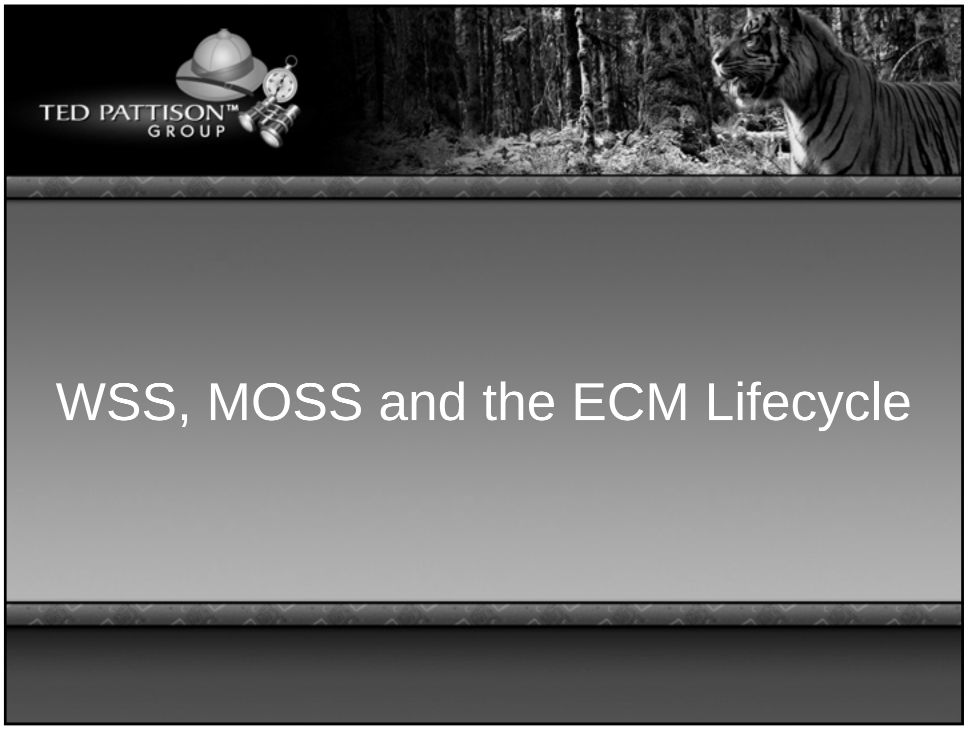
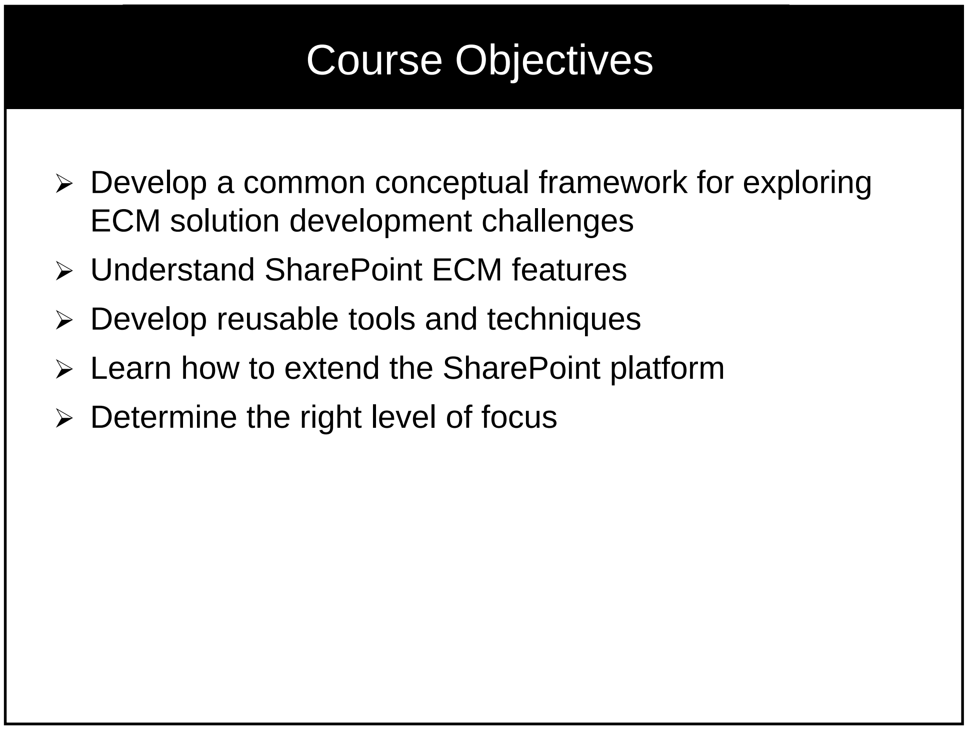
WC-ECM401 Slide Manual

Enterprise Content Management with SharePoint Server 2007

Schedule of Lectures

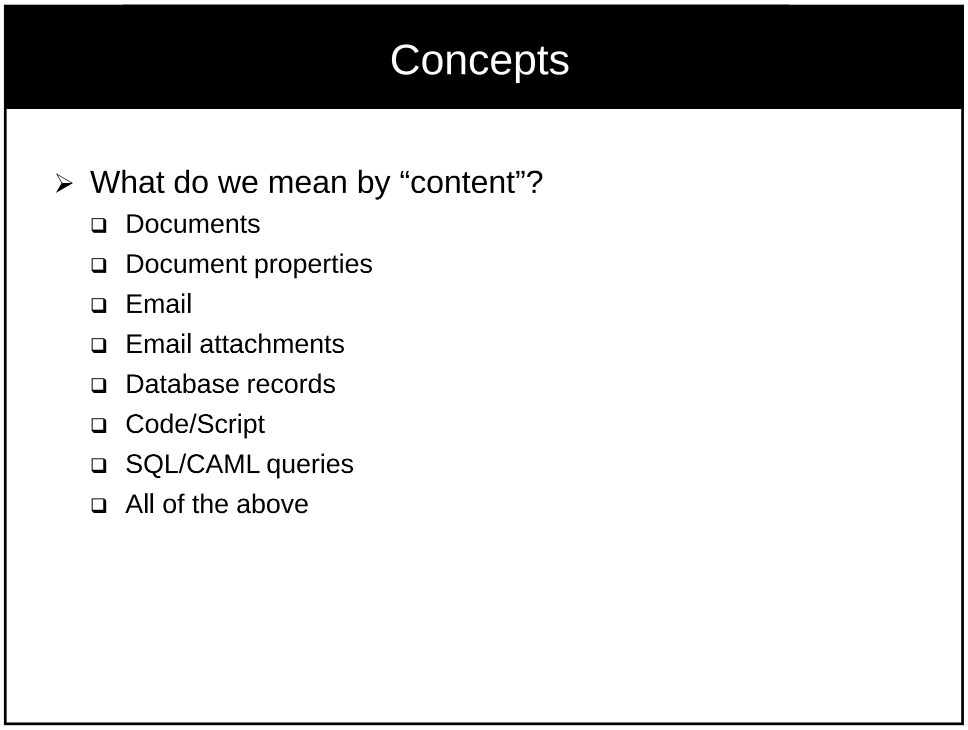


WSS, MOSS and the ECM Lifecycle



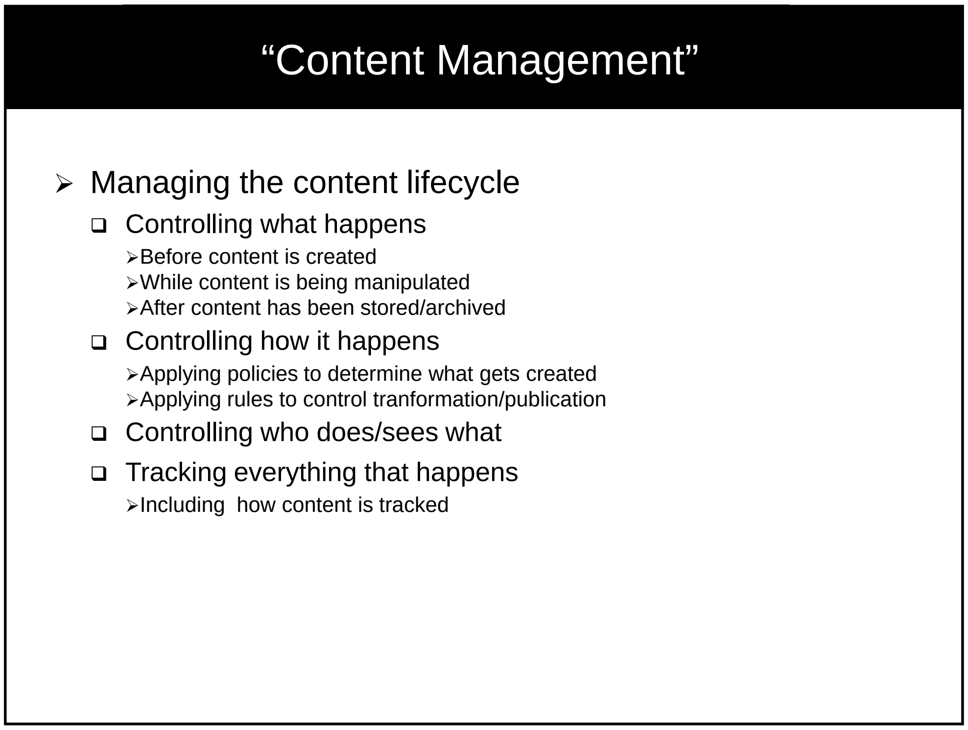
Course Objectives

- Develop a common conceptual framework for exploring ECM solution development challenges
- Understand SharePoint ECM features
- Develop reusable tools and techniques
- Learn how to extend the SharePoint platform
- Determine the right level of focus



Concepts

- What do we mean by “content”?
  - Documents
  - Document properties
  - Email
  - Email attachments
  - Database records
  - Code/Script
  - SQL/CAML queries
  - All of the above



“Content Management”

- Managing the content lifecycle
  - Controlling what happens
    - Before content is created
    - While content is being manipulated
    - After content has been stored/archived
  - Controlling how it happens
    - Applying policies to determine what gets created
    - Applying rules to control tranformation/publication
  - Controlling who does/sees what
  - Tracking everything that happens
    - Including how content is tracked

“Enterprise Content Management”

The Microsoft View:

- Document Management (DM)
  - Driven by day-to-day collaboration
  - Competition from traditional DM vendors
- Records Management (RM)
  - Driven by regulatory compliance
- Web Publishing (WCM)
  - Driven by traditional WCM requirements
  - Competition from simpler WCM solutions
- Electronic Forms
  - Driven by data requirements of DM, RM and WCM
  - Driven by need to integrate with other systems (WF, WCF)

Content Management Core Services

- Defining Metadata
- Tracking and Validating Changes
- Controlling Access
- Defining and Applying Processing Rules

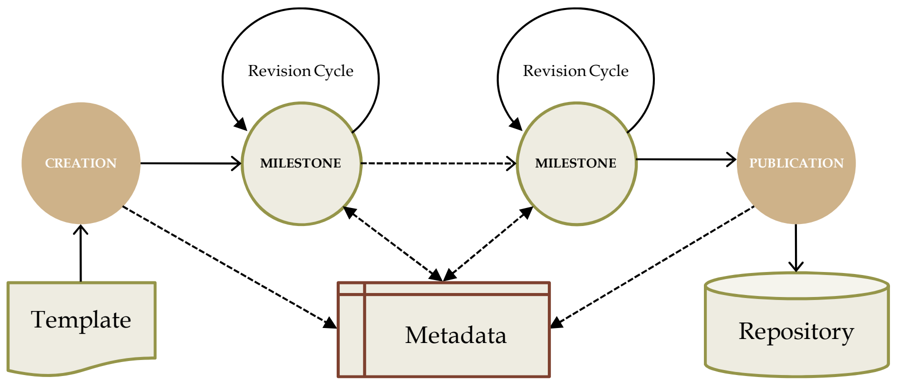
Content Management in SharePoint

- Defining Metadata
  - Document Libraries
  - Content Types
- Tracking and Validating Changes
  - Auditing
  - Event Receivers
- Controlling Access
  - SharePoint Permissions
  - Audiences
  - RMS/IRM
- Defining and Applying Rules
  - Event Receivers on Lists
  - Event Receivers on Content Types
  - Information Policy
  - Workflow

The Content Lifecycle

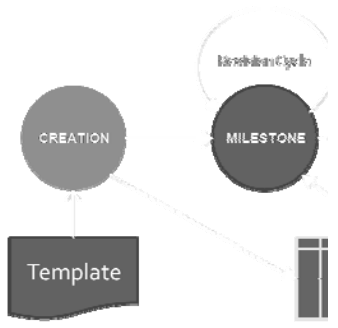
- Creation
- Collaboration/Review
- Approval/Publication
- Archival/Destruction

The Content Lifecycle (driven by metadata)



Content Creation in MOSS

- Primary Components
  - Document Templates (client or server)
    - Setup initial metadata
  - Forms
    - Gather additional metadata
  - Rules
    - Validate metadata
  - Policies
    - Enforce content/data relationships
- Other Considerations
  - Office Client Integration
  - Bulk Upload

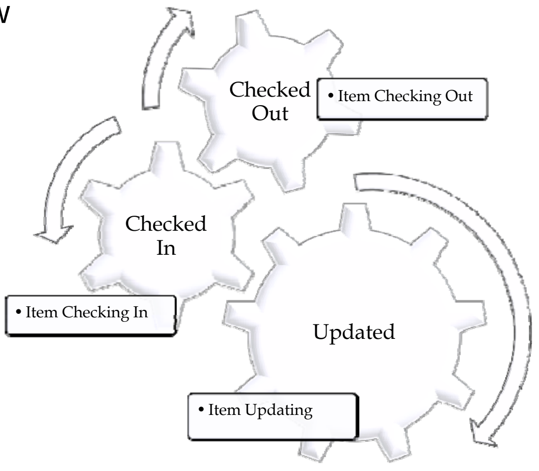


Collaboration/Review

- Check-in/Check-out
  - Supports standard check-in and checkout operations for any list item
  - Prevents concurrent access to items
  - Checkout status can be queried and controls via object model
- Versioning
  - Supports major and minor versioning
  - Used extensively by publishing framework
- Access Control
  - Managed through SharePoint Permissions
  - Fine-grained control possible at item and field level
  - May require additional support from client apps
    - RMS/IRM

Managing Revision Cycles

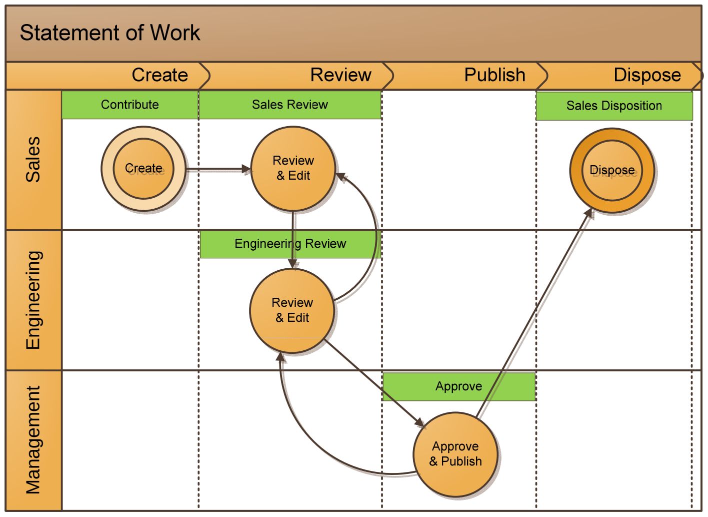
- Event receivers allow "pretty good" encapsulation of document revision behavior
- Content types allow tuning of behavior to document metadata
- Workflow enables delegation of control to long-running processes

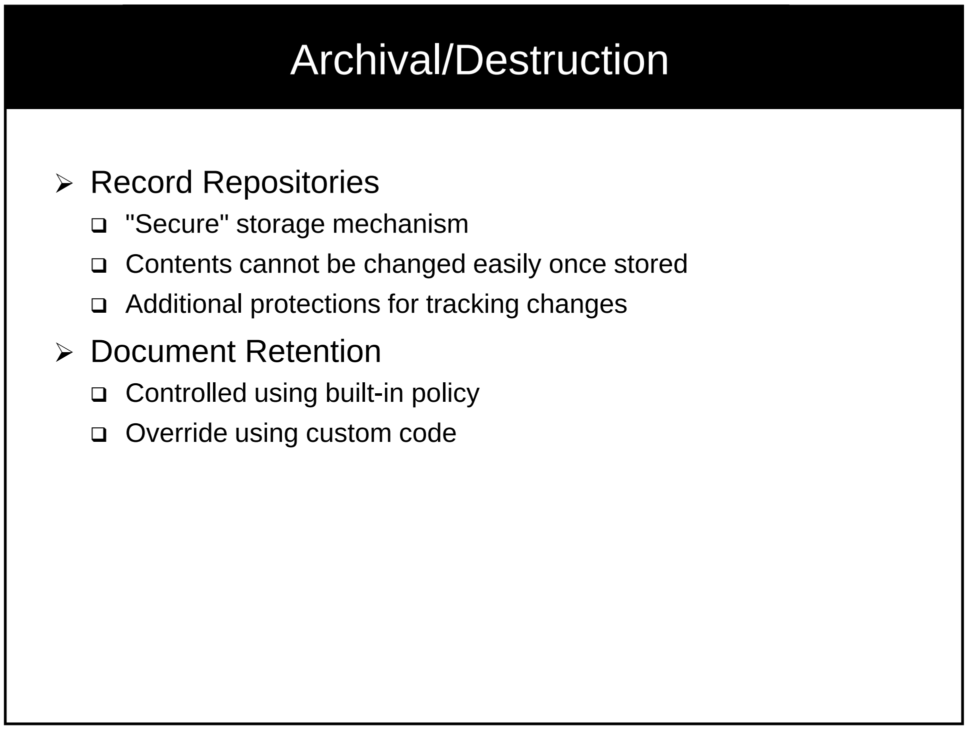


Approval / Publication

- Built-In Approval Mechanisms
  - Available for any list
  - Useful for "Content Moderator" role
- External Approval Mechanisms
  - Applied from outside the context of a list
  - Used to associate a single list item with multiple approval workflows

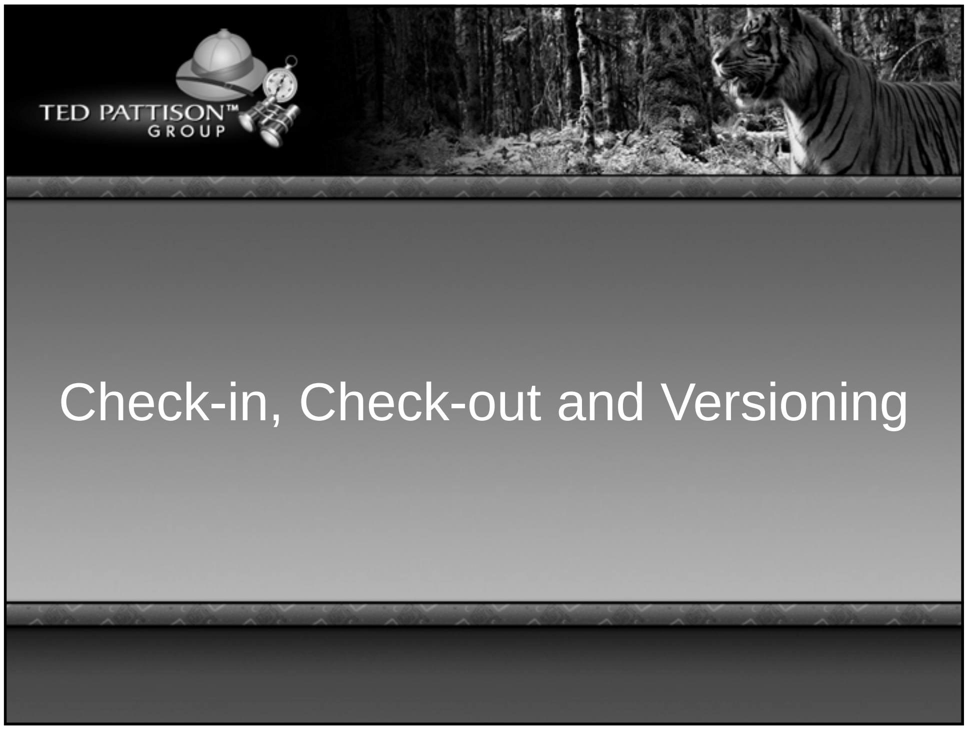
Role-Driven Lifecycles





Archival/Destruction

- Record Repositories
  - "Secure" storage mechanism
  - Contents cannot be changed easily once stored
  - Additional protections for tracking changes
- Document Retention
  - Controlled using built-in policy
  - Override using custom code



TED PATTISON™ GROUP

Check-in, Check-out and Versioning

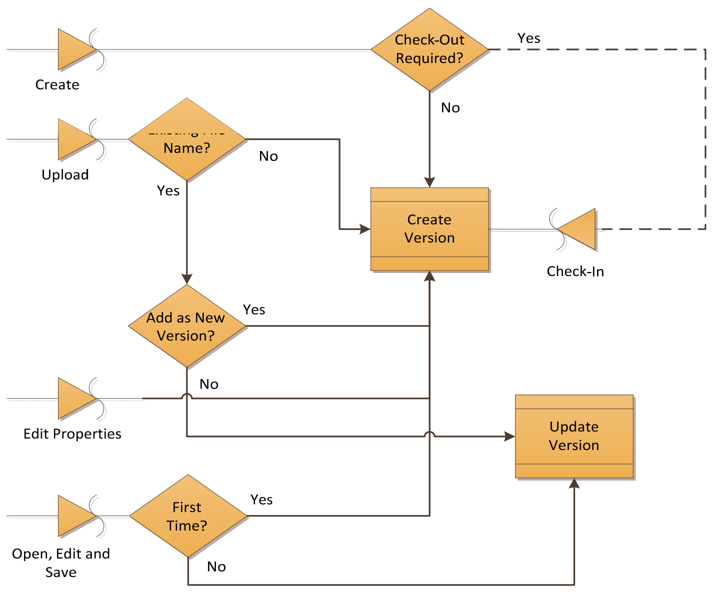
Agenda

- Understand the SharePoint versioning architecture
- Understand the advantages and disadvantages that versioning brings to content management on the SharePoint platform
- Understand the relationship between versioning and check-in/check-out processes

Versioning in SharePoint

- Key feature for content management
  - SharePoint tracks all changes to an item
  - Applies to lists and libraries only
    - Not to content types
  - Can be enabled for any type of list item in any type of list
    - Document Libraries support both major and minor versions
    - Lists support only major versions
- Primary benefits:
  - Ability to roll back to a prior version
  - Ability to selectively view/delete versions
  - Maintain a history of changes

