# Code Review Checklist

Referenced from Microsoft TechNet (<http://technet.microsoft.com/en-us/library/cc707802.aspx>)

## Security

This section of the code acceptance checklist contains suggested items to help ensure that solutions that are submitted for deployment in a SharePoint environment have been developed by using best security practices.

|  |  |
| --- | --- |
| ⃝ | The application uses an inclusion list (known, valid, and safe input) rather than an exclusion list (rejecting known malicious or dangerous input). |
| ⃝ | All user input is encoded with IOSec when displayed to clients. |
| ⃝ | Character encoding is set by the server (ISO-8859-1 is recommended). |
| ⃝ | Plain text passwords are not present in Web.config, Machine.config, or any files that contain configuration settings. Utilities such as Aspnet\_setreg.exe and Trustee or the identity setting in AppPool on IIS 6.0 or IIS 7.0 are used to encrypt credentials. |
| ⃝ | If cookies contain sensitive data, they are marked secure. |
| ⃝ | Input surfaces in Web parts and other customizations include boundary checks, input data integrity checks, and appropriate exception handling to protect from cross-site scripting and SQL injection. |
| ⃝ | The design addresses potential canonicalization issues. |
| ⃝ | You avoid using AllowUnsafeUpdates. You use ValidateFormDigest() and, if necessary, use elevated privileges to interact with SharePoint objects. In cases where AllowUnsafeUpdates must be used, you ensure that AllowUnsafeUpdates is set to False in your try-catch-finally block, or you use a Dispose() method (as required by the IDisposable interface) to avoid security issues. |

## Session Management

This section of the code acceptance checklist contains suggested items to help ensure that solutions that are submitted for deployment in your SharePoint environment have been developed by using best practices for managing sessions.

|  |  |
| --- | --- |
| ⃝ | Session state is strong, unpredictable, and protected from unauthorized access or replay attacks. |
| ⃝ | Session lifetime is limited to 30 minutes maximum of inactivity. |
| ⃝ | Session identifiers are not passed in the URL, and the ASP.NET feature, cookieless session, is not used. |
| ⃝ | The session state service is disabled if not used. |

## Validation

This section of the code acceptance checklist contains suggested items to help ensure that solutions that are submitted for deployment in your SharePoint environment have been developed by using best practices for validating input.

|  |  |
| --- | --- |
| ⃝ | Input validation is applied at all identified entry points (including form fields, querystrings, cookies, HTTP headers, and Web service parameters). |
| ⃝ | The ASP.NET validateRequest option is enabled, if possible. |
| ⃝ | Data is validated for type, length, format, and range. |
| ⃝ | Security does not rely on client-side validation. Instead, validation is performed on the server side. |
| ⃝ | The application consistently uses standardized input validation such as RegEx throughout. |

## Sensitive Data

This section of the code acceptance checklist contains suggested items to help ensure that solutions that are submitted for deployment in your SharePoint environment have been developed by using best practices for protecting sensitive data.

|  |  |
| --- | --- |
| ⃝ | The application does not log sensitive data in clear text. |
| ⃝ | Sensitive data is not stored in cookies. |
| ⃝ | Sensitive data is not stored in unencrypted, hidden form fields or query strings. It is maintained by using server-side state management. |
| ⃝ | SSL, IPSEC with encryption, or application layer encryption prior to transmittal is used to protect sensitive data during transmission. |
| ⃝ | Sensitive data is not cached. Output caching is off by default. |
| ⃝ | Sensitive data that is transferred via e-mail uses S/MIME encryption or Information Rights Management (IRM), depending upon the intended recipient. |

## Exception Handling

This section of the code acceptance checklist contains suggested items to help ensure that solutions that are submitted for deployment in your SharePoint environment have been developed by using best practices for handling exceptions.

|  |  |
| --- | --- |
| ⃝ | The application uses a standardized approach to structured error and exception handling throughout. |
| ⃝ | Error-handling code inherits from the SPException class to maintain a consistent SharePoint look and feel for errors. |
| ⃝ | The application fails securely in the event of error and exceptions. |
| ⃝ | Exception conditions do not allow a user to bypass security checks to run privileged code. |
| ⃝ | The application returns generic custom error messages to the client. |
| ⃝ | The code uses exception handling. The code catches only the exceptions that you know about. For example, do not use try{} catch(Exception ex){} unless you throw the error again. |
| ⃝ | If code uses exception filters, it is not sensitive to filter execution sequence (filter runs before finally block). |
| ⃝ | Application errors do not contain sensitive information or information that could be used to exploit the fault. . |

## Web Parts

This section of the code acceptance checklist contains suggested items to help ensure that solutions that are submitted for deployment in your SharePoint environment have been developed by using best practices for developing Web parts.

