE TABLE does not log the changes in the transaction file, so it is faster and uses less resources (though it requires a security context greater than datawriter).