Versioning Rules



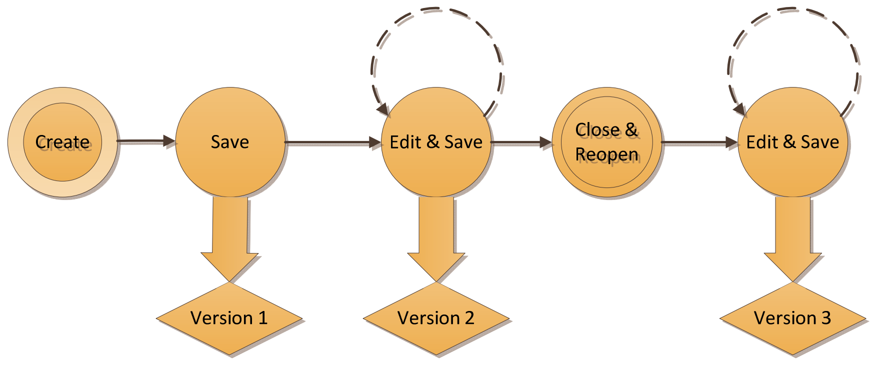
Versioning Pitfalls

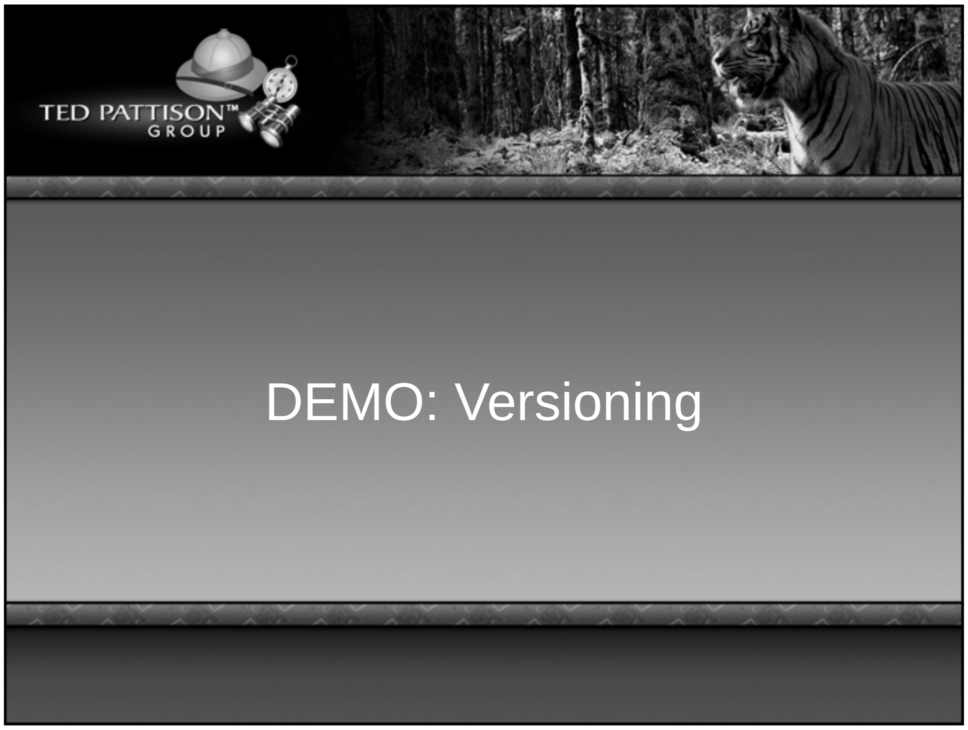
- Storage Cost
  - Each new version copies the bits of the prior version
    - No “differencing engine”
  - No limit to the number of minor versions
- User Adoption Headaches
  - Numbering from zero for minor versions
  - Unwanted side-effects of setting major version limit
    - Auto-deletion when limit is reached

Versioning Issues

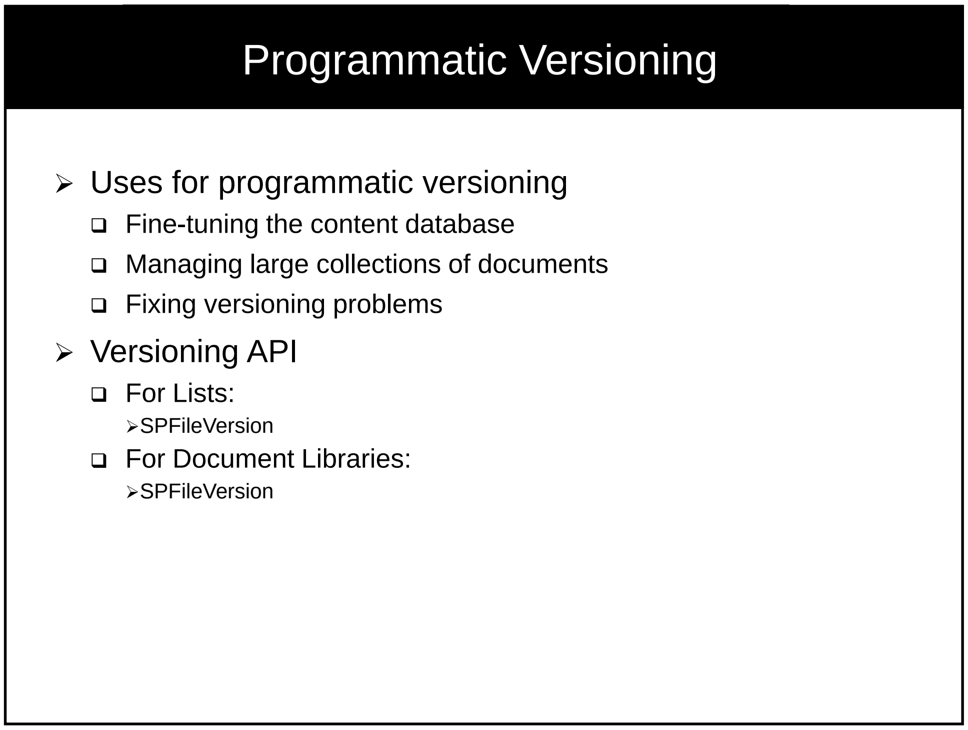
- Numbering
  - Major versions start with “1”
  - Minor versions start with “0.1”
- Content Approval
  - Version always created for “pending” items
    - whether approved or rejected
- Version Limits
  - Keeps older versions if they already exist
    - limit == count if set lower
  - Permanently deletes oldest version when limit exceeded
    - No recycle bin
    - Limit==current count if less than count

Versioning Sequence





DEMO: Versioning



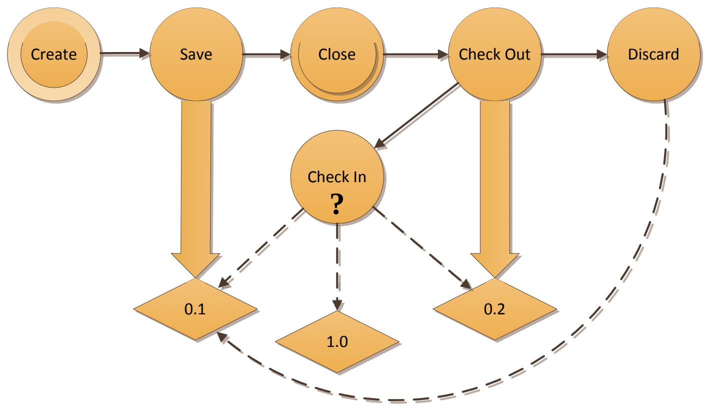
Programmatic Versioning

- Uses for programmatic versioning
  - Fine-tuning the content database
  - Managing large collections of documents
  - Fixing versioning problems
- Versioning API
  - For Lists:
    - SPFileVersion
  - For Document Libraries:
    - SPFileVersion

Check-In / Check-Out

- Standard “locking” mechanism
  - Person who checks out “owns” the item
  - Owner (or delegate) can check in or undo
    - Undo discards new version
- Common Questions:
  - How does check-in / check-out affect versioning?
  - What happens when undoing a check-out?
  - How to force users to check-out documents?

Check-Out Versioning Sequence

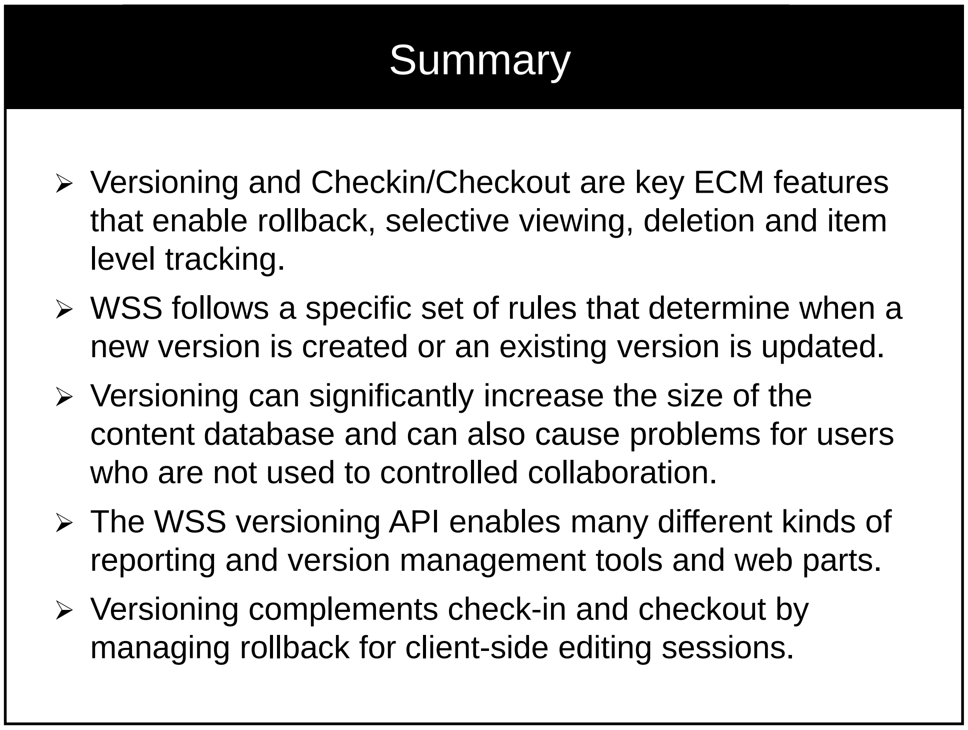


Forcing Check-Out

- Important feature for content management
  - Prevents users from mistaken overwrites
  - Works well with versioning
    - Automatically creates new version on check-in
- Problematic when using content types
  - Check-out applies to entire library
  - Need special code to apply to individual types
    - Custom field type with a hyperlink to an ASPX page that checks out the file.
    - Difficult to disable default links in the list for other content types.

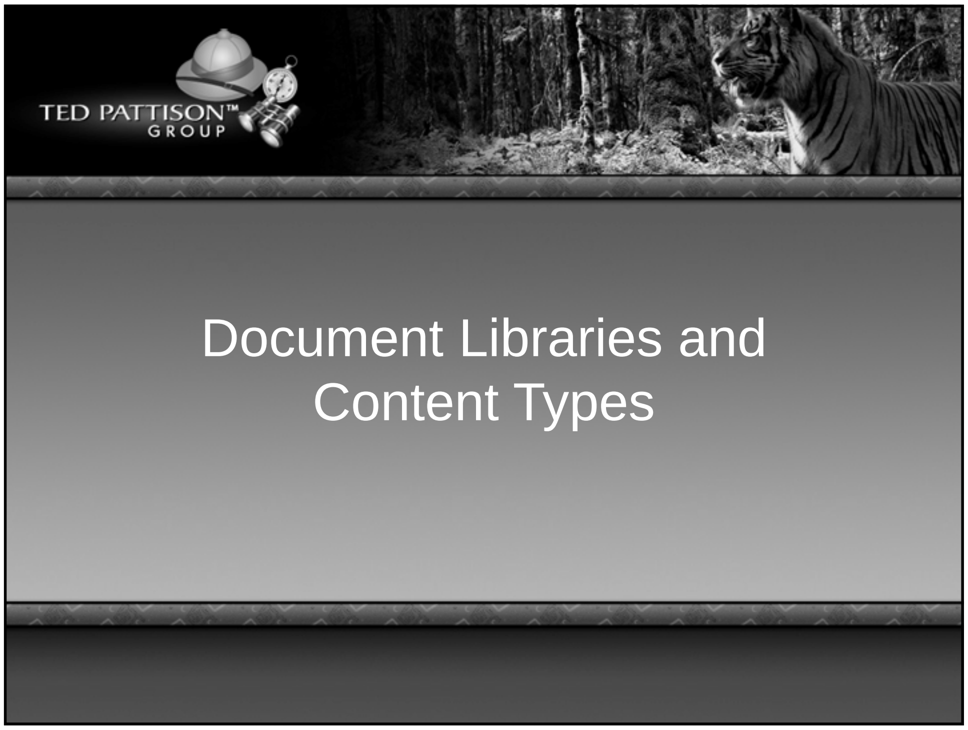
Summary

- The content lifecycle model provides a convenient conceptual framework for understanding ECM on the SharePoint platform.
- Microsoft views ECM as consisting of four "pillars", namely Document Management, Records Management, Web Publishing and Electronic Forms. Much of the SharePoint platform, especially MOSS, is built around these core technologies.
- Role-based design is a useful technique for designing ECM solutions because of the interplay between each role and each stage of the content lifecycle.

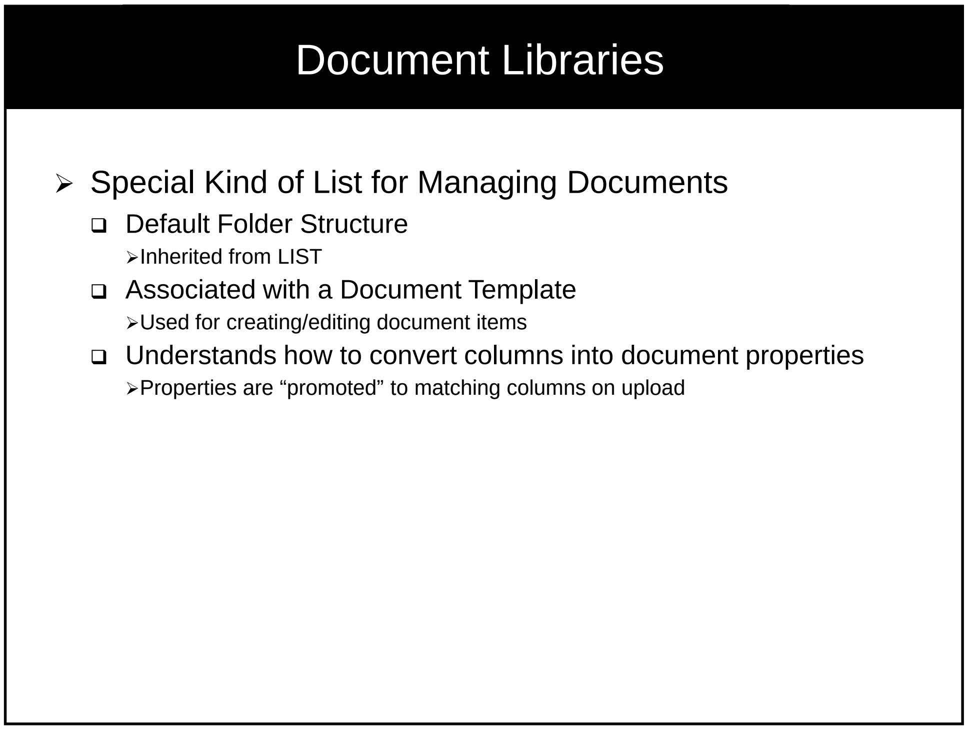


Summary

- Versioning and Checkin/Checkout are key ECM features that enable rollback, selective viewing, deletion and item level tracking.
- WSS follows a specific set of rules that determine when a new version is created or an existing version is updated.
- Versioning can significantly increase the size of the content database and can also cause problems for users who are not used to controlled collaboration.
- The WSS versioning API enables many different kinds of reporting and version management tools and web parts.
- Versioning complements check-in and checkout by managing rollback for client-side editing sessions.



Document Libraries and Content Types

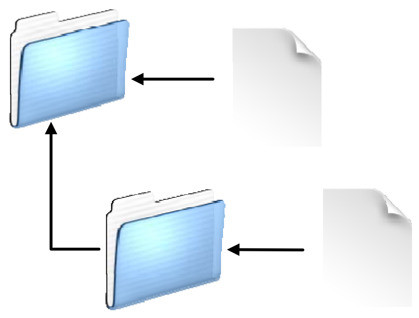


Document Libraries

- Special Kind of List for Managing Documents
  - Default Folder Structure
    - Inherited from LIST
  - Associated with a Document Template
    - Used for creating/editing document items
  - Understands how to convert columns into document properties
    - Properties are "promoted" to matching columns on upload

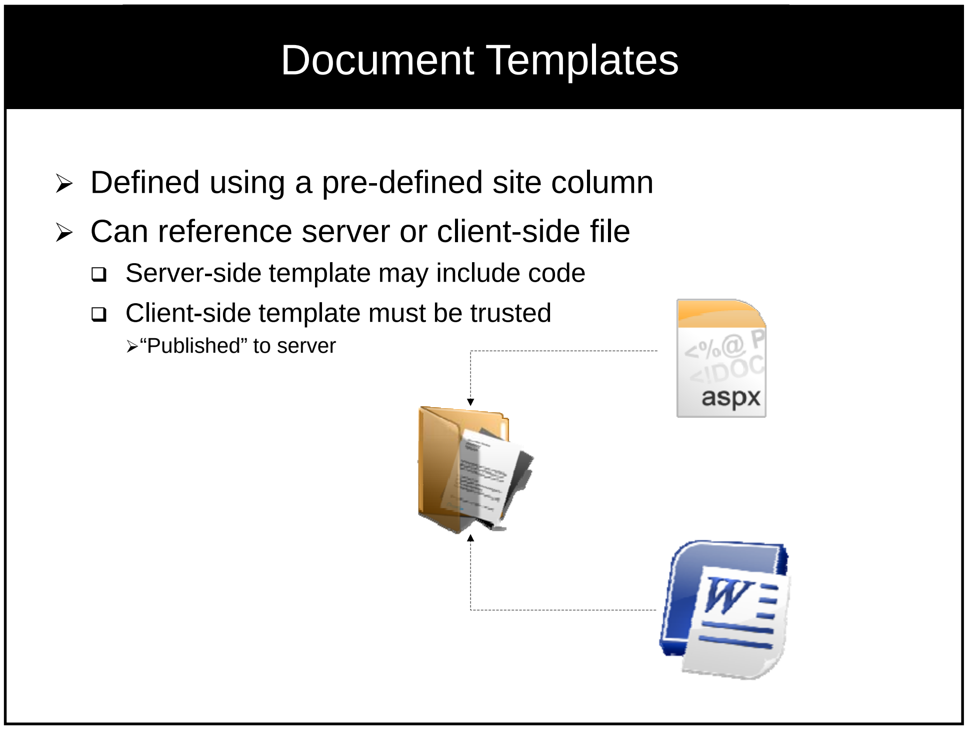
Document Library Folder Structure

- Inherited from List
- Supports standard file i/o operations
  - SPFolder
  - SPFile



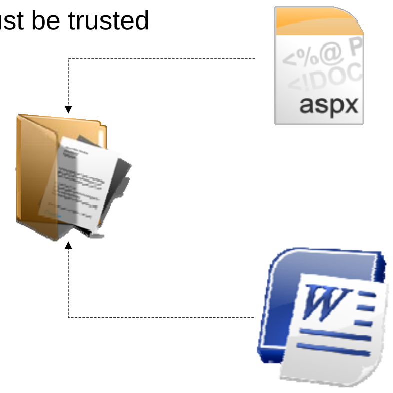
Documents Stored as List Items

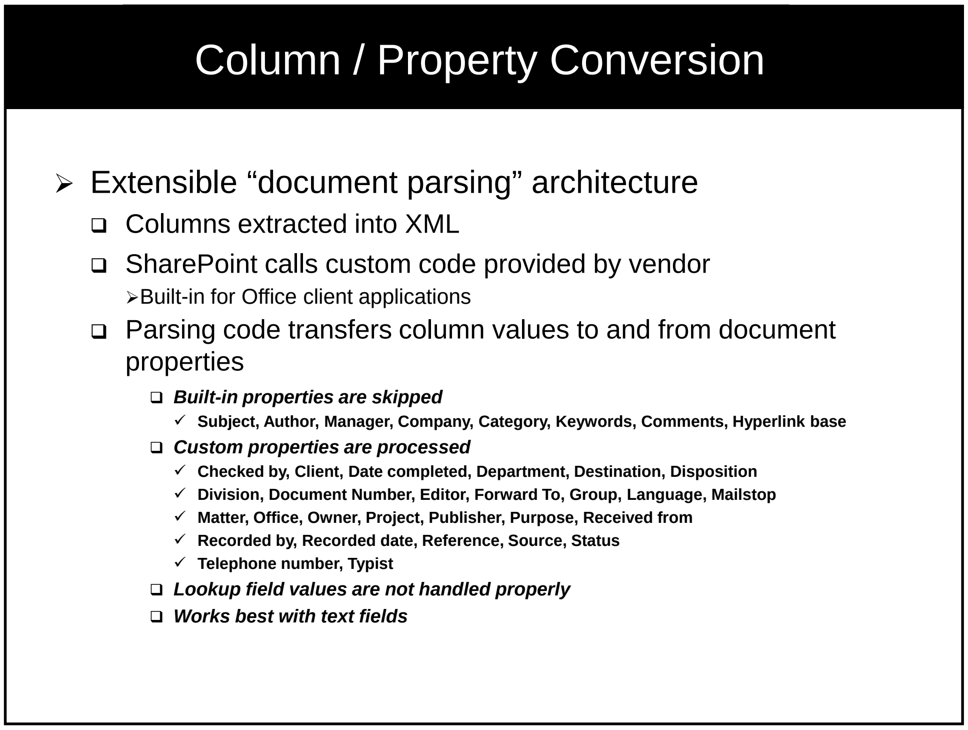
- Easy to manage metadata
  - Site columns
  - Field types
- Easy to leverage other list management code
- Easy to extend with custom behavior
  - Workflow



Document Templates

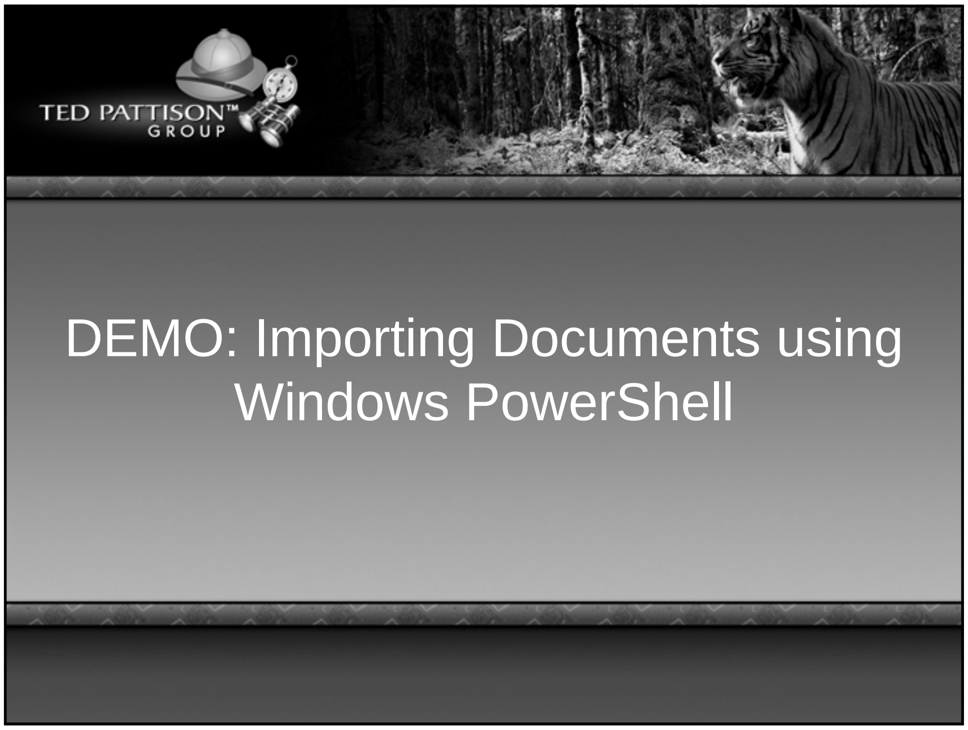
- ➢ Defined using a pre-defined site column
- ➢ Can reference server or client-side file
  - ❑ Server-side template may include code
  - ❑ Client-side template must be trusted
    - ➢"Published" to server



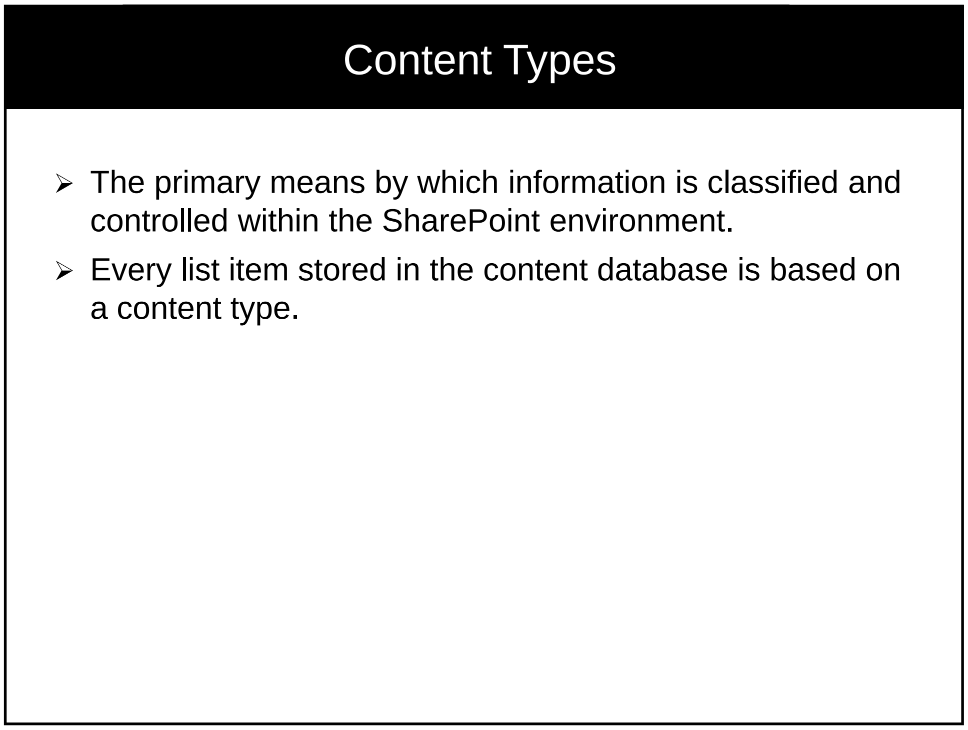


Column / Property Conversion

- ➢ Extensible "document parsing" architecture
  - ❑ Columns extracted into XML
  - ❑ SharePoint calls custom code provided by vendor
    - ➢Built-in for Office client applications
  - ❑ Parsing code transfers column values to and from document properties
    - ❑ ***Built-in properties are skipped***
      - ✓ **Subject, Author, Manager, Company, Category, Keywords, Comments, Hyperlink base**
    - ❑ ***Custom properties are processed***
      - ✓ **Checked by, Client, Date completed, Department, Destination, Disposition**
      - ✓ **Division, Document Number, Editor, Forward To, Group, Language, Mailstop**
      - ✓ **Matter, Office, Owner, Project, Publisher, Purpose, Received from**
      - ✓ **Recorded by, Recorded date, Reference, Source, Status**
      - ✓ **Telephone number, Typist**
    - ❑ ***Lookup field values are not handled properly***
    - ❑ ***Works best with text fields***



DEMO: Importing Documents using Windows PowerShell



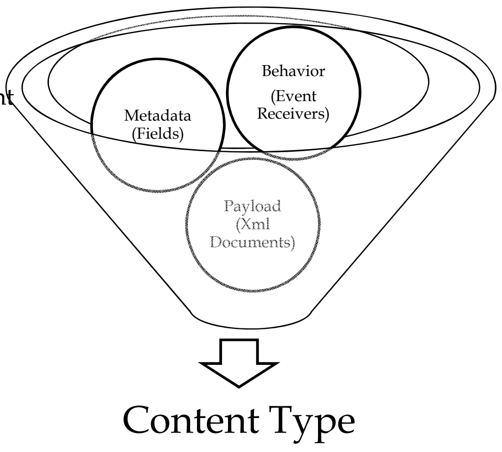
Content Types

- The primary means by which information is classified and controlled within the SharePoint environment.
- Every list item stored in the content database is based on a content type.