|  |  |
| --- | --- |
| ⃝ | Custom Web parts (including resource files) are contained within a SharePoint Feature and are packaged as a SharePoint solution in order to be deployed. |
| ⃝ | The configuration of Web parts that are being deployed gives the administrator the flexibility of deploying to the Web application level or lower. |
| ⃝ | You use the SharePoint Web part infrastructure's standardized set of connection interfaces for Web parts to exchange information with each other at run time. |
| ⃝ | Source code for third party Web parts solutions, whenever possible, is provided with adequate documentation to ensure good technical support. |
| ⃝ | All custom Web parts utilize the SharePoint architecture to ensure consistent behavior across the application for functionality such as single sign-on, feature deployment, and so on. |

## Documentation

You should require adequate documentation to ensure that customizations that you are being asked to deploy are installable, supportable, and well tested. Furthermore, documentation indicates that all errors that are generated by the customizations are properly described and diagnosed. This section of the code acceptance checklist contains suggested items to help ensure that solutions that are submitted for deployment in your SharePoint environment have been developed using best practices for documentation.

|  |  |
| --- | --- |
| ⃝ | Customizations are accompanied by installation instructions that detail how to install and uninstall the package. Architecture diagrams that are related to the installation of the solution are included. If it is not possible to roll back a solution, this must be explained in the installation instructions so that you can discuss the risks and prepare a plan for a system recovery. |
| ⃝ | Customizations are accompanied by test documents and results. |
| ⃝ | Customizations are accompanied by a list of all dependencies. This could include account/passwords, Web services, databases, other solutions or Features, patches, tool sets or libraries, and other dependencies. |
| ⃝ | A list of all event entries that are generated by the customizations and the actions to take are supplied. This can take the form of a table of error codes, where the severity and root cause of each code is supplied. |
| ⃝ | Optionally, source code is provided to expedite validation and testing by the IT organization. |
| ⃝ | Customizations that are an upgrade of previously deployed customizations are accompanied by documentation that describes the changes, considerations in upgrading the customizations, and rollback instructions. |

## General Software Development Best Practices

This section of the code acceptance checklist contains suggested items to help ensure that solutions that are submitted for deployment in your SharePoint environment have been developed by using best practices for software development.

|  |  |
| --- | --- |
| ⃝ | Assemblies have a strong name. (Dynamically generated ASP.NET Web page assemblies cannot currently have a strong name.) |
| ⃝ | You use delay signing as a way to protect and restrict the private key that is used in the strong name and signing process. |
| ⃝ | Assemblies include declarative security attributes (with SecurityAction.RequestMinimum) to specify minimum permission requirements. |
| ⃝ | Highly privileged assemblies are separated from lower privileged assemblies. |
| ⃝ | If an assembly is to be used in a partial-trust environment (for example, it is called from a partial-trust Web application), then privileged code is in a separate assembly. |
| ⃝ | You rely on a native configuration file to support the application instead of changing the configuration to the Web.config. |
| ⃝ | You use .NET Framework 2.0, 3.0, or 3.5. |
| ⃝ | You use a single .NET Framework version. You do not mix multiple versions. |
| ⃝ | Your code is 64 bit compatible. |
| ⃝ | Your application does not try to directly access any SharePoint databases. Data stores in SharePoint databases are only updated by using the SharePoint object model. |
| ⃝ | You avoid hard coding strings and labels. You use resources or language files instead. |
| ⃝ | When referencing the SPWeb or SPSite objects, you employ a using statement or, alternatively, you use an explicit call of the .Dispose method to ensure proper use and disposing of the memory objects. |
| ⃝ | You use caching as appropriate to reduce unnecessary round trips. For Web parts, you expose the cache expiration (duration) as a Web part property. |
| ⃝ | When packaging your solution, you include a Code Access Security policy for the solution and, if necessary, include your assembly in the Safe Controls list though the solution. |
| ⃝ | When logging code, you use the Portal Log class to log the SharePoint Unified Logging Service (ULS) logs. |
| ⃝ | If you need to update multiple list items by using remote code, you use the Web service to update list items. You only use SPListItem.Update() if you have to update more than one item at a time by using local OM-based code. |
| ⃝ | When using the Count property of a SPListItemCollection, you only call it once and then store it in a variable that you can refer to when looping. You do not call it inside a loop. |
| ⃝ | The solution uses the AppSettings object to implement XML mapping. (This can be provided by using the settings persistence framework in .NET 2.0, 3.0, or 3.5.) The solution avoids creating custom XML files and a strongly typed object for XML mapping. |
| ⃝ | Installation and deployment logging are provided in the event logs to enable appropriate operational troubleshooting during installation and uninstallation. |