What is a Content Type?

Three Primary Components:

1. Metadata (Columns)
   Added to list when content type is bound to it.
2. Behavior (Code)
   Called when list items are added or modified.
3. Payload (XML)
   Used to find event receivers, policies and custom data.



Uses for Content Types

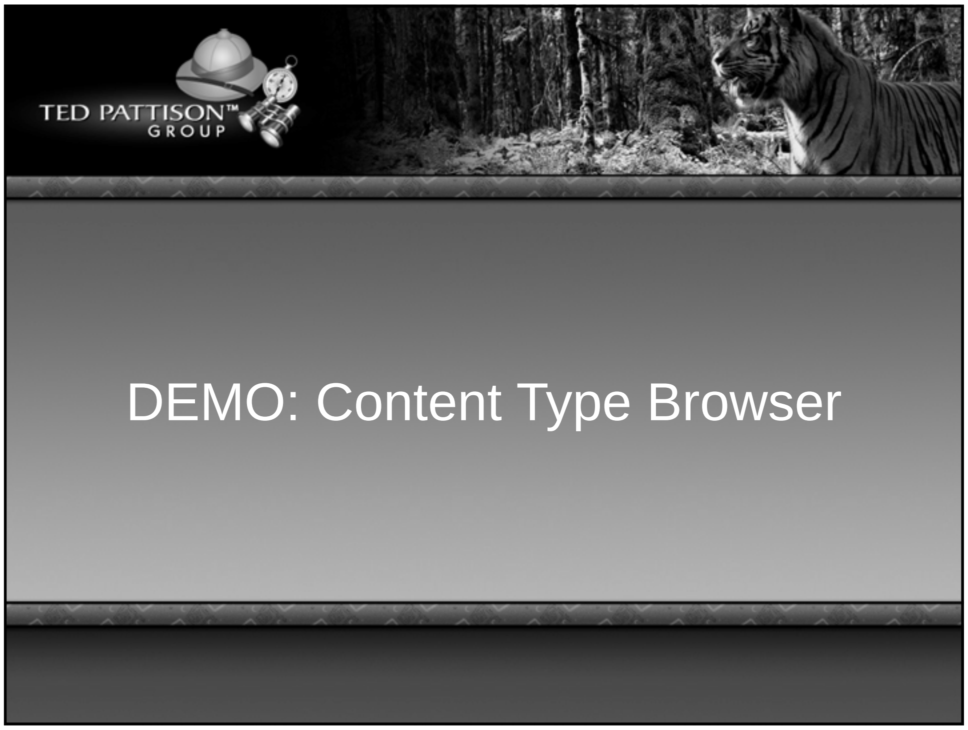
- Classification
  - Can define content type hierarchies to manage metadata
  - Problematic if too deeply nested
- Organization
  - Good way to organize groups of documents by type
  - Best used when organization is not likely to change.
- Validation
  - Provides a way to "encapsulate" metadata.
  - Can be used to apply consistent rules to a group of documents
- Access Control
  - Provides a way to encapsulate access rules.
  - Rules may be embedded in the type and referenced from code.

Content Types and Lists

- Content types can be used on multiple lists
- Lists can be associated with multiple content types
- Lists can have its own columns in addition to content type columns
- Limitations:
  - Only one content type per document at a time

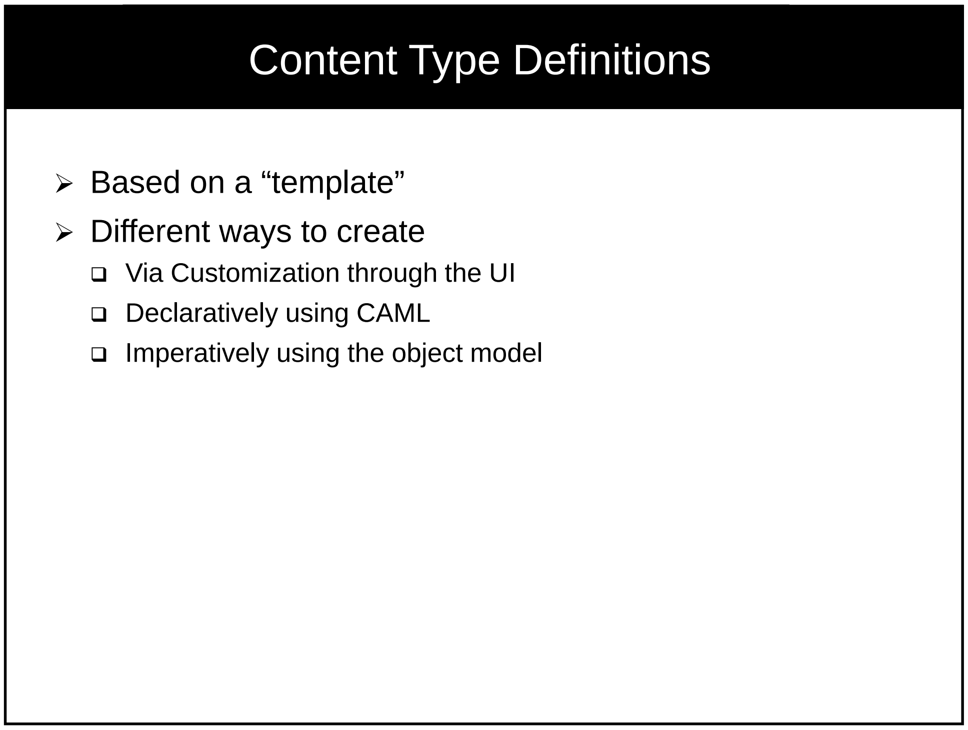
Content Type Templates

- Specifies the layout of metadata
- Specifies class/assembly for event receivers
- Specifies which events are enabled
- May include custom XML schemas



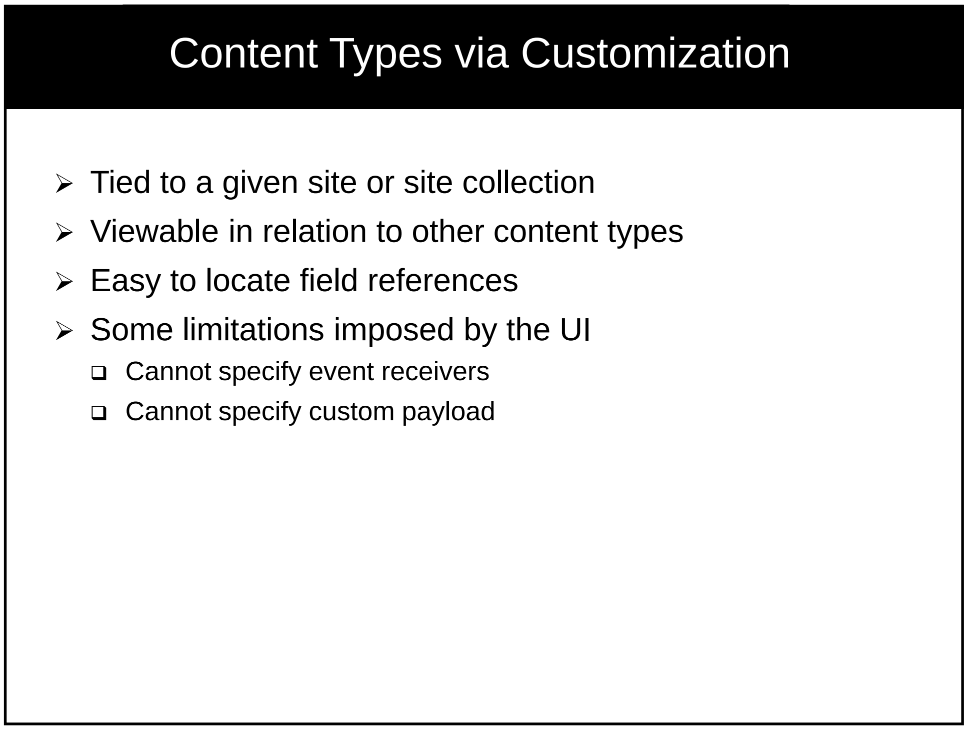
TED PATTISON™ GROUP

DEMO: Content Type Browser



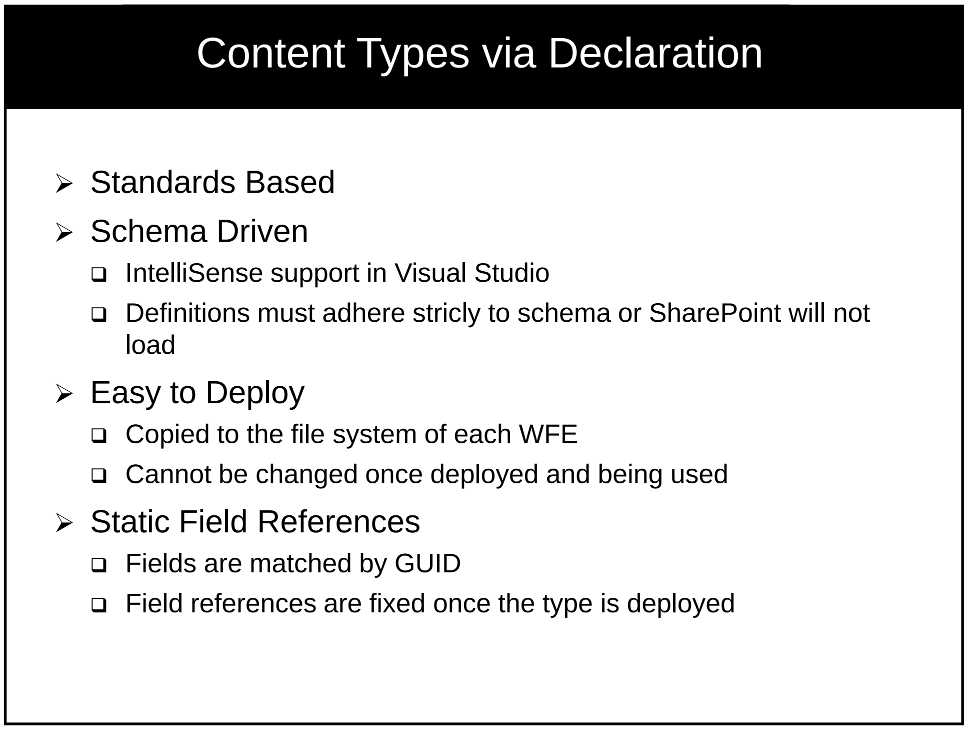
Content Type Definitions

- Based on a "template"
- Different ways to create
  - Via Customization through the UI
  - Declaratively using CAML
  - Imperatively using the object model



Content Types via Customization

- Tied to a given site or site collection
- Viewable in relation to other content types
- Easy to locate field references
- Some limitations imposed by the UI
  - Cannot specify event receivers
  - Cannot specify custom payload



Content Types via Declaration

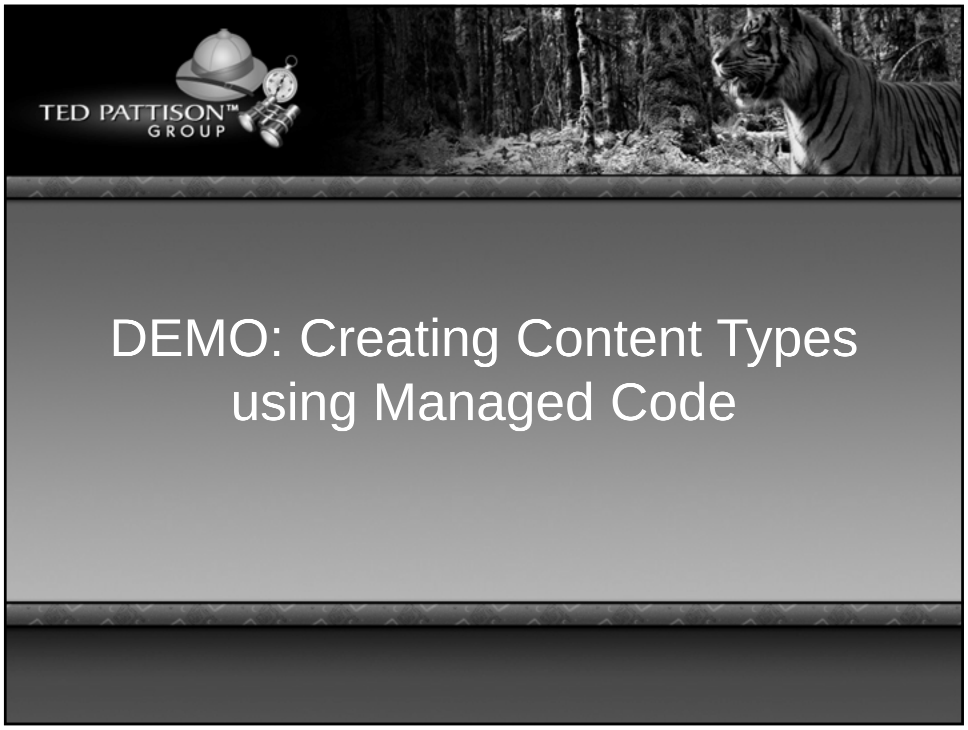
- Standards Based
- Schema Driven
  - IntelliSense support in Visual Studio
  - Definitions must adhere stricly to schema or SharePoint will not load
- Easy to Deploy
  - Copied to the file system of each WFE
  - Cannot be changed once deployed and being used
- Static Field References
  - Fields are matched by GUID
  - Field references are fixed once the type is deployed

Content Types via Code

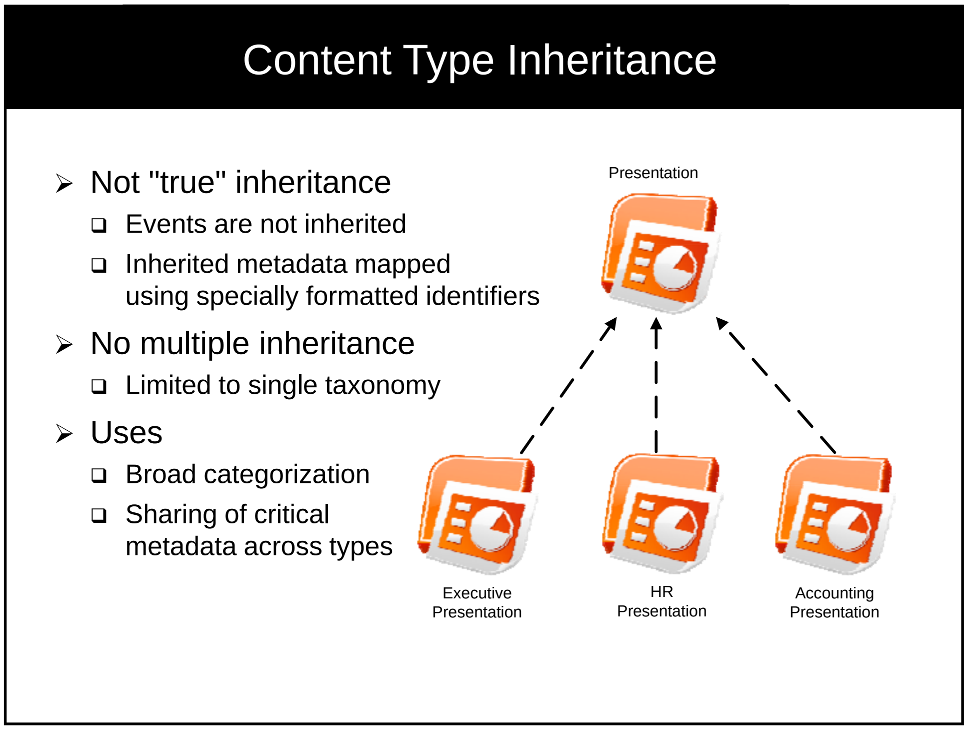
- Standards Based
  - Uses the same CAML schema
- Strongly Typed
  - Object model provides strong typing for many elements
- Easy to Deploy
  - Code model means compiled assemblies
- Static and Dynamic Field References
  - Object model performs field lookups by name
  - Actual fields may be determined when code is run
- Reusable and Extensible
  - Redeploy assembly that builds the content type
  - Code based approach easier to extend

Creating Content Types in Code

```
using (SPSite site = new SPSIte(url)) {

    using (SPWeb web = site.OpenWeb()) {

        SPContentType baseType =
            web.AvailableContentTypes[“Document”];

        SPContentType proposalType = new SPContentType(
            baseType, web.ContentTypes, “Project Proposal”);

        web.ContentTypes.Add(proposalType);

        proposalType.FieldLinks.Add(
            new SPFieldLink(web.AvailableFields[“Author”]));
    }
  }
```

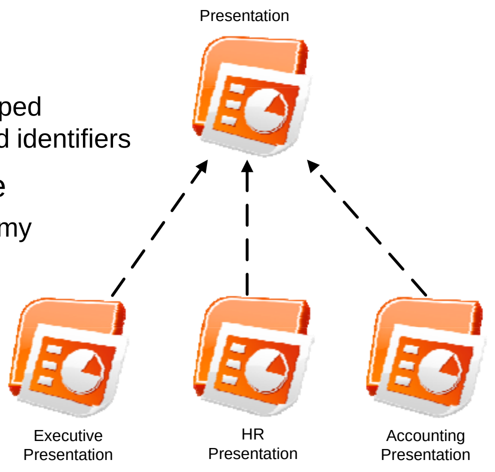


DEMO: Creating Content Types using Managed Code



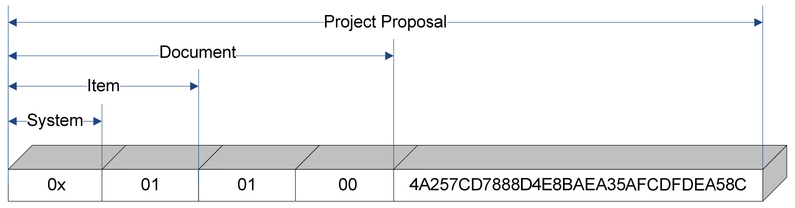
Content Type Inheritance

- Not "true" inheritance
  - Events are not inherited
  - Inherited metadata mapped using specially formatted identifiers
- No multiple inheritance
  - Limited to single taxonomy
- Uses
  - Broad categorization
  - Sharing of critical metadata across types



Content Type Identifiers

- Special Format for maintaining inheritance relationships between content types



- Difficult to work with manually
  - Assigned automatically when using the object model

Extending Content Types

- Event Receivers
  - Provides a way to add custom behavior to a content type
  - Many levels of events are supported
    - Adding and removing items
    - Changes to items
    - Check-in/Check-out of items
    - Changes to schema (add/remove columns)
- XML Documents
  - Extensible way to embed custom information into a content type
  - Useful for content validation (store custom validation schema)
  - Used internally to map types to event receivers

Policies in Content Types

- Policies can be assigned to Content Types
- Cannot change policy while assigned
- Content Types inherit policies
  - Happens when derived type is created
  - Policy definition copied into derived type
  - Changes to parent policy are propagated
- Policy Definition stored in XML Documents
- Policy Definition copied into Office Documents

Summary

- Document libraries are a specialization of list that adds document templates and document property promotion.
- Document libraries and lists support the same standard file i/o operations, providing a consistent API based on folders, subfolders and files.
- Document templates can be bound to server-side (.ASPX) or client-side (.DOTX, etc.) files, enabling a variety of methods for ensuring that documents are created properly.

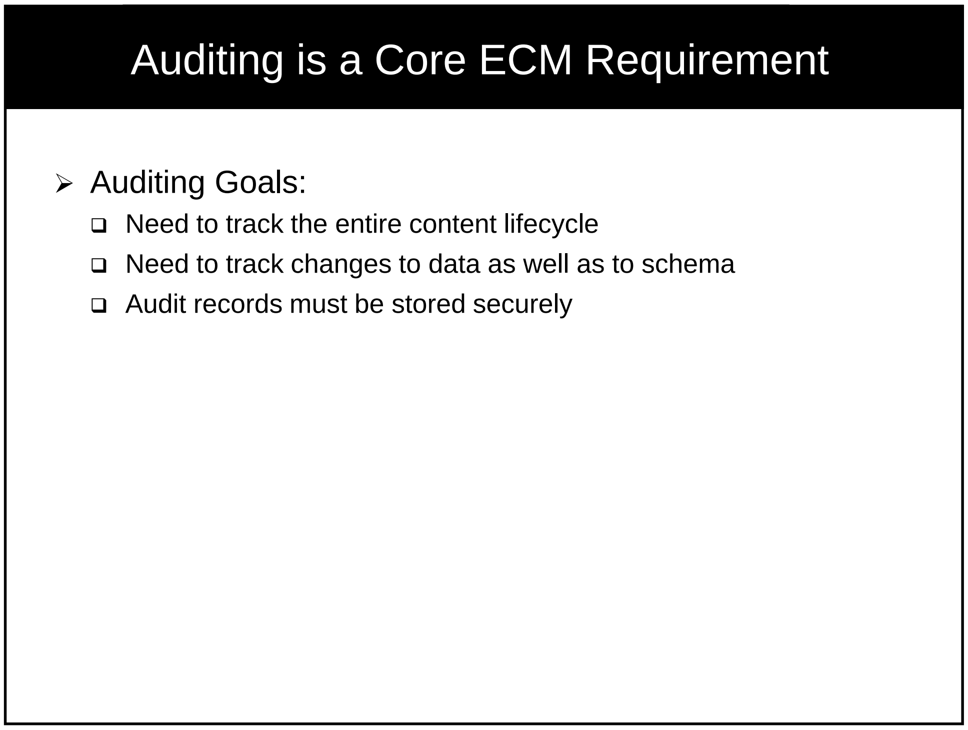
Summary (cont')

- The three primary components of content types are metadata (fields), behavior (event receivers), and an extensible payload (XML documents). Together, they provide the foundation for ECM on the SharePoint platform.
- Although content types support simple metadata inheritance, they do not support behavioral inheritance, limiting their usefulness for building deep taxonomies.
- A single document can only be associated with one content type at a time, which can affect the extensibility of solutions based on them.

Summary

- Metadata drives all stages of the content lifecycle.
- Content types provide control of metadata throughout the content lifecycle.
- Using schematized rules allow flexible validation of content metadata.
- Rules can be embedded within content types and applied using event receivers or information policy features.

Content Tracking, Auditing and Security



Auditing is a Core ECM Requirement

- Auditing Goals:
  - Need to track the entire content lifecycle
  - Need to track changes to data as well as to schema
  - Audit records must be stored securely

Auditing applies to all kinds of content

- Site Collections
- Sites
- Lists
- Document Libraries
- List Items
- ?

Auditing in WSS

- Audit records stored in the content database
- Audit records backed up along with content
- OOB support for all content related operations
- Ability to define custom audit record types

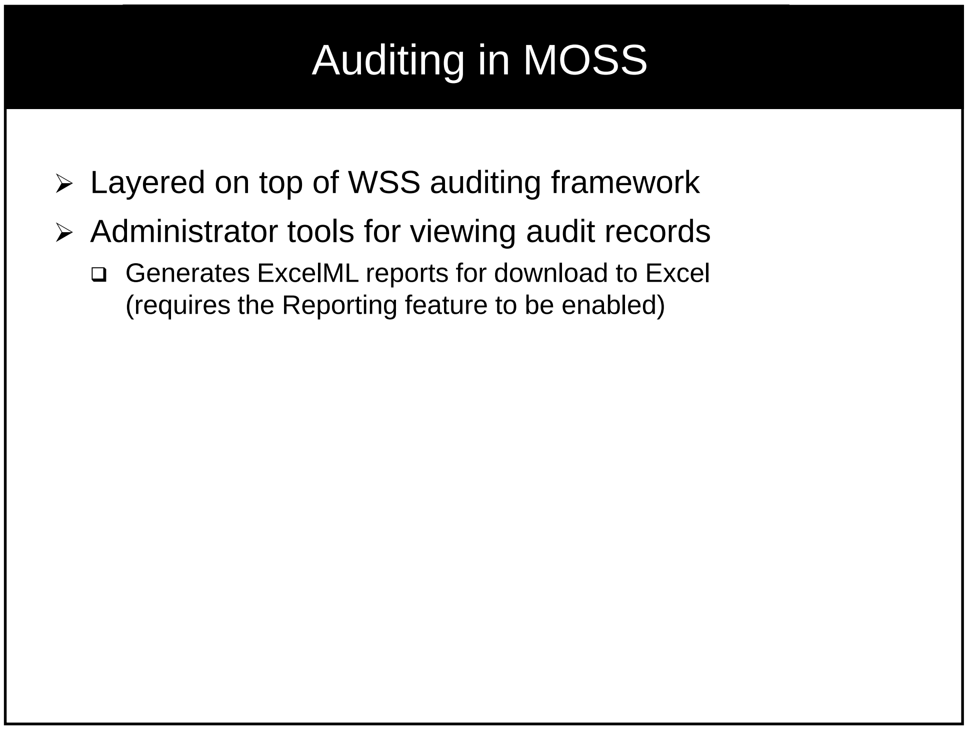
- No UI for viewing audit records
- Auditing turned off by default
  - Must be enabled via code

Auditing in WSS

- No UI for viewing audit records
  - Protects against accidental viewing
  - Reduces security risks
    - Unintentionally turning it off
    - Unintentionally changing audit behavior
- Auditing turned off by default
  - Must be enabled via code

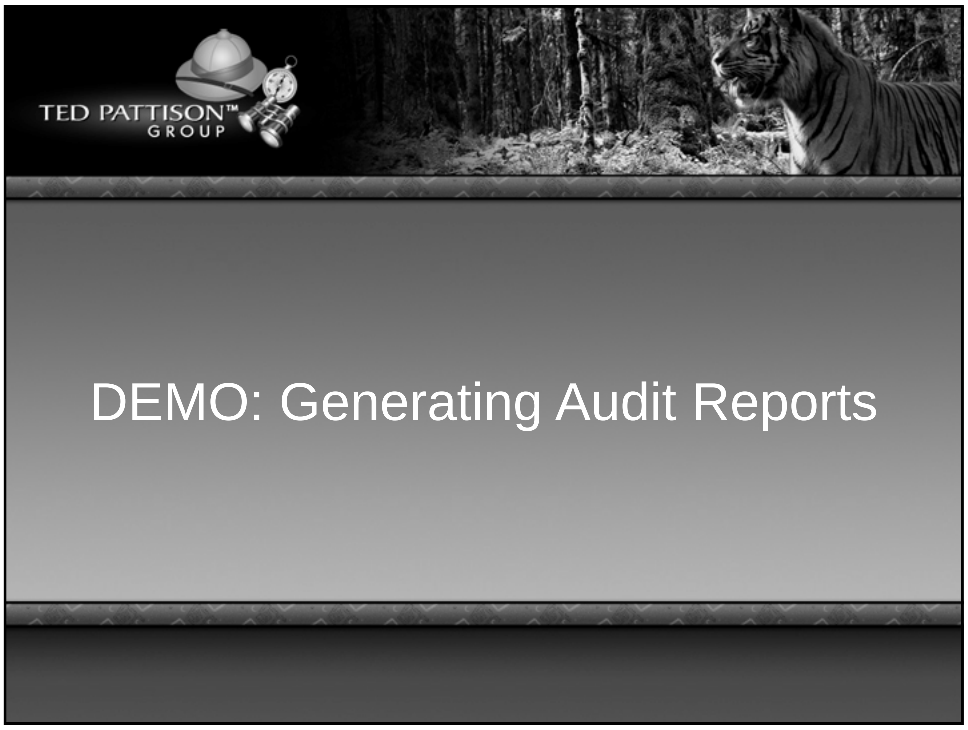
Auditable Events in WSS

SPAuditMask Enumeration Values	
All	ProfileChange
CheckIn	SchemaChange
CheckOut	Search
ChildDelete	SecurityChange
Copy	Undelete
Delete	Update
Move	View
None	Workflow



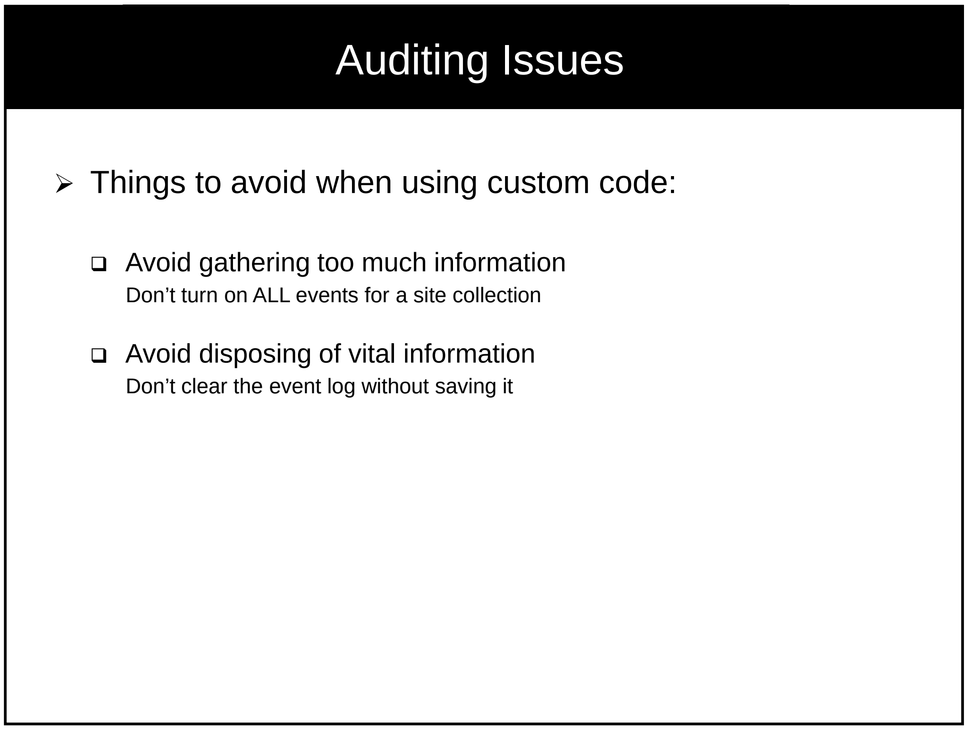
Auditing in MOSS

- Layered on top of WSS auditing framework
- Administrator tools for viewing audit records
  - Generates ExcelML reports for download to Excel (requires the Reporting feature to be enabled)



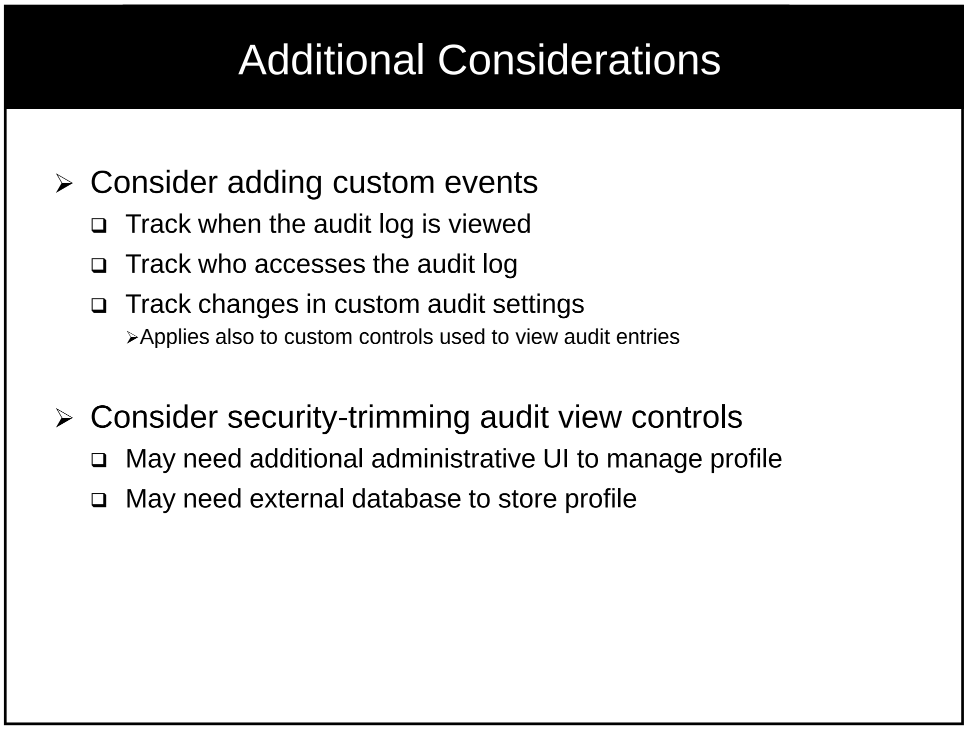
TED PATTISON™ GROUP

DEMO: Generating Audit Reports



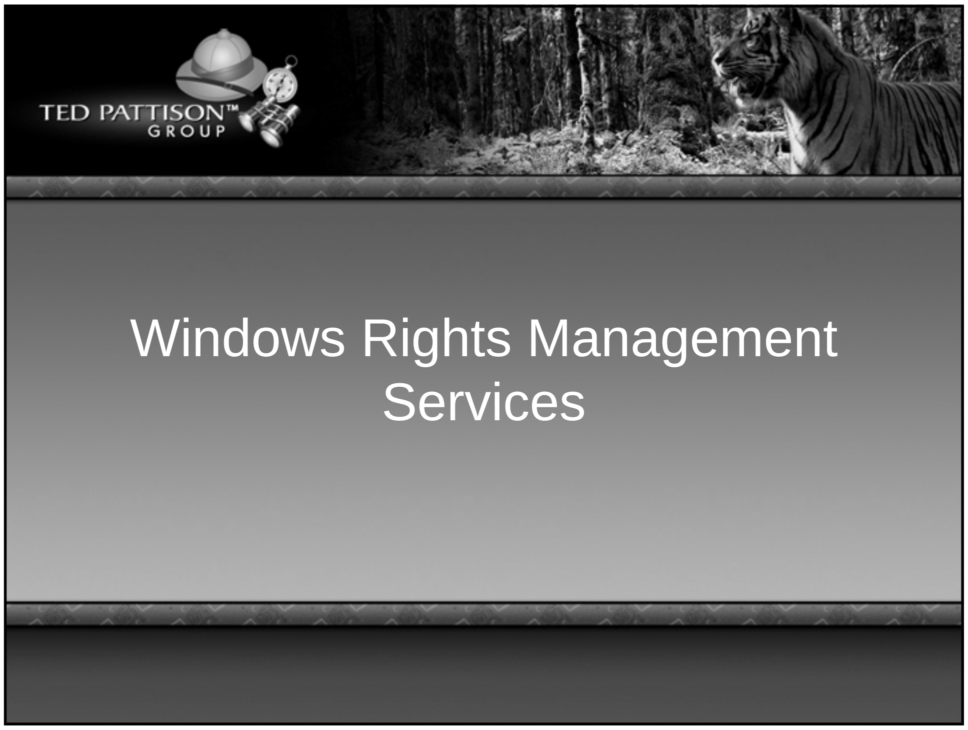
Auditing Issues

- Things to avoid when using custom code:
  - Avoid gathering too much information
    Don’t turn on ALL events for a site collection
  - Avoid disposing of vital information
    Don’t clear the event log without saving it

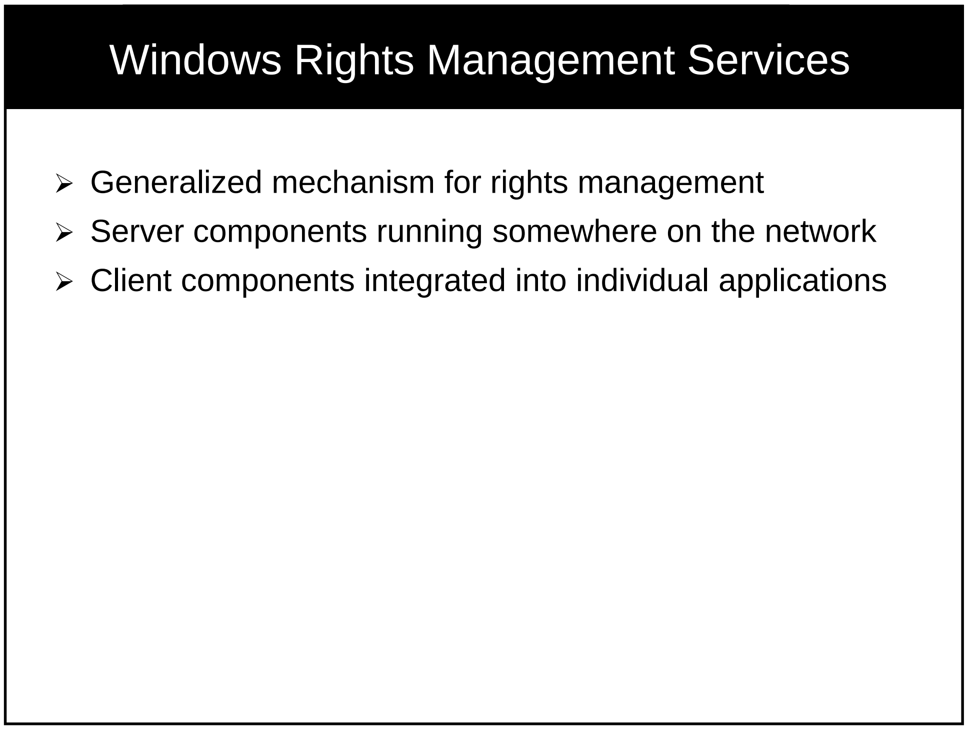


Additional Considerations

- Consider adding custom events
  - Track when the audit log is viewed
  - Track who accesses the audit log
  - Track changes in custom audit settings
    - Applies also to custom controls used to view audit entries
- Consider security-trimming audit view controls
  - May need additional administrative UI to manage profile
  - May need external database to store profile

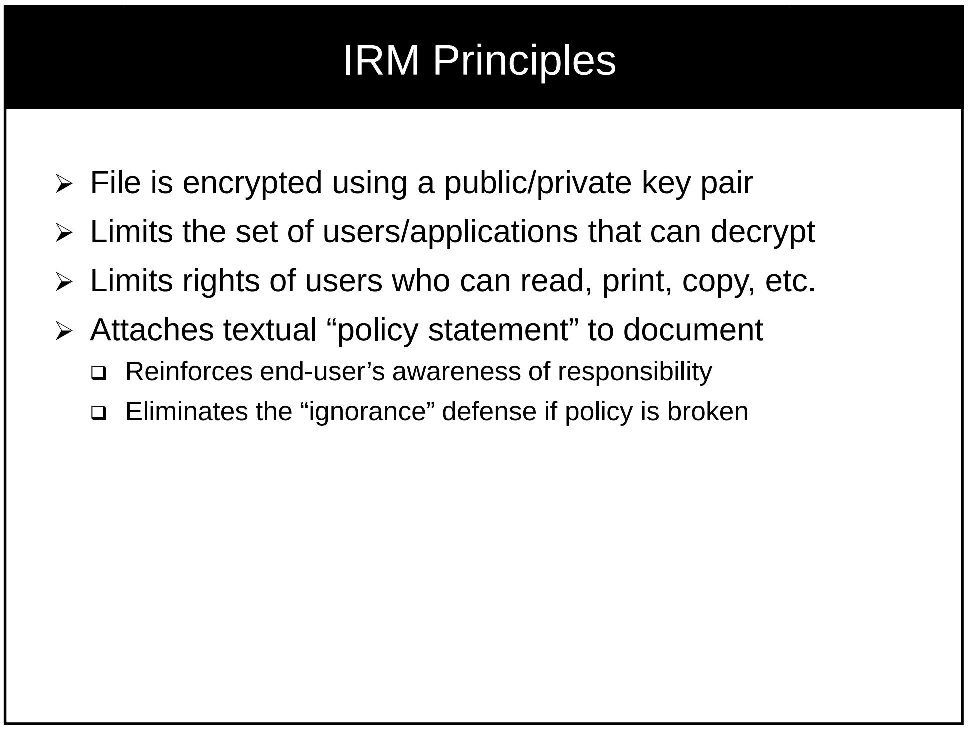


Windows Rights Management Services



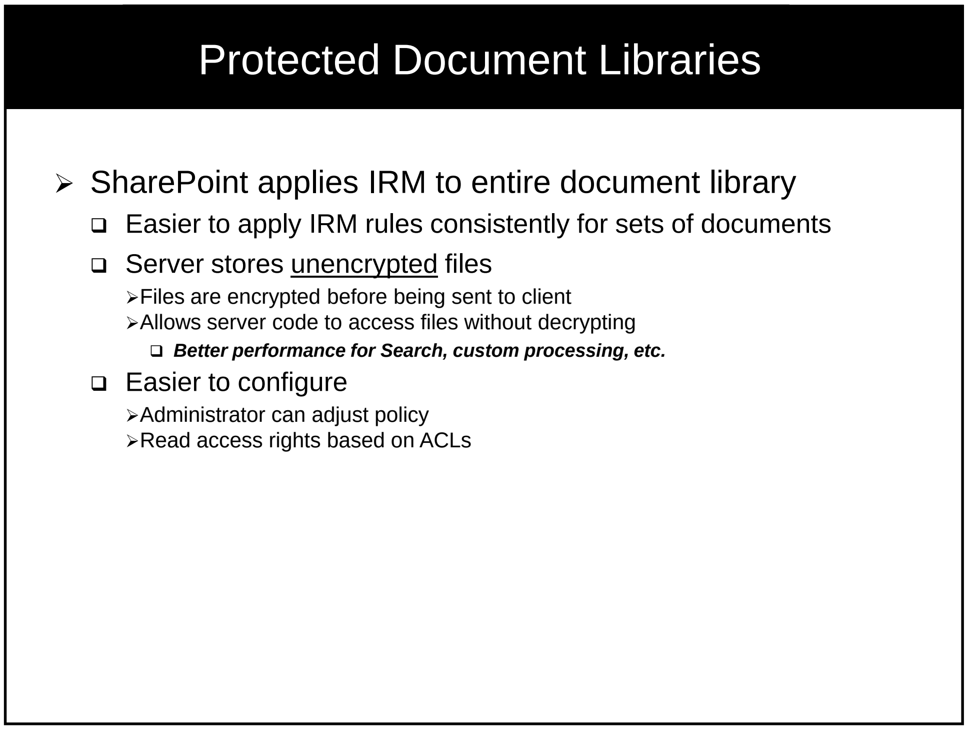
Windows Rights Management Services

- Generalized mechanism for rights management
- Server components running somewhere on the network
- Client components integrated into individual applications



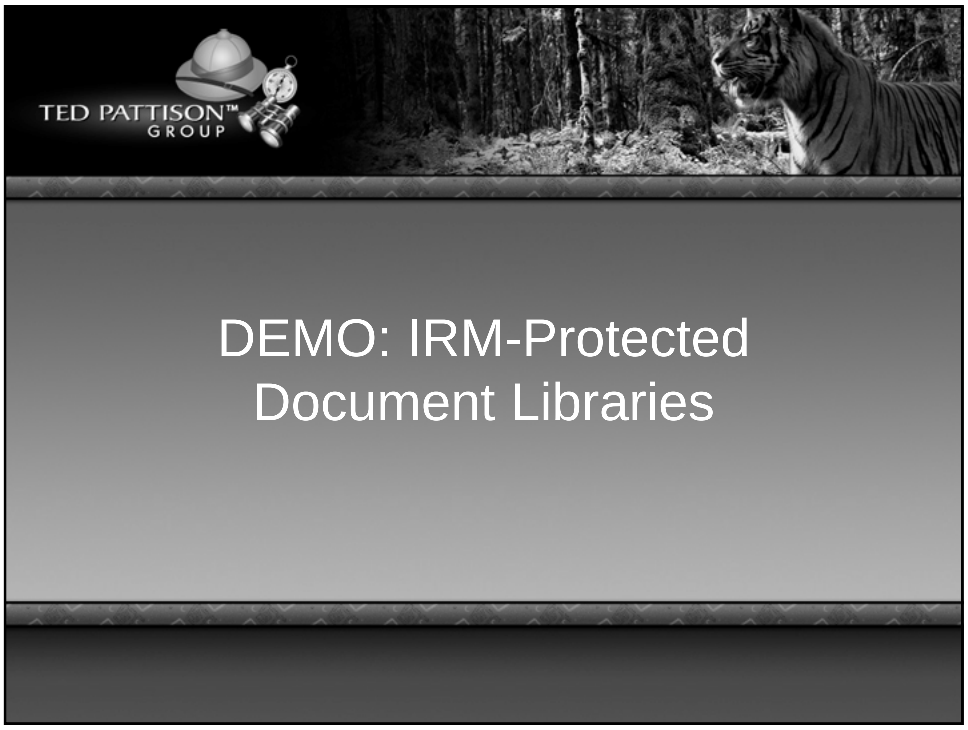
IRM Principles

- File is encrypted using a public/private key pair
- Limits the set of users/applications that can decrypt
- Limits rights of users who can read, print, copy, etc.
- Attaches textual “policy statement” to document
  - Reinforces end-user’s awareness of responsibility
  - Eliminates the “ignorance” defense if policy is broken



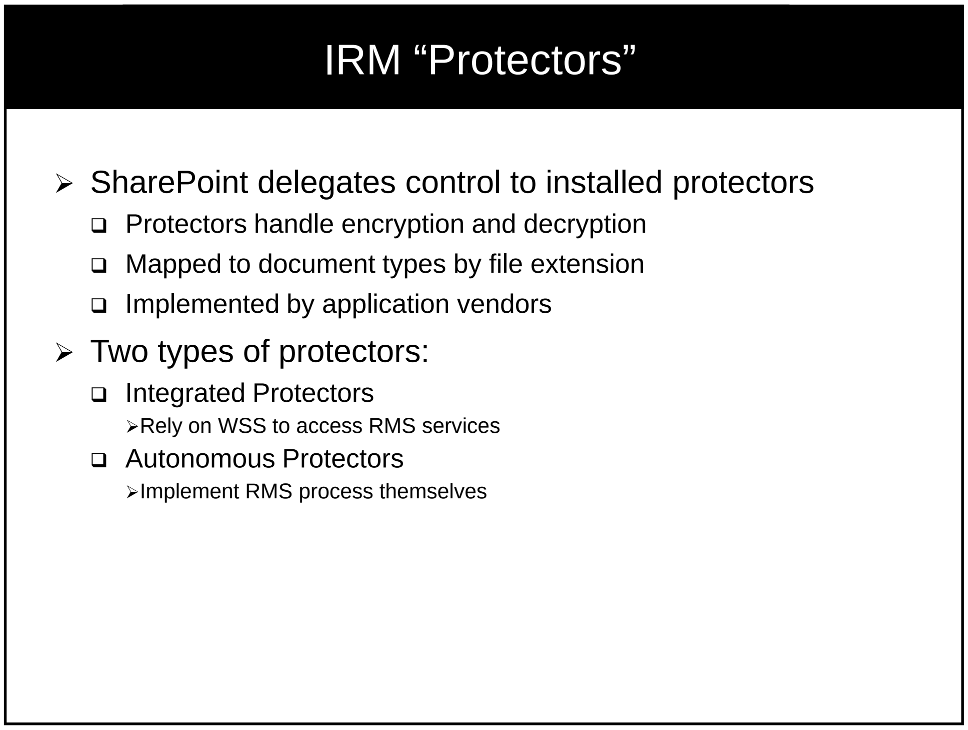
Protected Document Libraries

- SharePoint applies IRM to entire document library
  - Easier to apply IRM rules consistently for sets of documents
  - Server stores unencrypted files
    - Files are encrypted before being sent to client
    - Allows server code to access files without decrypting
      - ***Better performance for Search, custom processing, etc.***
  - Easier to configure
    - Administrator can adjust policy
    - Read access rights based on ACLs



TED PATTISON™ GROUP

DEMO: IRM-Protected Document Libraries



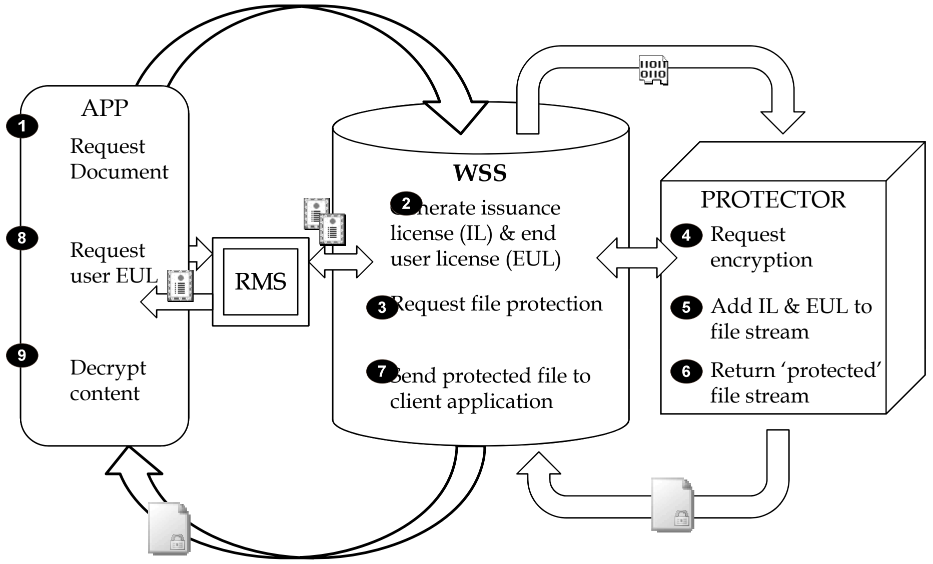
IRM "Protectors"

- SharePoint delegates control to installed protectors
  - Protectors handle encryption and decryption
  - Mapped to document types by file extension
  - Implemented by application vendors
- Two types of protectors:
  - Integrated Protectors
    - Rely on WSS to access RMS services
  - Autonomous Protectors
    - Implement RMS process themselves

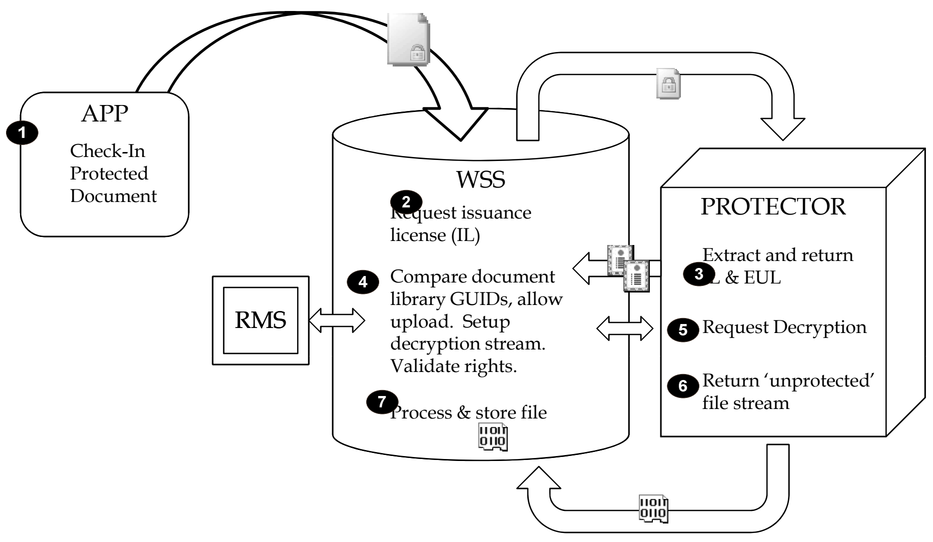
Integrated IRM Protectors

- Rely on SharePoint to communicate with RMS
- SharePoint calls the protector during the encryption / decryption process
- Protector is only responsible for providing the core protection mechanism.

Integrated Protector (read)


APP
1 Request Document
8 Request user EUL
9 Decrypt content
RMS
WSS
2 Generate issuance license (IL) & end user license (EUL)
3 Request file protection
7 Send protected file to client application
PROTECTOR
4 Request encryption
5 Add IL & EUL to file stream
6 Return 'protected' file stream

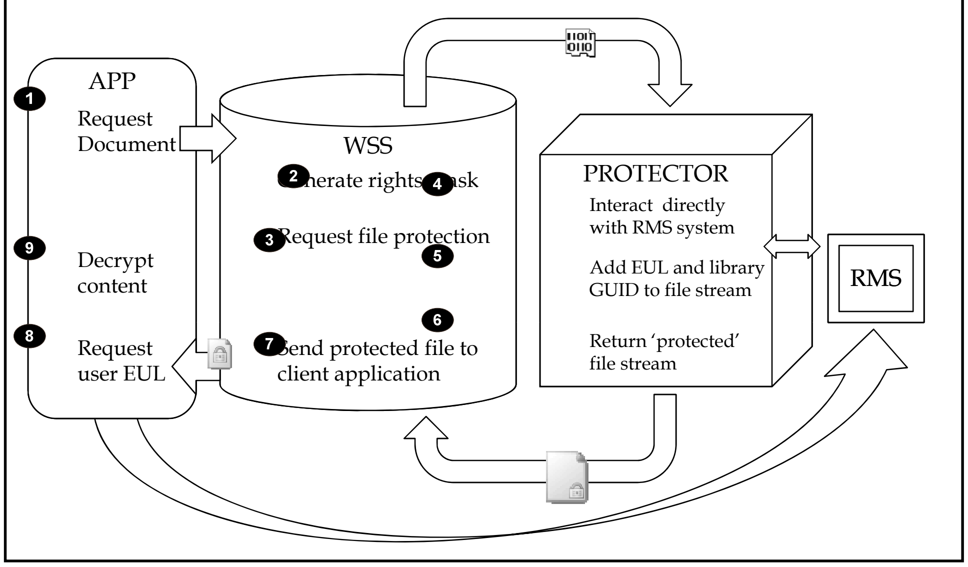
Integrated Protector (check-in)


APP
1 Check-In Protected Document
WSS
2 Request issuance license (IL)
4 Compare document library GUIDs, allow upload. Setup decryption stream. Validate rights.
7 Process & store file
RMS
PROTECTOR
3 Extract and return IL & EUL
5 Request Decryption
6 Return 'unprotected' file stream

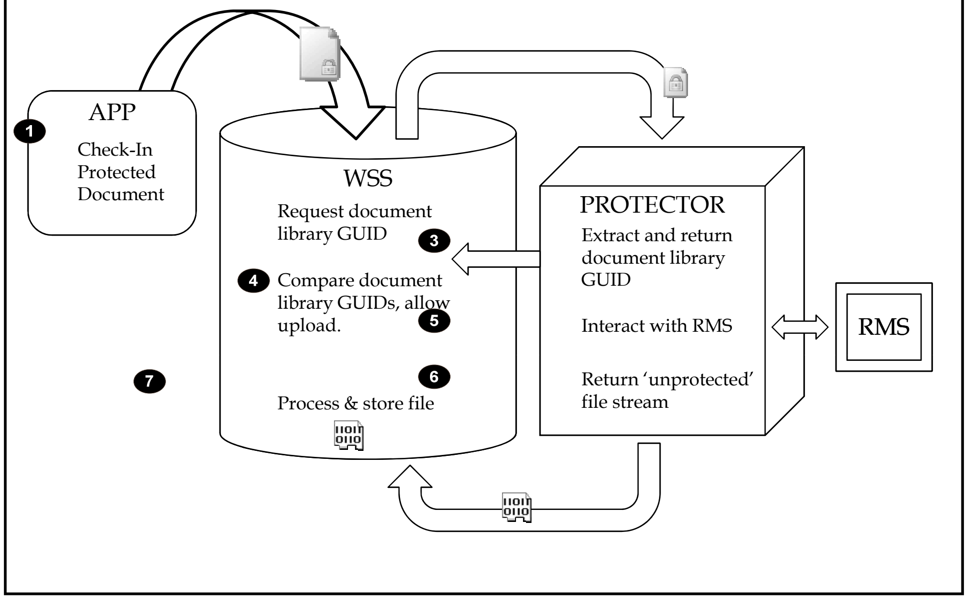
Autonomous IRM Protectors

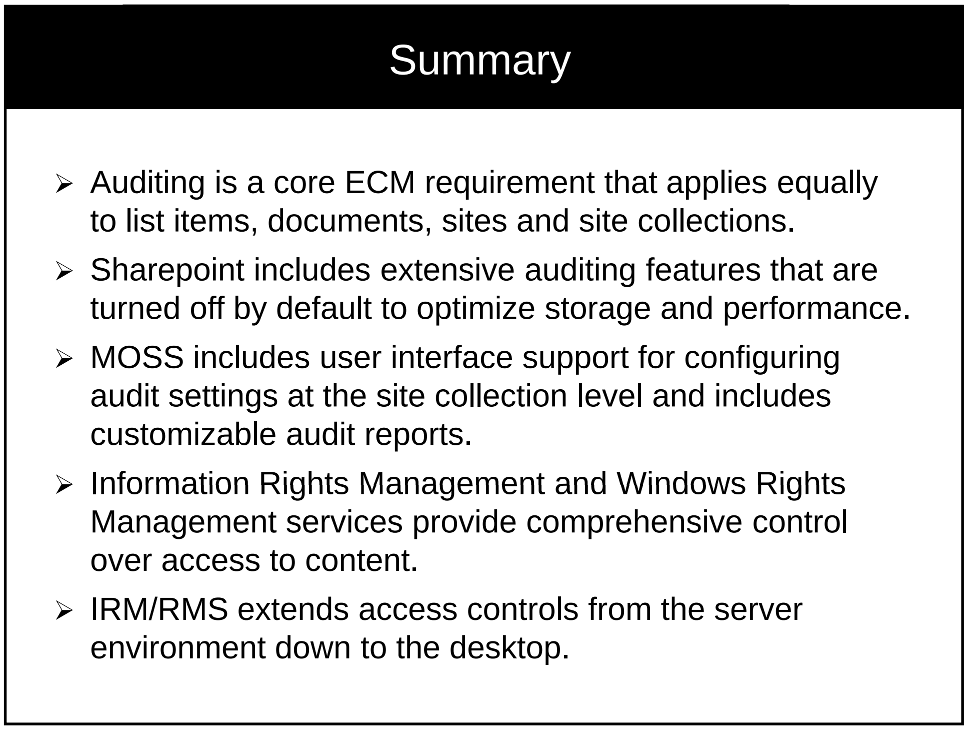
- Responsible for everything dealing with RMS
  - Locating the RMS server
  - Communicating with the RMS server
  - Performing the encryption/decryption
  - Managing the entire IRM process

Autonomous Protector (read)


APP
1 Request Document
9 Decrypt content
8 Request user EUL
WSS
2 Generate rights mask 4
3 Request file protection 5
6
7 Send protected file to client application
PROTECTOR
Interact directly with RMS system
Add EUL and library GUID to file stream
Return 'protected' file stream
RMS

Autonomous Protector (check-in)

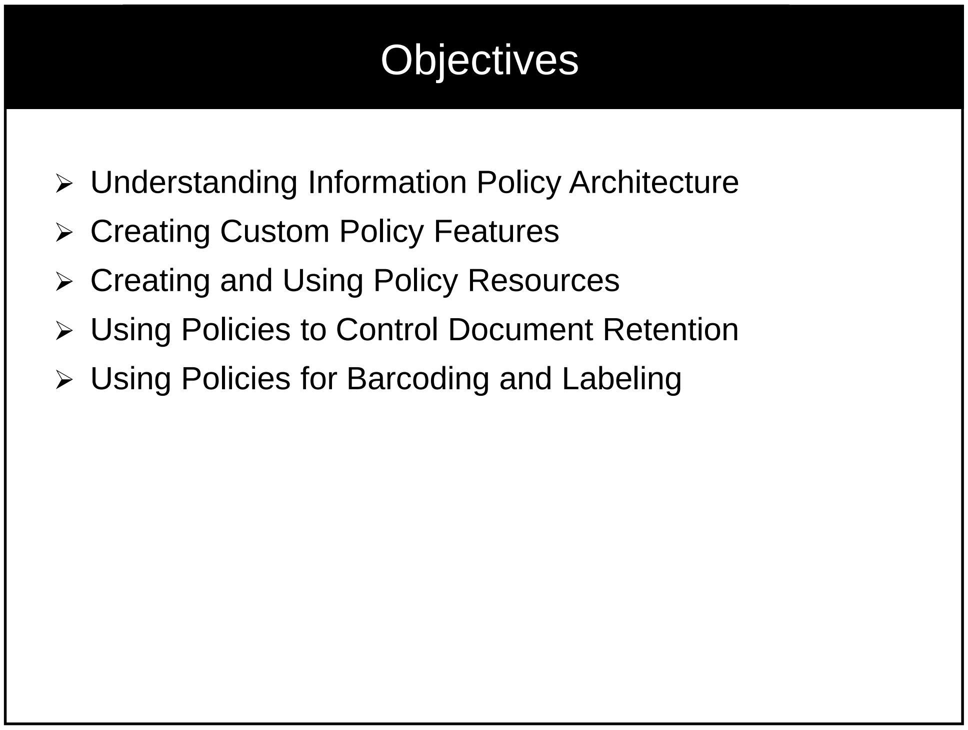

APP
1 Check-In Protected Document
WSS
Request document library GUID 3
4 Compare document library GUIDs, allow upload. 5
6
Process & store file
7
PROTECTOR
Extract and return document library GUID
Interact with RMS
Return 'unprotected' file stream
RMS



Summary

- Auditing is a core ECM requirement that applies equally to list items, documents, sites and site collections.
- Sharepoint includes extensive auditing features that are turned off by default to optimize storage and performance.
- MOSS includes user interface support for configuring audit settings at the site collection level and includes customizable audit reports.
- Information Rights Management and Windows Rights Management services provide comprehensive control over access to content.
- IRM/RMS extends access controls from the server environment down to the desktop.

SharePoint 2007 Information Management Policy



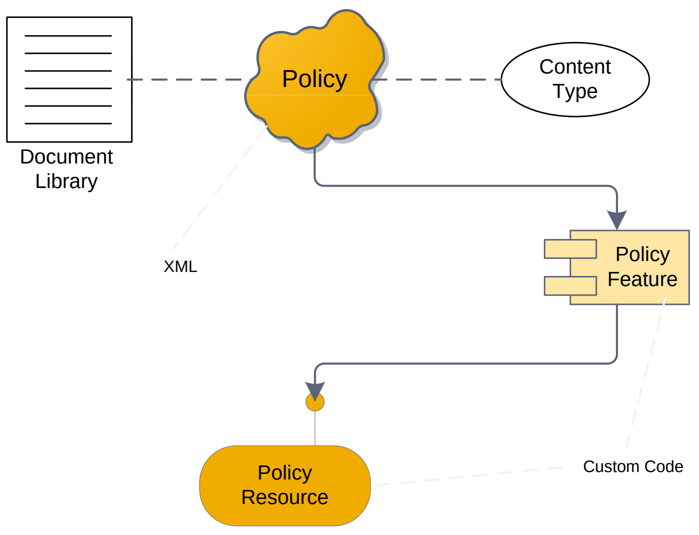
Objectives

- Understanding Information Policy Architecture
- Creating Custom Policy Features
- Creating and Using Policy Resources
- Using Policies to Control Document Retention
- Using Policies for Barcoding and Labeling

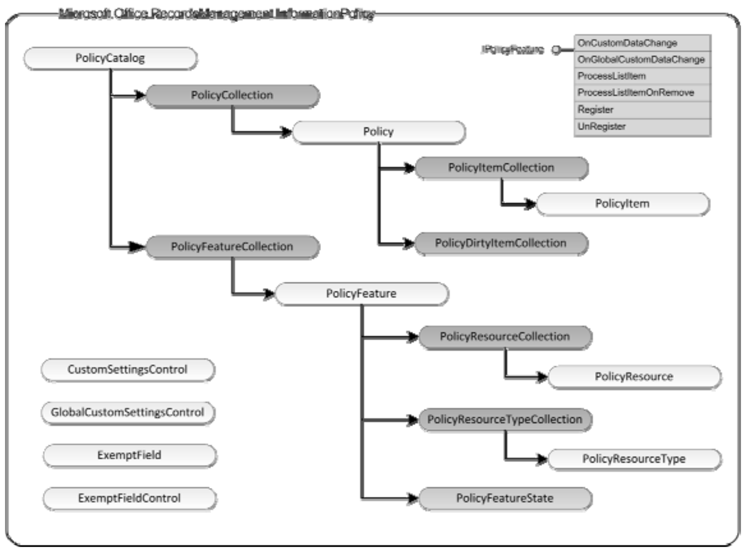
What is an Information Policy?

- A non-invasive way to manage content
- A collection of policy items (policy features)
- Can be created at two scope levels:
  - Site Collection Level
    - •Acts as a template for instances
    - •Copied into lists and content types when assigned
    - •MOSS propagates changes from template to instances
  - Content Type or List Level
    - •Attached directly to the object, without creating a template
    - •Can later be exported.

Information Policy Architecture



Information Policy API



Policy Features

- What is a policy feature?
  - An assembly that adds content management functionality to…
    - •MOSS, and optionally Office Clients
  - Implements IPolicyFeature interface
- Installed into global catalog
  - Assembly must be in the GAC
- May include custom administrative controls
  - Global Controls
    - •in \12\TEMPLATE\ADMIN
  - Item-Level Controls
    - •in \12\TEMPLATE\LAYOUTS

IPolicyFeature

- Purpose
  - To handle changes in a policy
  - To propagate policy changes to list items
  - To handle removal of policy from list items
  - To register/unregister a policy item
- Methods
  - OnCustomDataChange
  - OnGlobalCustomDataChange
  - ProcessListItem
  - ProcessListItemOnRemove
  - Register/Unregister

Built-In Policy Features

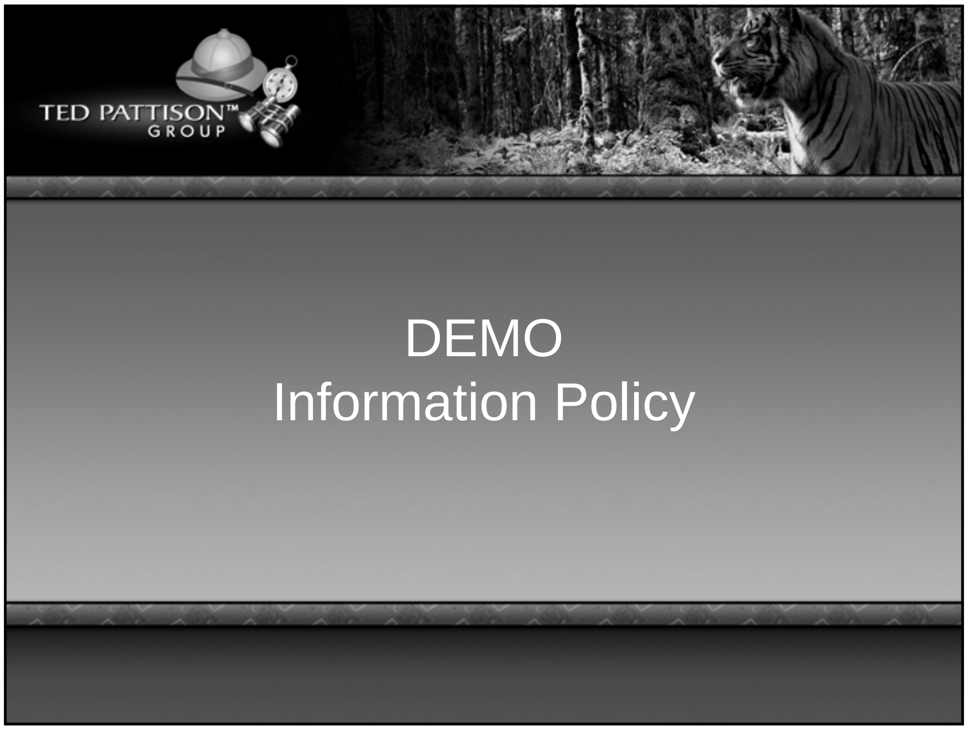
- Barcode
  - Creates a unique identifier for a document
  - Generates a barcode image and inserts into document
- Labeling
  - Searchable text area containing selected metadata
  - Provides a way to display metadata for printed documents
  - Can be used for list items as well as documents
- Auditing
  - Provides a way to control auditing via information policy
- Expiration
  - Enables the use of formulas to determine item expiration
  - Can specify actions to take when items expire

Policy Resources

- Custom code that is called by a given set of policy features
- Typically contains feature-specific code
- May implement custom interfaces that policy features must know how to use
- Deployed by adding definition (XML) to the policy resource list of a feature

Policy Event Processing

- If policy item is added…
  - → Register
- If policy item is removed…
  - → Unregister
- If policy definition changes,
  or a new policy is attached to the list…
  - → ProcessListItem
- If policy feature settings change…
  - → OnCustomDataChange
  - → OnGlobalCustomDataChange



DEMO
Information Policy

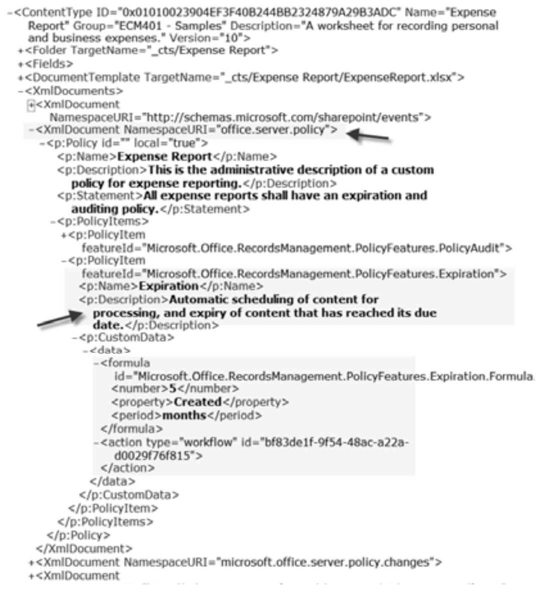
Policies in Content Types

- Policies can be assigned to content types
  - Cannot change policy while assigned
- Content Types inherit policies
  - Happens at instance level (when derived type is created)
  - Policy definition (XML) is copied into derived type
  - Changes to parent policy are propagated
- Policy definition stored in content type payload
  - XMLDocuments
- Policy definition copied into Office documents

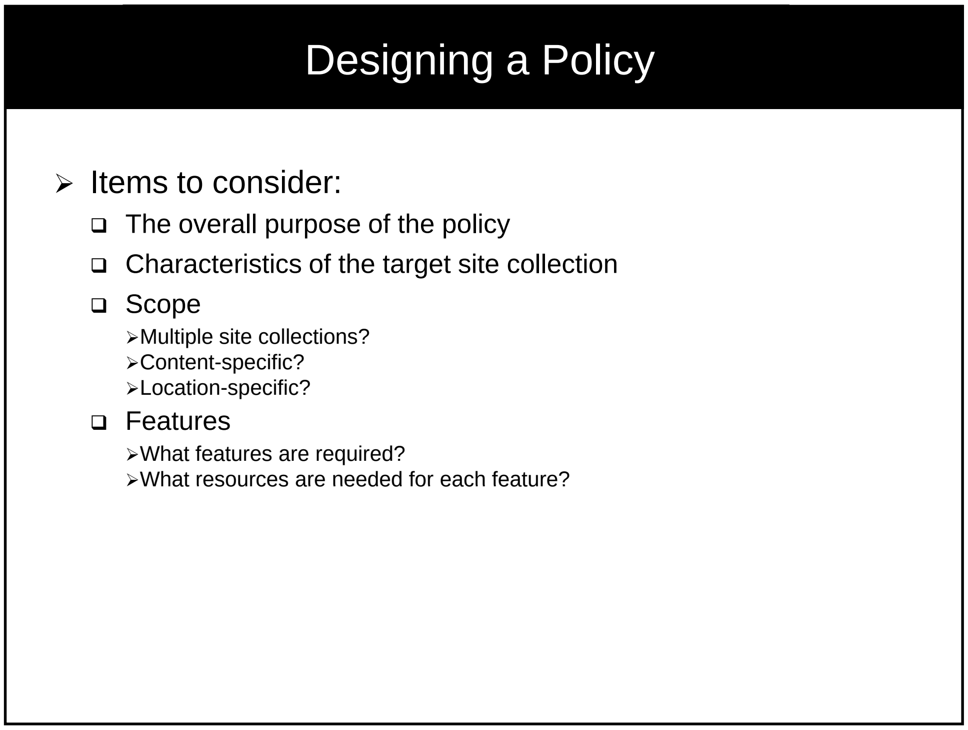
Assigning Policies to Content Types

- Cannot assign policies to ‘core’ content types
  - Item, Document, etc.
- Must derive a new content type to add a policy

Policy Definitions in Content Types

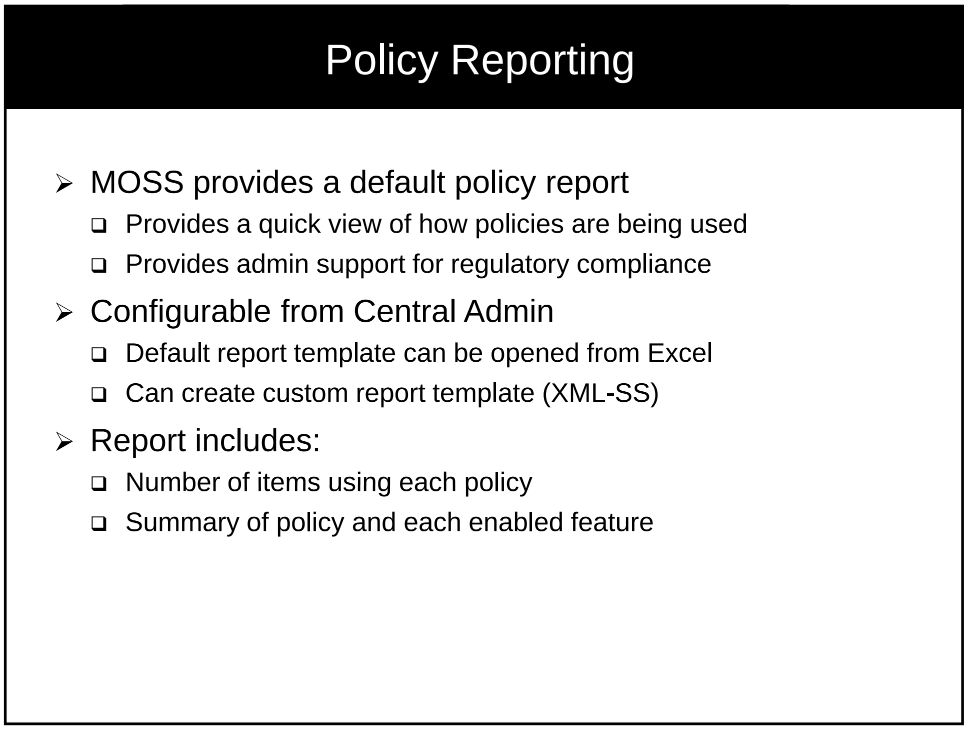


```
-<ContentType ID="0x01010023904EF3F40B244BB2324879A29B3ADC" Name="Expense
   Report" Group="ECM401 - Samples" Description="A worksheet for recording personal
   and business expenses." Version="10">
 +<Folder TargetName="_cts/Expense Report">
 +<Fields>
 +<DocumentTemplate TargetName="_cts/Expense Report/ExpenseReport.xlsx">
 -<XmlDocuments>
   +<XmlDocument
      NamespaceURI="http://schemas.microsoft.com/sharepoint/events">
   -<XmlDocument NamespaceURI="office.server.policy">
     -<p:Policy id="" local="true">
        <p:Name>Expense Report</p:Name>
        <p:Description>This is the administrative description of a custom
          policy for expense reporting.</p:Description>
        <p:Statement>All expense reports shall have an expiration and
          auditing policy.</p:Statement>
       -<p:PolicyItems>
         +<p:PolicyItem
            featureId="Microsoft.Office.RecordsManagement.PolicyFeatures.PolicyAudit">
         -<p:PolicyItem
            featureId="Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration">
           <p:Name>Expiration</p:Name>
           <p:Description>Automatic scheduling of content for
             processing, and expiry of content that has reached its due
             date.</p:Description>
          -<p:CustomData>
            -<data>
              -<formula
                 id="Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Formula
                <number>5</number>
                <property>Created</property>
                <period>months</period>
              </formula>
              -<action type="workflow" id="bf83de1f-9f54-48ac-a22a-
                 d0029f76f815">
              </action>
            </data>
          </p:CustomData>
        </p:PolicyItem>
      </p:PolicyItems>
    </p:Policy>
  </XmlDocument>
 +<XmlDocument NamespaceURI="microsoft.office.server.policy.changes">
 +<XmlDocument
```



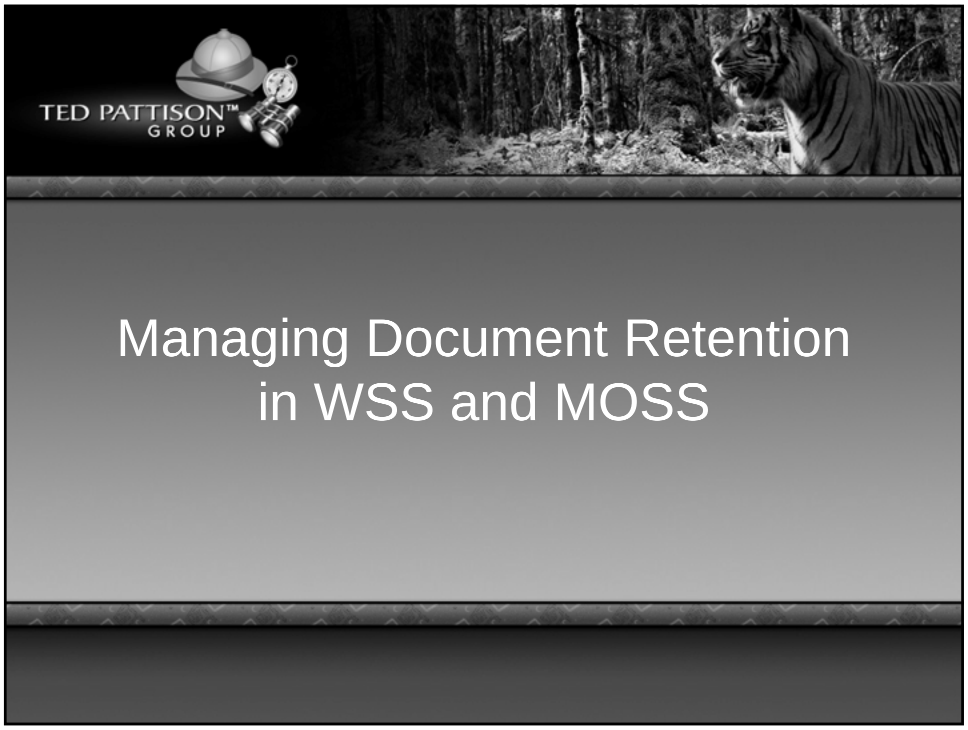
Designing a Policy

- Items to consider:
  - The overall purpose of the policy
  - Characteristics of the target site collection
  - Scope
    - Multiple site collections?
    - Content-specific?
    - Location-specific?
  - Features
    - What features are required?
    - What resources are needed for each feature?

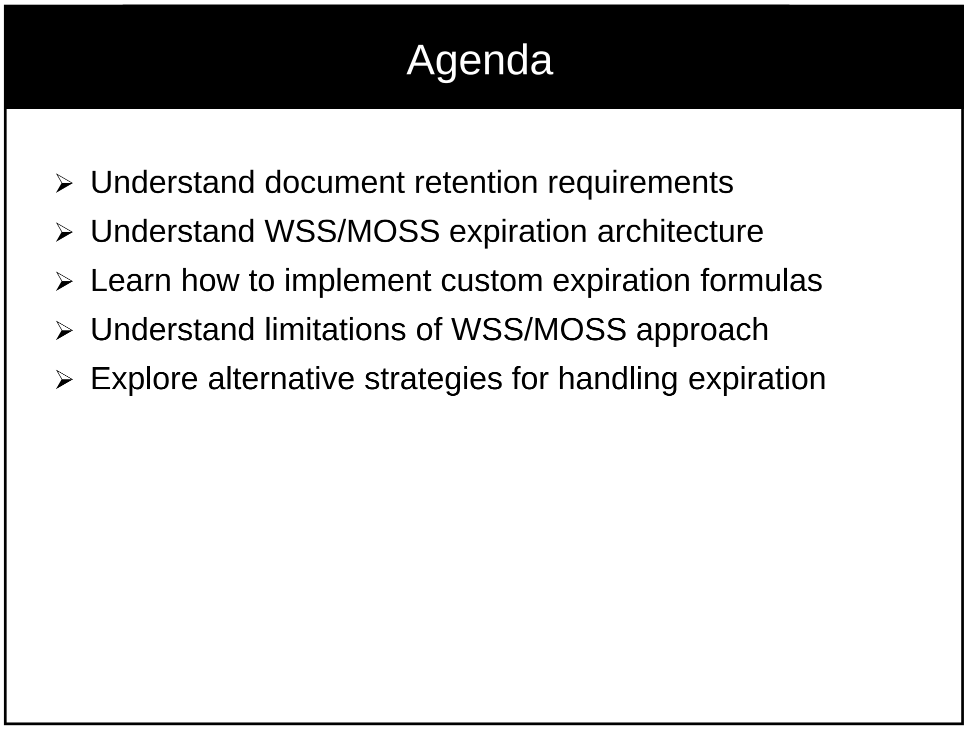


Policy Reporting

- MOSS provides a default policy report
  - Provides a quick view of how policies are being used
  - Provides admin support for regulatory compliance
- Configurable from Central Admin
  - Default report template can be opened from Excel
  - Can create custom report template (XML-SS)
- Report includes:
  - Number of items using each policy
  - Summary of policy and each enabled feature

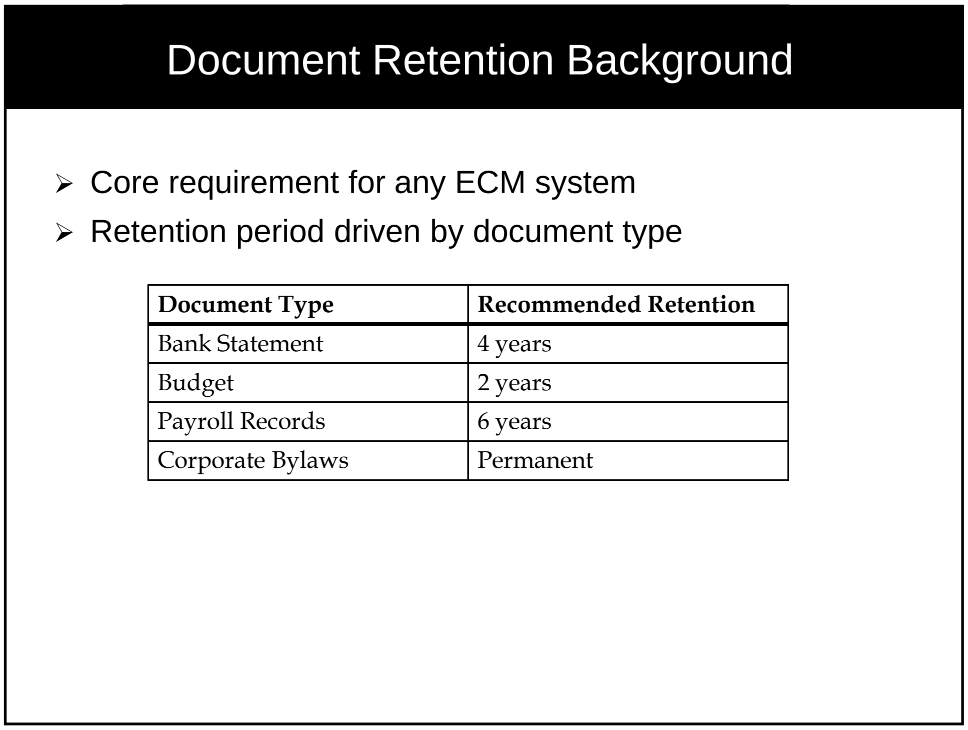


Managing Document Retention in WSS and MOSS



Agenda

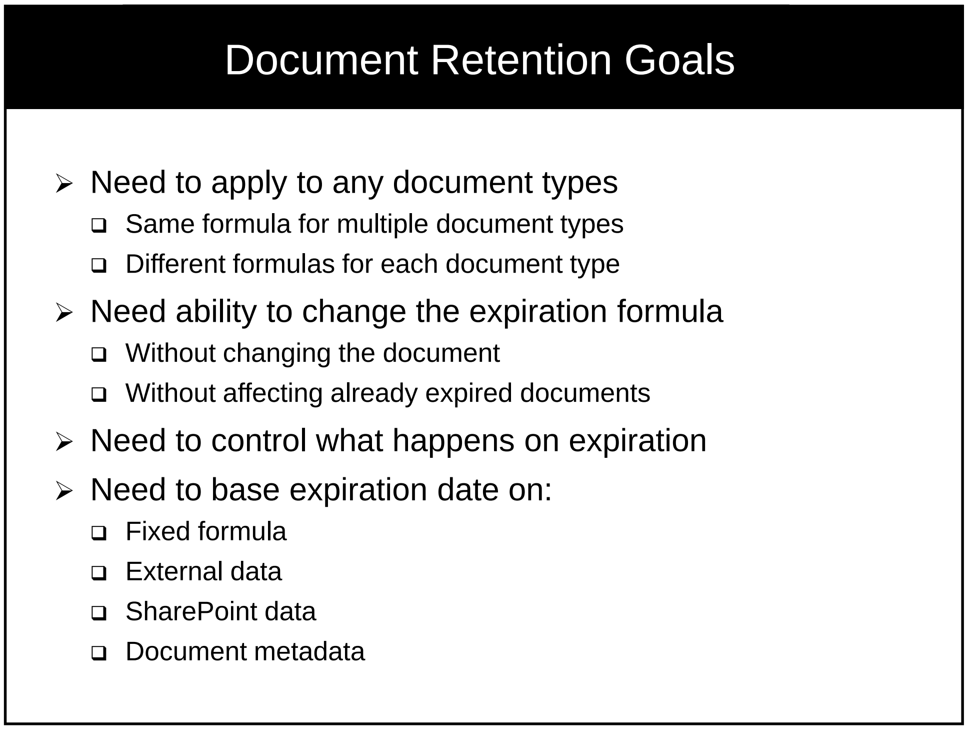
- Understand document retention requirements
- Understand WSS/MOSS expiration architecture
- Learn how to implement custom expiration formulas
- Understand limitations of WSS/MOSS approach
- Explore alternative strategies for handling expiration



Document Retention Background

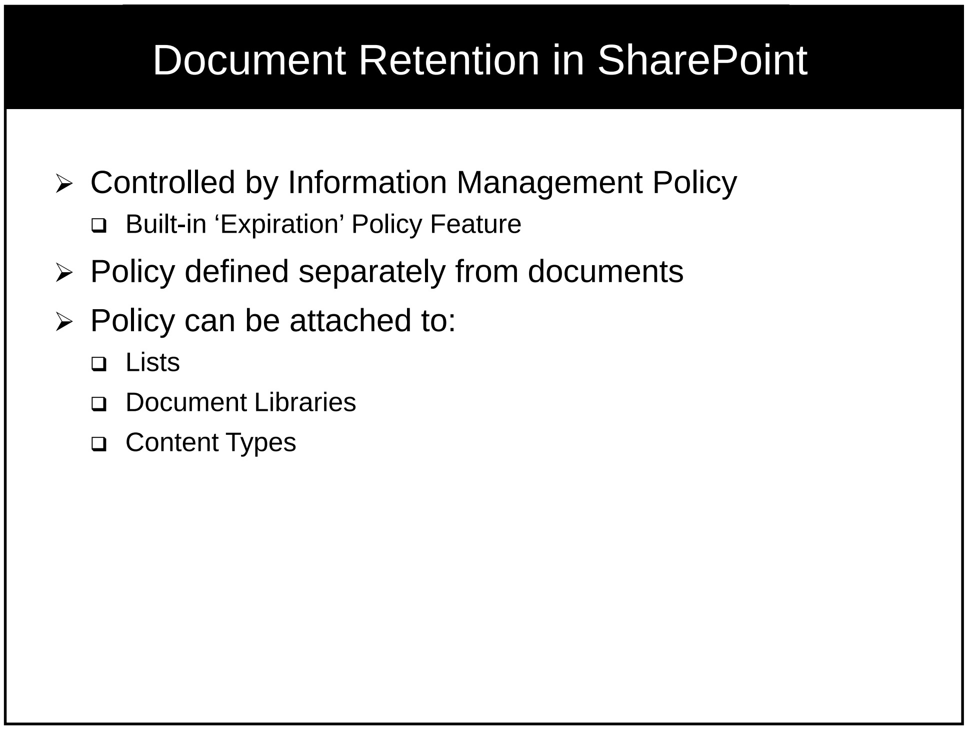
- Core requirement for any ECM system
- Retention period driven by document type

Document Type	Recommended Retention
Bank Statement	4 years
Budget	2 years
Payroll Records	6 years
Corporate Bylaws	Permanent



Document Retention Goals

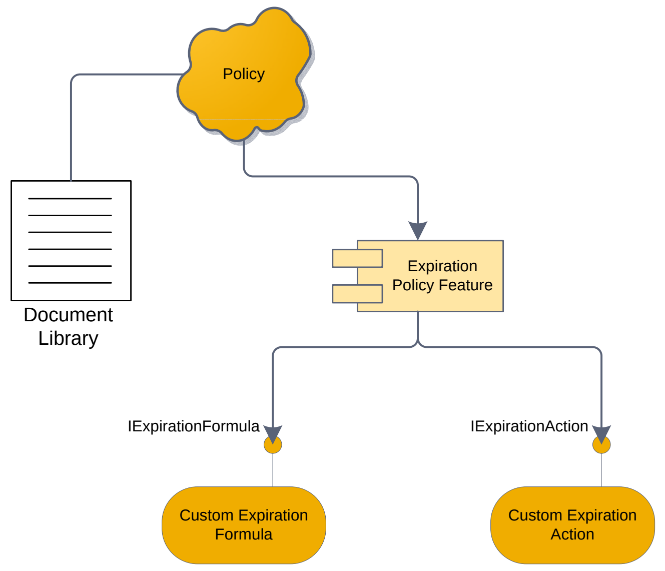
- Need to apply to any document types
  - Same formula for multiple document types
  - Different formulas for each document type
- Need ability to change the expiration formula
  - Without changing the document
  - Without affecting already expired documents
- Need to control what happens on expiration
- Need to base expiration date on:
  - Fixed formula
  - External data
  - SharePoint data
  - Document metadata



Document Retention in SharePoint

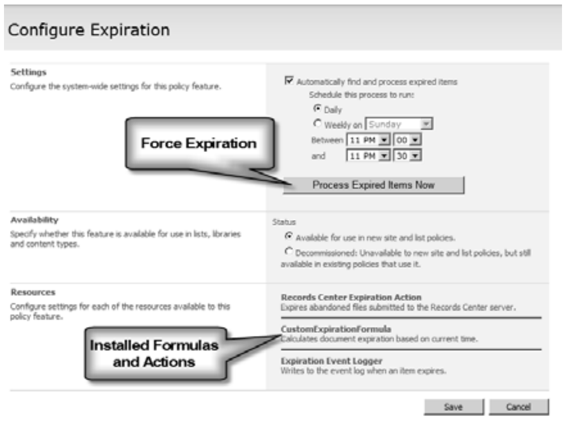
- Controlled by Information Management Policy
  - Built-in ‘Expiration’ Policy Feature
- Policy defined separately from documents
- Policy can be attached to:
  - Lists
  - Document Libraries
  - Content Types

Expiration Policy Feature



Expiration Policy Feature

- Implemented as a SharePoint Timer Job
- Configurable by Administrator
  - Can be forced to run:
    - via Central Admin
    - via custom code



Configure Expiration

Settings
Configure the system-wide settings for this policy feature.

[✓] Automatically find and process expired items
Schedule this process to run:
[✓ Daily]
Weekly on Sunday
Between 11 PM 00
and 11 PM 30

Availability
Specify whether this feature is available for use in lists, libraries and content types.

Status
[✓ Available for use in new site and list policies.]
Decommissioned: Unavailable to new site and list policies, but still available in existing policies that use it.

Resources
Configure settings for each of the resources available to this policy feature.

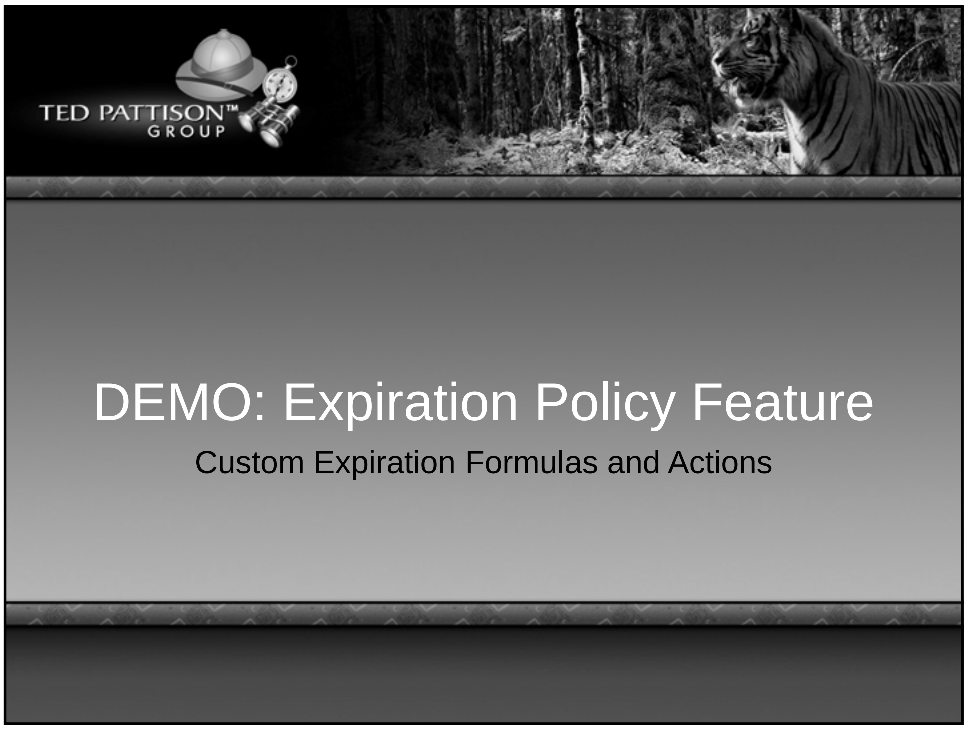
Records Center Expiration Action
Expires abandoned files submitted to the Records Center server.

CustomExpirationFormula
Calculates document expiration based on current time.

Expiration Event Logger
Writes to the event log when an item expires.

Forcing Expiration via Code

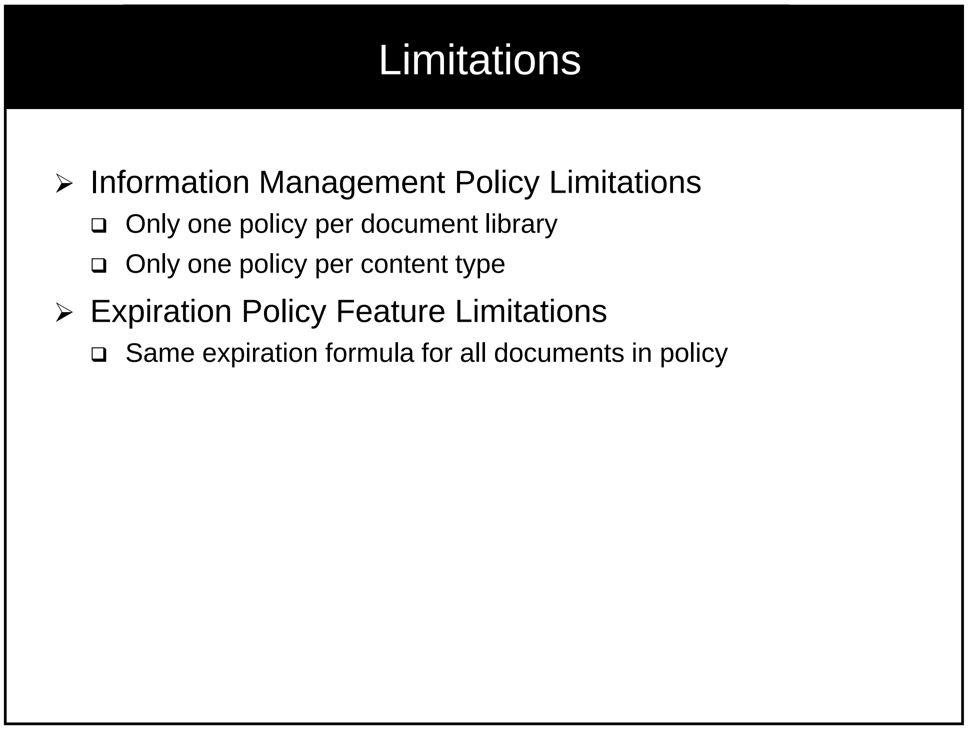
```
foreach (SPService service in SPFarm.Local.Services)
   foreach (SPJobDefinition job in service.JobDefinitions)
       if (job.Title == “Expiration policy”)
           job.Execute(Guid.Empty);
```



TED PATTISON™ GROUP

DEMO: Expiration Policy Feature

Custom Expiration Formulas and Actions



Limitations

- Information Management Policy Limitations
  - Only one policy per document library
  - Only one policy per content type
- Expiration Policy Feature Limitations
  - Same expiration formula for all documents in policy

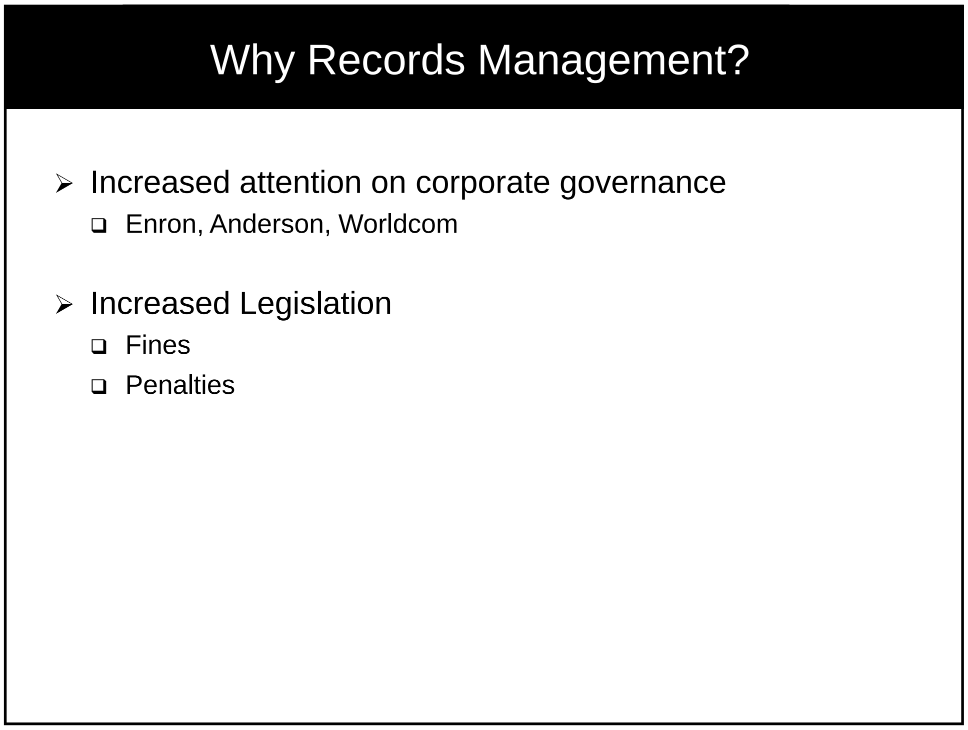
Solution Strategies

1. Don’t use Information Policy
   - ❑ Costly: must re-implement expiration timer job logic
   - ❑ Risky: may corrupt the content database
2. Don’t use expiration policy feature
   - ➢ Less costly: implement custom policy feature
   - ➢ Can access external expiration logic
   - ➢ Can access item metadata directly
3. Build a smarter expiration formula
   - ➢ Evaluate custom rules attached to item
   - ➢ Load and execute custom code configured at item level
     - ❑ ***AssemblyName + ClassName***

Summary

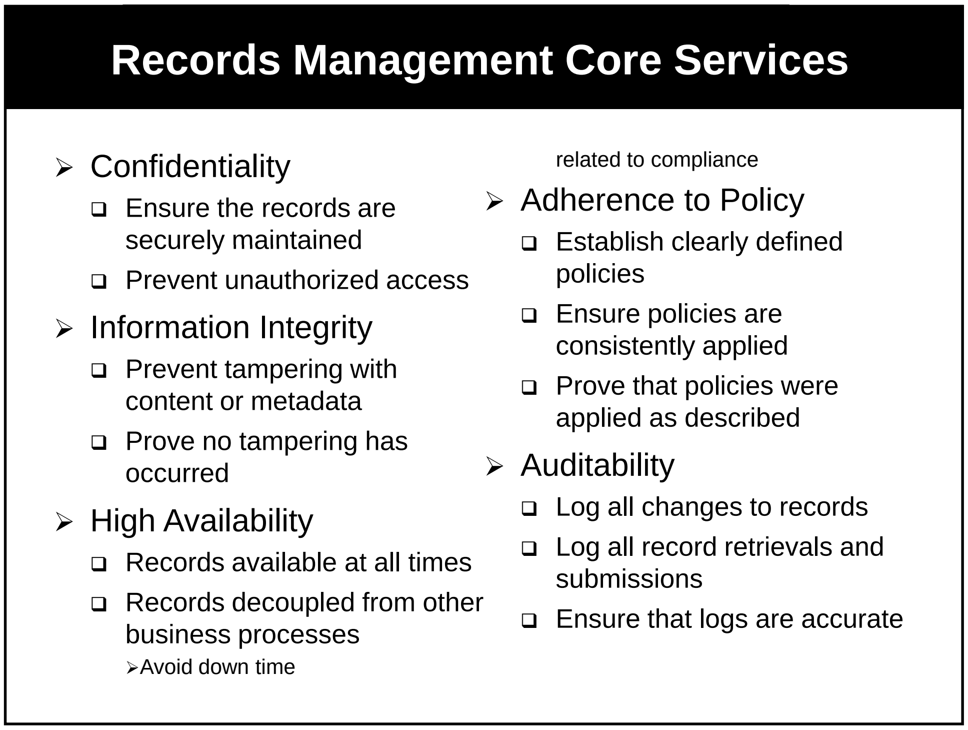
- ➢ WSS/MOSS provides an extensible framework for defining and applying custom information policies.
- ➢ The WSS/MOSS Information Management Policy architecture is limited by the restriction of one policy per document library or content type.
- ➢ The WSS/MOSS expiration policy framework is further limited by the common requirement to apply expiration policies along with other metadata-driven policies.
- ➢ One strategy for overcoming such limitations is to build a secondary framework that evaluates custom rules attached to each document.

Managing Official Records using SharePoint Server 2007



Why Records Management?

- Increased attention on corporate governance
  - Enron, Anderson, Worldcom
- Increased Legislation
  - Fines
  - Penalties

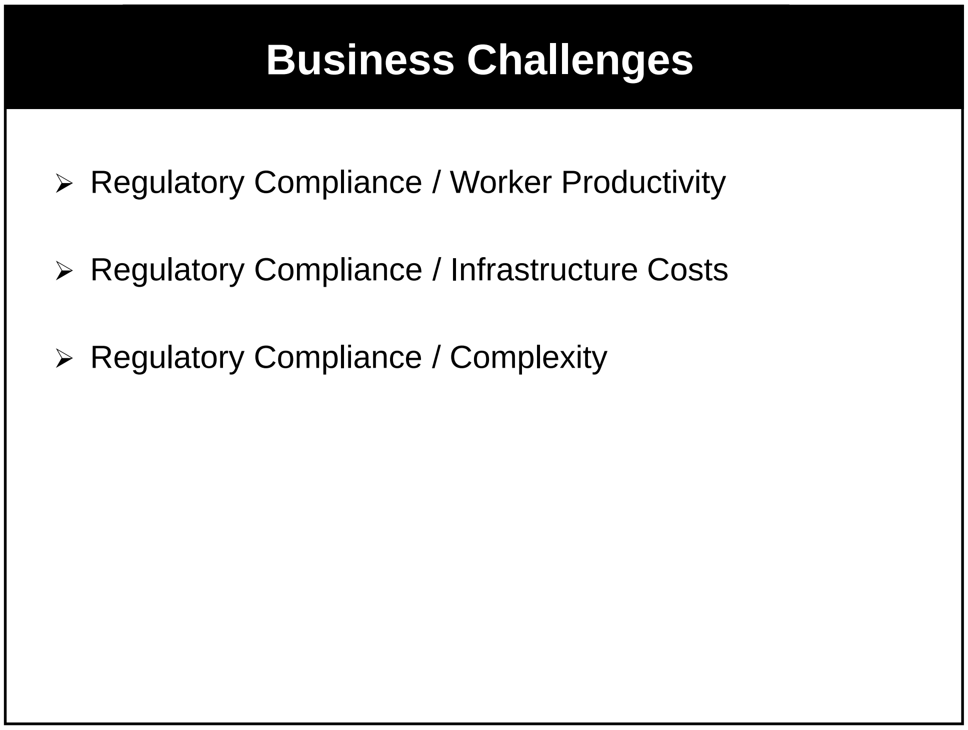


Records Management Core Services

- Confidentiality
  - Ensure the records are securely maintained
  - Prevent unauthorized access
- Information Integrity
  - Prevent tampering with content or metadata
  - Prove no tampering has occurred
- High Availability
  - Records available at all times
  - Records decoupled from other business processes
    - Avoid down time

related to compliance

- Adherence to Policy
  - Establish clearly defined policies
  - Ensure policies are consistently applied
  - Prove that policies were applied as described
- Auditability
  - Log all changes to records
  - Log all record retrievals and submissions
  - Ensure that logs are accurate



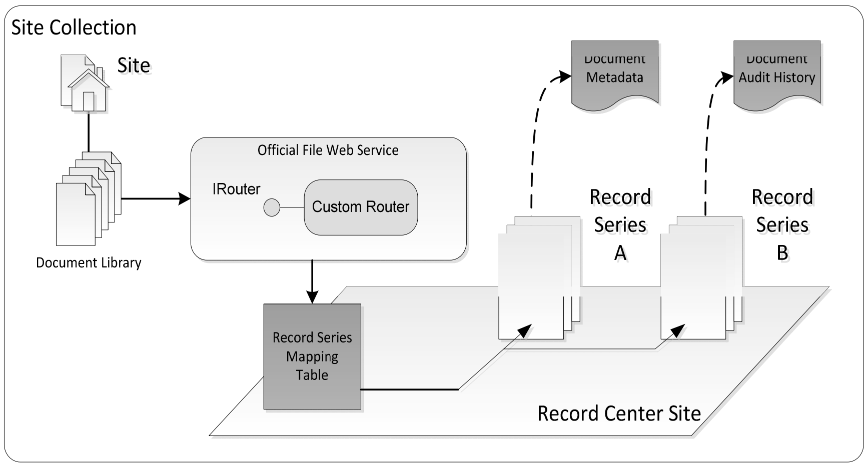
Business Challenges

- Regulatory Compliance / Worker Productivity
- Regulatory Compliance / Infrastructure Costs
- Regulatory Compliance / Complexity

Records Management in MOSS

- Records Repository Site Template
- Official Record Routing Infrastructure
- Official File Web Service
- Information Management Policy Architecture

How it Works

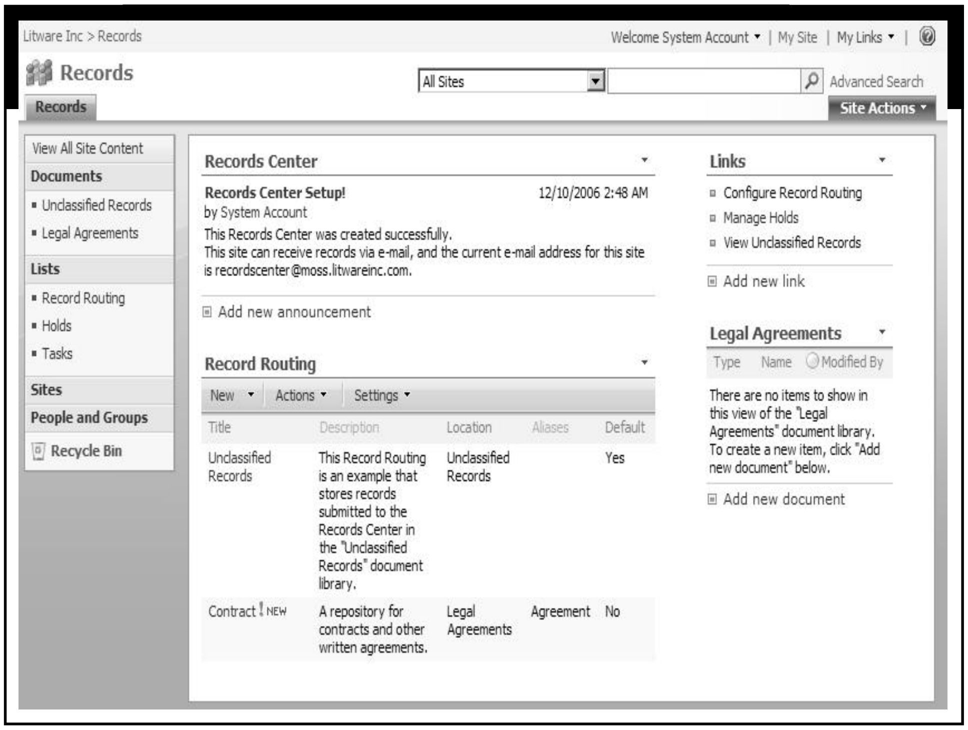


Processing Overview

- Document library configured to send records
- Official File WS called to get routing types
- Routing types mapped to content types
- File submitted with custom routing applied
- Metadata is promoted to document library
- Information management policies applied

Setting up a Records Repository

- Use the built-in Records Center site template
- Add a document library to receive files
- Setup record routing types



Records Center

Records Center Setup! 12/10/2006 2:48 AM
by System Account
This Records Center was created successfully.
This site can receive records via e-mail, and the current e-mail address for this site is recordscenter@moss.litwareinc.com.

Record Routing

Title	Description	Location	Aliases	Default
Unclassified Records	This Record Routing is an example that stores records submitted to the Records Center in the "Unclassified Records" document library.	Unclassified Records		Yes
Contract ! NEW	A repository for contracts and other written agreements.	Legal Agreements	Agreement	No

Legal Agreements

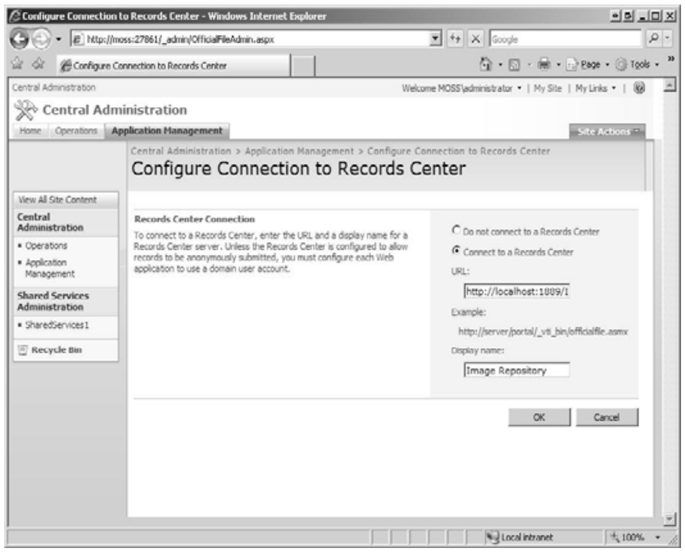
Type	Name	Modified By

There are no items to show in this view of the "Legal Agreements" document library. To create a new item, click "Add new document" below.

Submitting Records from the UI

- Configure the document library (admin)
- Send documents to repository (user)

Configure Document Library



Records Center Connection

To connect to a Records Center, enter the URL and a display name for a Records Center server. Unless the Records Center is configured to allow records to be anonymously submitted, you must configure each Web application to use a domain user account.

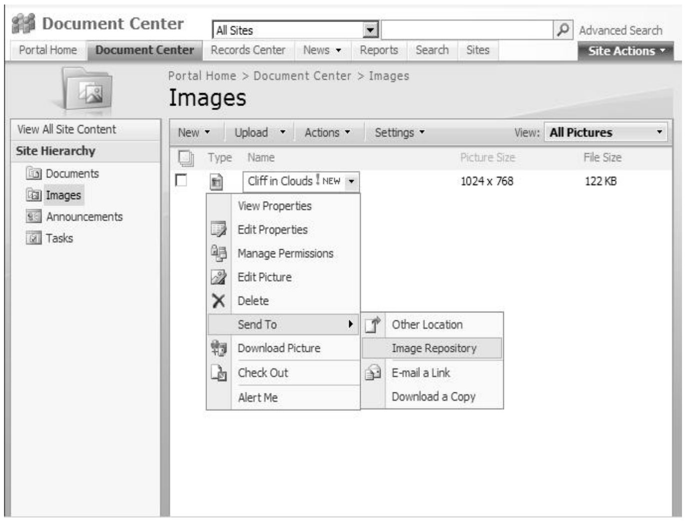
Do not connect to a Records Center [✓ Connect to a Records Center]

URL: http://localhost:1889/I

Example: http://server/portal/_vti_bin/officialfile.asmx

Display name: Image Repository

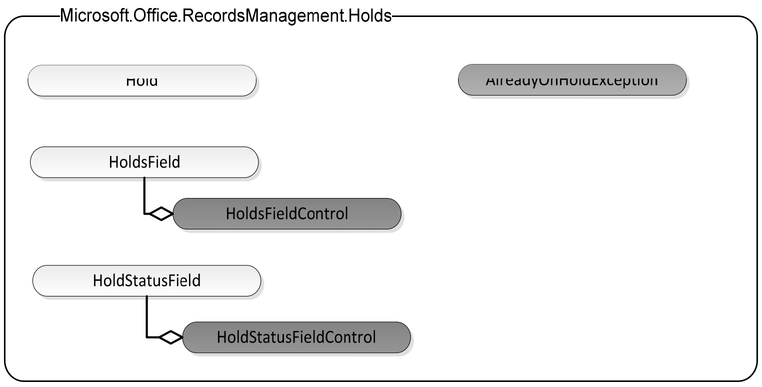
Send to Repository

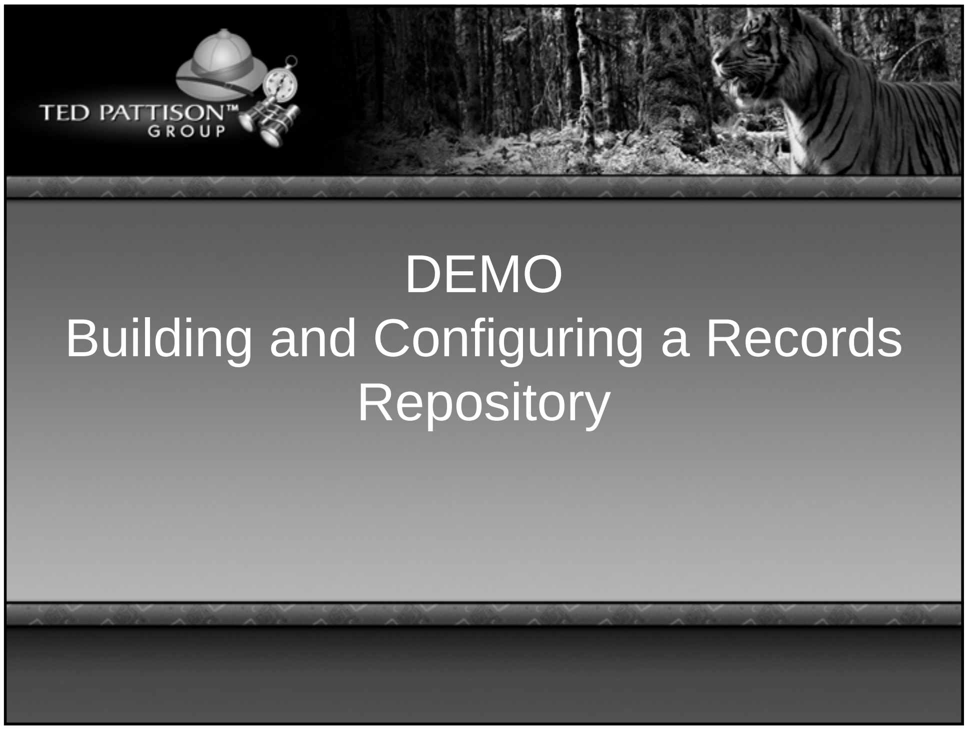


Holds

- Important for Litigation Support applications
  - Need to suspend normal processing
    - “Freeze” orders from court
    - Prevent files from being changed or deleted

Holds Architecture

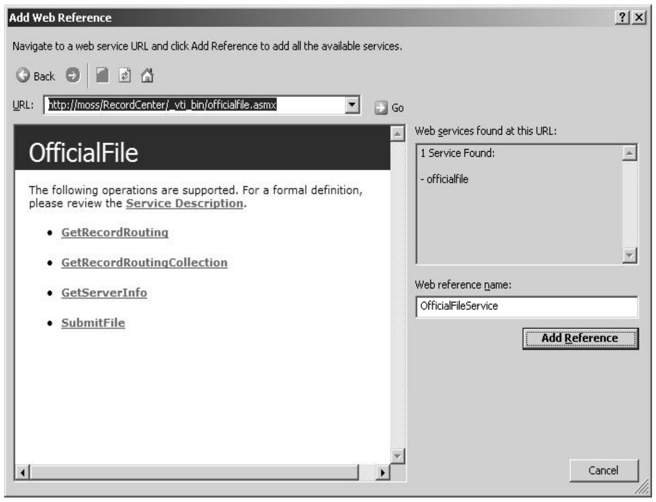




DEMO
Building and Configuring a Records Repository

Using the SharePoint OfficialFile Web Service

Official File Web Service



Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://moss/RecordCenter/_vti_bin/officialfile.asmx

OfficialFile

The following operations are supported. For a formal definition, please review the Service Description.

- GetRecordRouting
- GetRecordRoutingCollection
- GetServerInfo
- SubmitFile

Web services found at this URL:

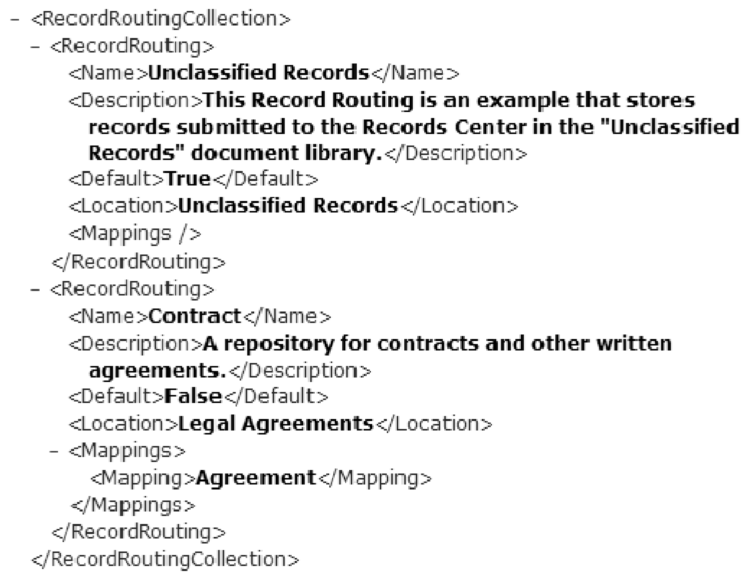
1 Service Found:

- officialfile

Web reference name: OfficialFileService

- Only 4 methods
- Relatively easy to implement

Record Routing Collection



```
- <RecordRoutingCollection>
  - <RecordRouting>
      <Name>Unclassified Records</Name>
      <Description>This Record Routing is an example that stores
        records submitted to the Records Center in the "Unclassified
        Records" document library.</Description>
      <Default>True</Default>
      <Location>Unclassified Records</Location>
      <Mappings />
    </RecordRouting>
  - <RecordRouting>
      <Name>Contract</Name>
      <Description>A repository for contracts and other written
        agreements.</Description>
      <Default>False</Default>
      <Location>Legal Agreements</Location>
    - <Mappings>
        <Mapping>Agreement</Mapping>
      </Mappings>
    </RecordRouting>
  </RecordRoutingCollection>
```

Submitting Records using Code

- Add a web reference to the Official File WS
- Connect to the repository
- Get the available record routing types
- Submit the file

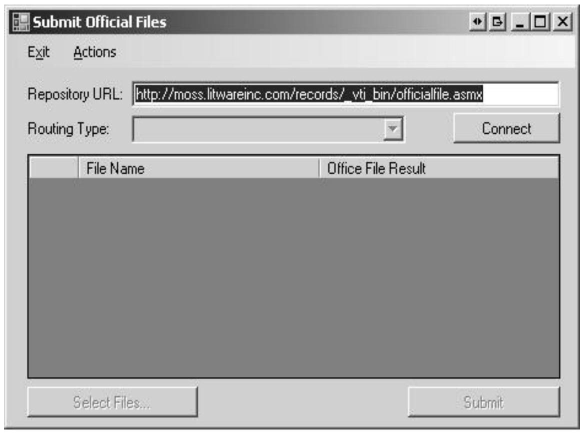
ValidateRepository()

```
/// <summary>
/// Determines whether the specified repository is available.
/// </summary>
private bool ValidateRepository(OfficialFileService.RecordsRepository repository)
{
    bool isValid = false;
    try {
        isValid = repository.GetRecordRoutingCollection().Length > 0;
    } catch {}
    return isValid;
}
```

Connecting to the Repository

```
// Get the web service proxy and point it at the designated url.
m_repository = new OfficialFileService.RecordsRepository();
m_repository.Credentials = System.Net.CredentialCache.DefaultCredentials;
m_repository.Url = repositoryUrl.Text;

if (ValidateRepository(m_repository))
{
    // Get the list of available record routing types.
    string rTypes = m_repository.GetRecordRoutingCollection();
    string expr = "/RecordRoutingCollection/RecordRouting/Name";
    XPathDocument doc = new XPathDocument(new StringReader(rTypes));
    XPathNavigator nav = doc.CreateNavigator();
    XPathNodeIterator iter = nav.Select(expr);

    // Initialize the list of routing types in the UI
    …
}
```

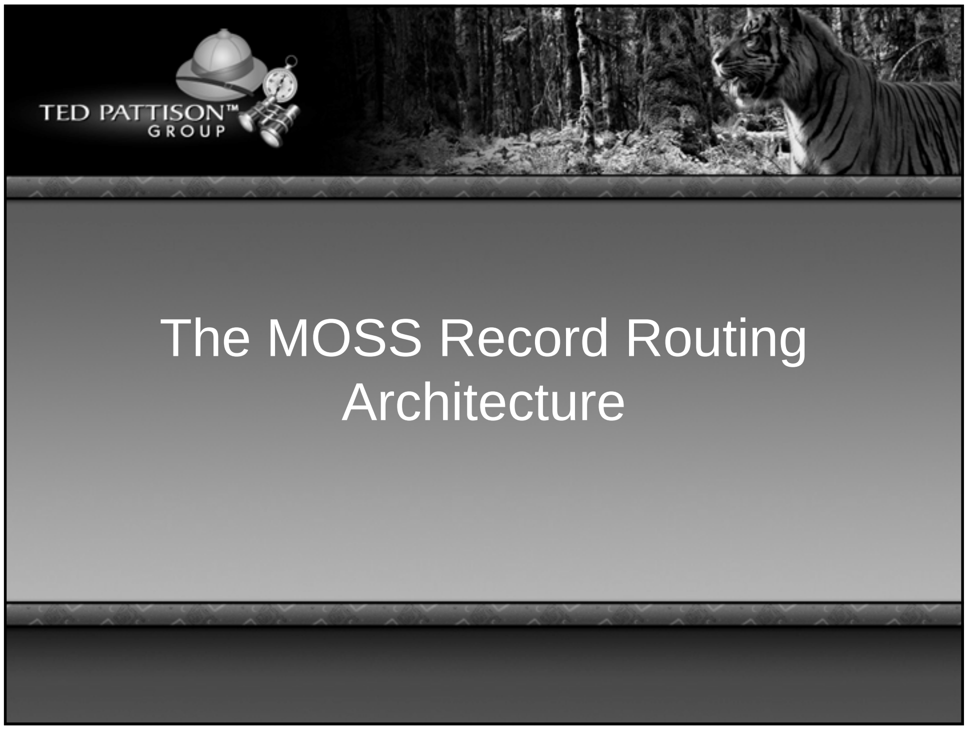
Official File Web Service Client



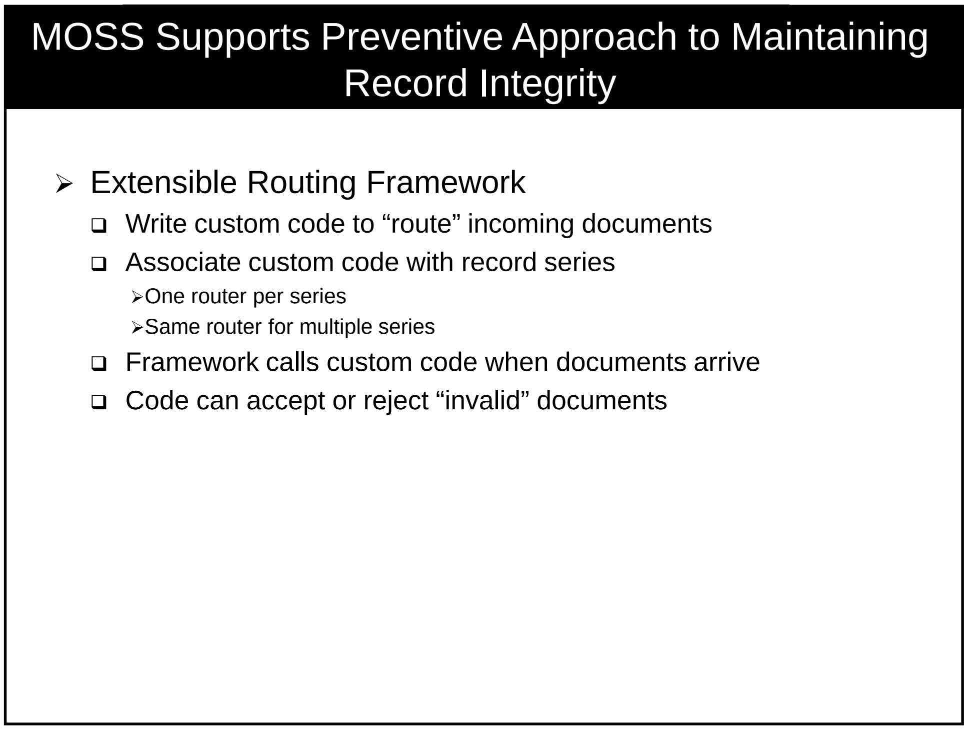
Repository URL: http://moss.litwareinc.com/records/_vti_bin/officialfile.asmx
Routing Type:

- An example of how to connect to a repository using the Official File web service

DEMO: Using the Official File Web Service

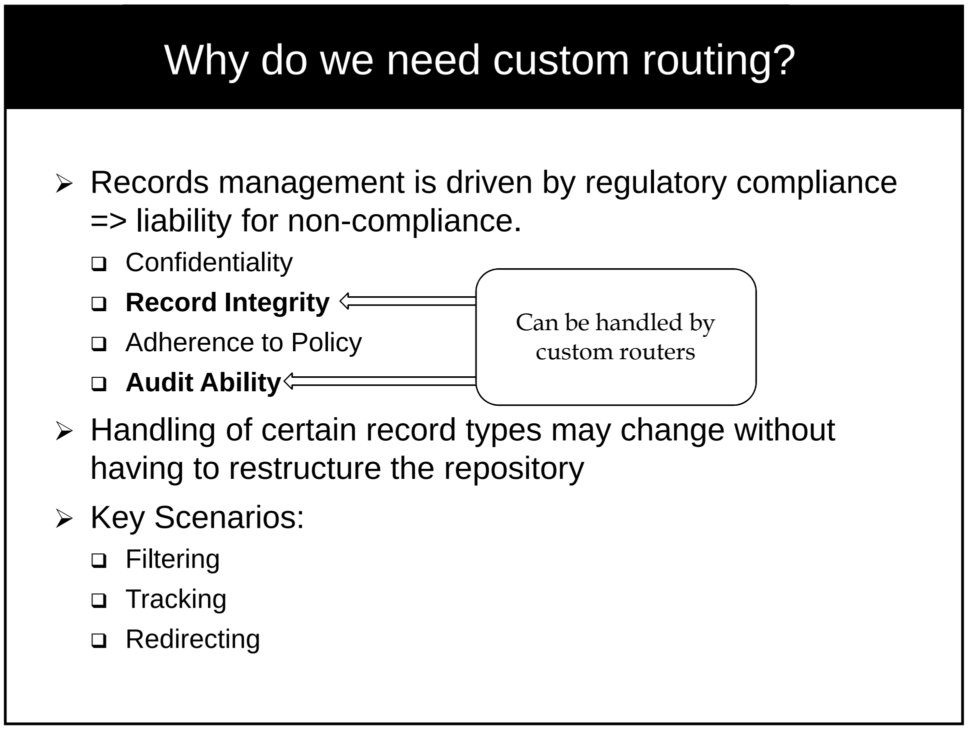


The MOSS Record Routing Architecture



MOSS Supports Preventive Approach to Maintaining Record Integrity

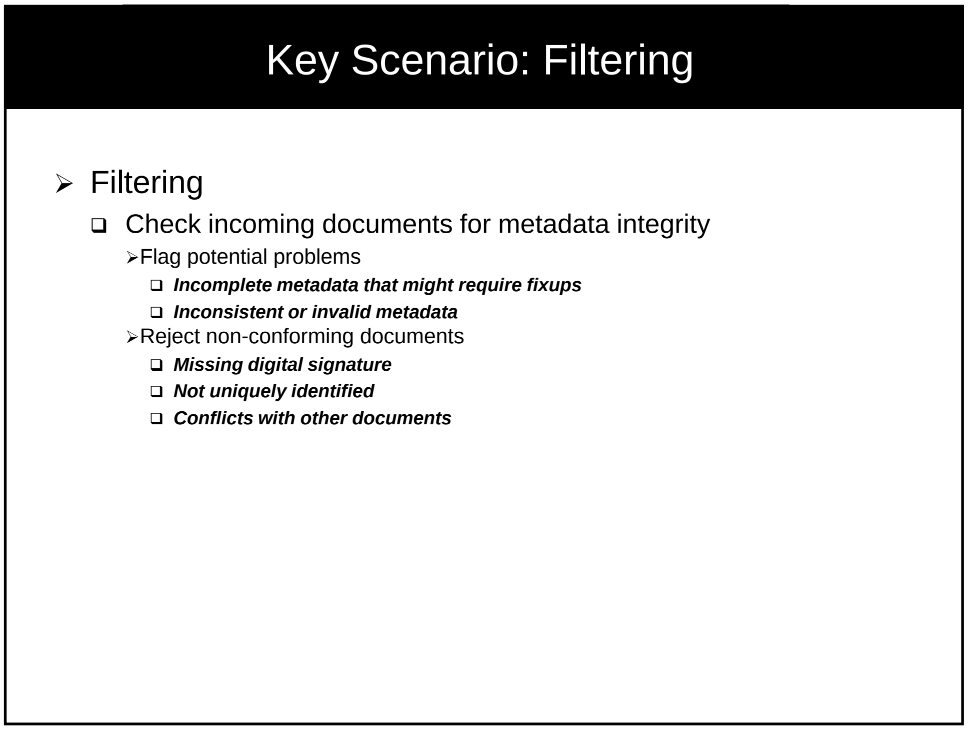
- Extensible Routing Framework
  - Write custom code to “route” incoming documents
  - Associate custom code with record series
    - One router per series
    - Same router for multiple series
  - Framework calls custom code when documents arrive
  - Code can accept or reject “invalid” documents



Why do we need custom routing?

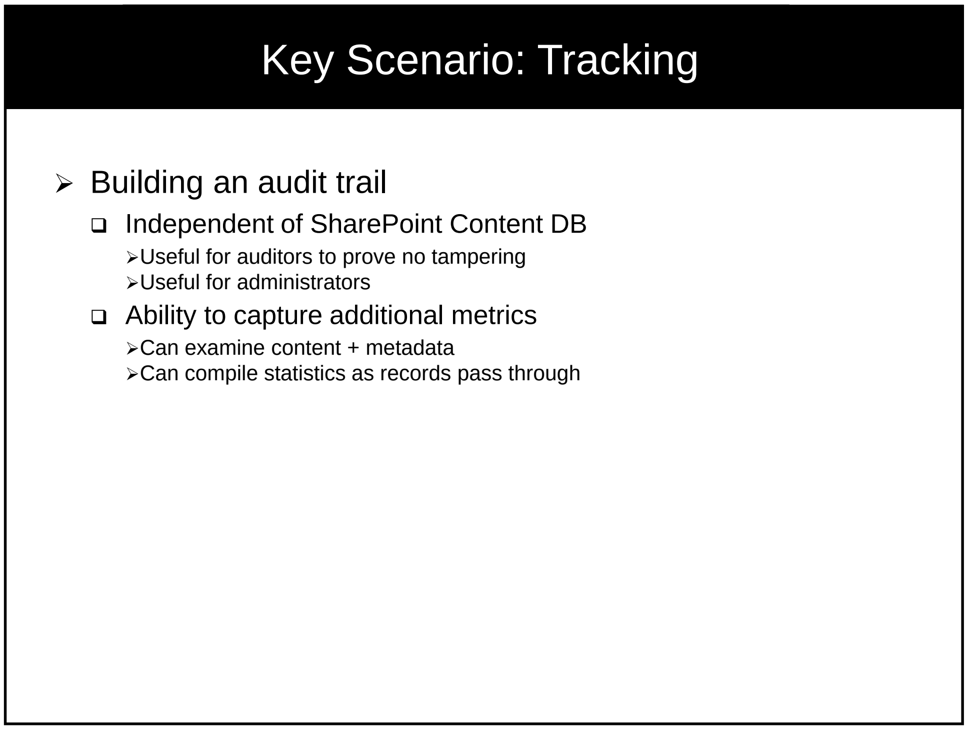
- Records management is driven by regulatory compliance => liability for non-compliance.
  - Confidentiality
  - **Record Integrity**
  - Adherence to Policy
  - **Audit Ability**
- Handling of certain record types may change without having to restructure the repository
- Key Scenarios:
  - Filtering
  - Tracking
  - Redirecting

Can be handled by custom routers



Key Scenario: Filtering

- Filtering
  - Check incoming documents for metadata integrity
    - Flag potential problems
      - ***Incomplete metadata that might require fixups***
      - ***Inconsistent or invalid metadata***
    - Reject non-conforming documents
      - ***Missing digital signature***
      - ***Not uniquely identified***
      - ***Conflicts with other documents***



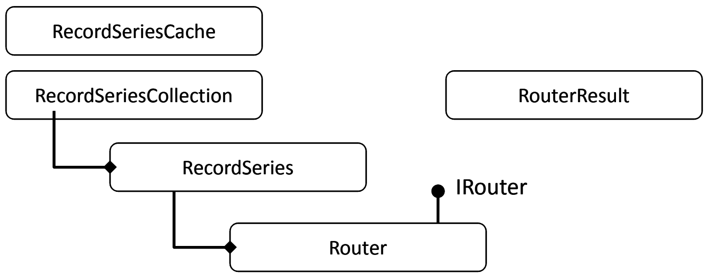
Key Scenario: Tracking

- Building an audit trail
  - Independent of SharePoint Content DB
    - Useful for auditors to prove no tampering
    - Useful for administrators
  - Ability to capture additional metrics
    - Can examine content + metadata
    - Can compile statistics as records pass through

Key Scenario: Redirecting

- Classic Scenario
  - Storing records dynamically
    - Choose location based on algorithm(s)
    - Driven by content and/or metadata

MOSS Record Routing Architecture

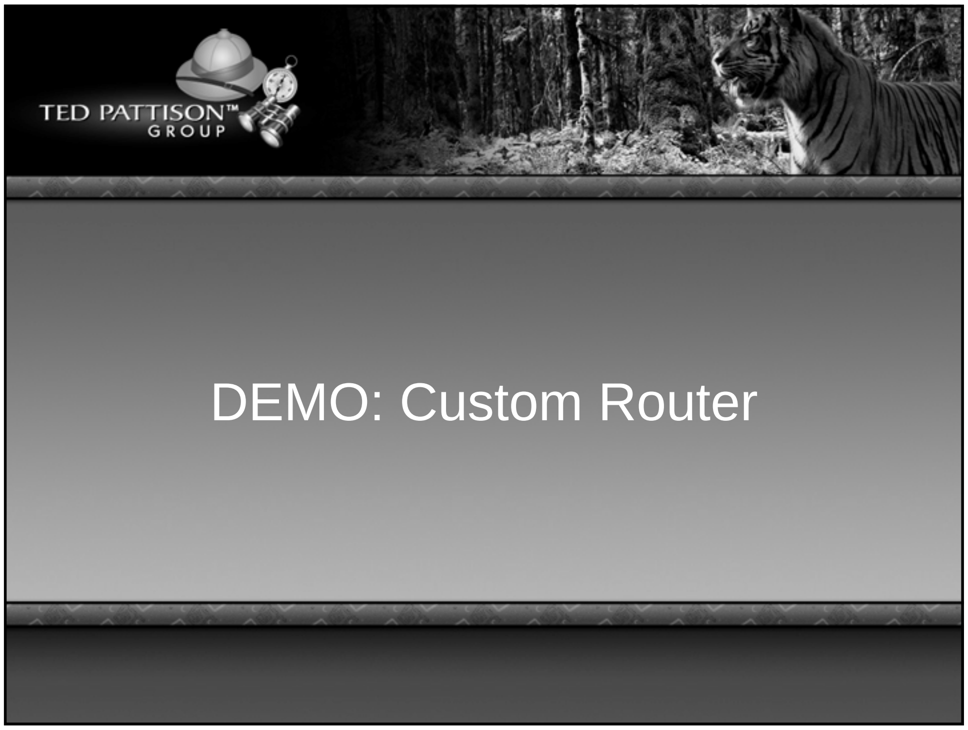


Creating a Custom Router

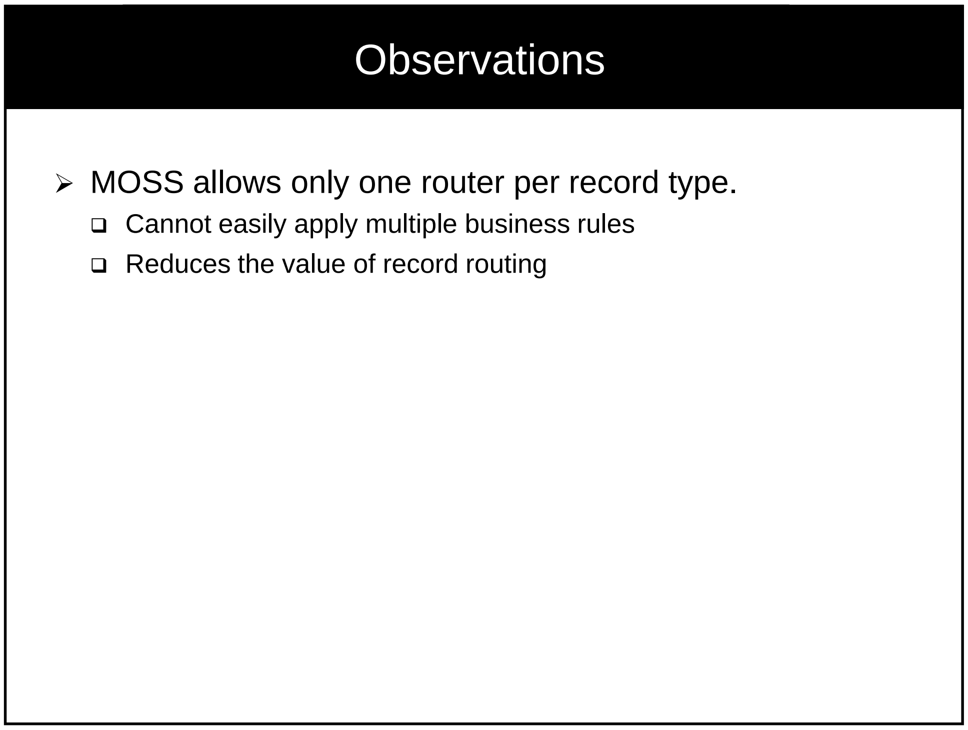
- Implement the IRouter interface
- Add the custom router to a records center
- Activate the router for a routing type

Router Configuration

```
using (SPSite site = new SPSite(url))  {
      using (SPWeb web = site.OpenWeb())  {
      try  {
                RecordSeriesCollection coll = new RecordSeriesCollection( web );
                switch (command)   {
                case "REGISTER":
                                coll.AddRouter( ROUTER_NAME,  ROUTER_ASSEMBLY,
                                                                ROUTER_CLASS);
                             break;
                case "UNREGISTER":
                                coll.RemoveRouter( ROUTER_NAME );
                             break;
                }
      } catch {}
}}
```



DEMO: Custom Router



Observations

- MOSS allows only one router per record type.
  - Cannot easily apply multiple business rules
  - Reduces the value of record routing

Routing Wish List

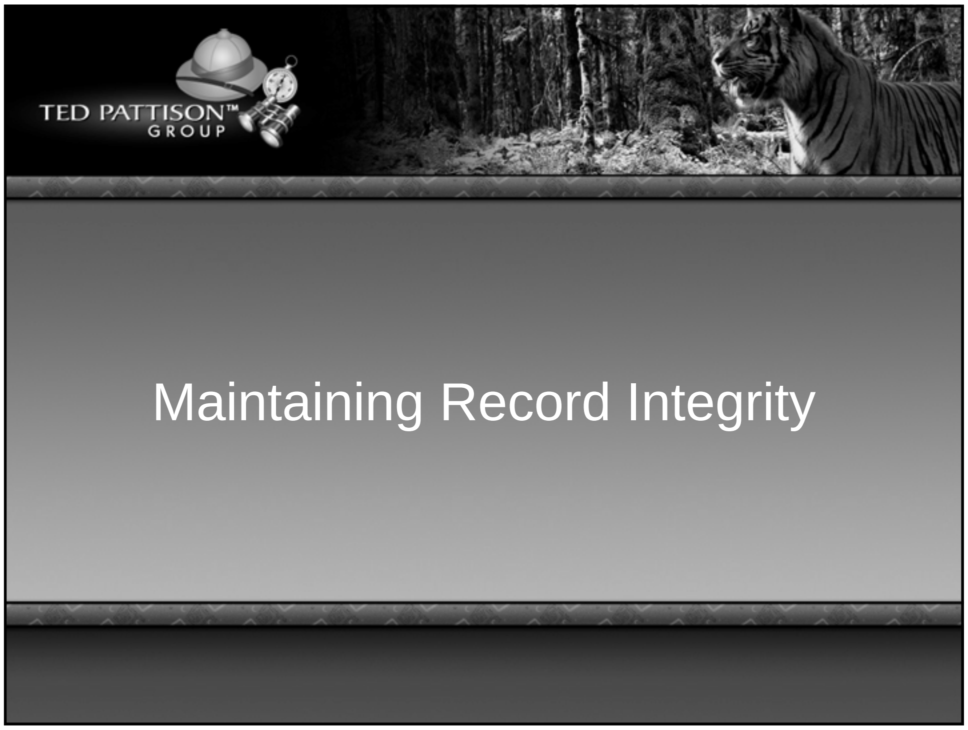
- More extensible routing framework
  - Ability to attach actions to incoming documents
  - Ability to attach multiple actions to the same document
  - Ability to add routing actions by activating a feature (installing an assembly)
  - Ability to administer routing actions

Summary

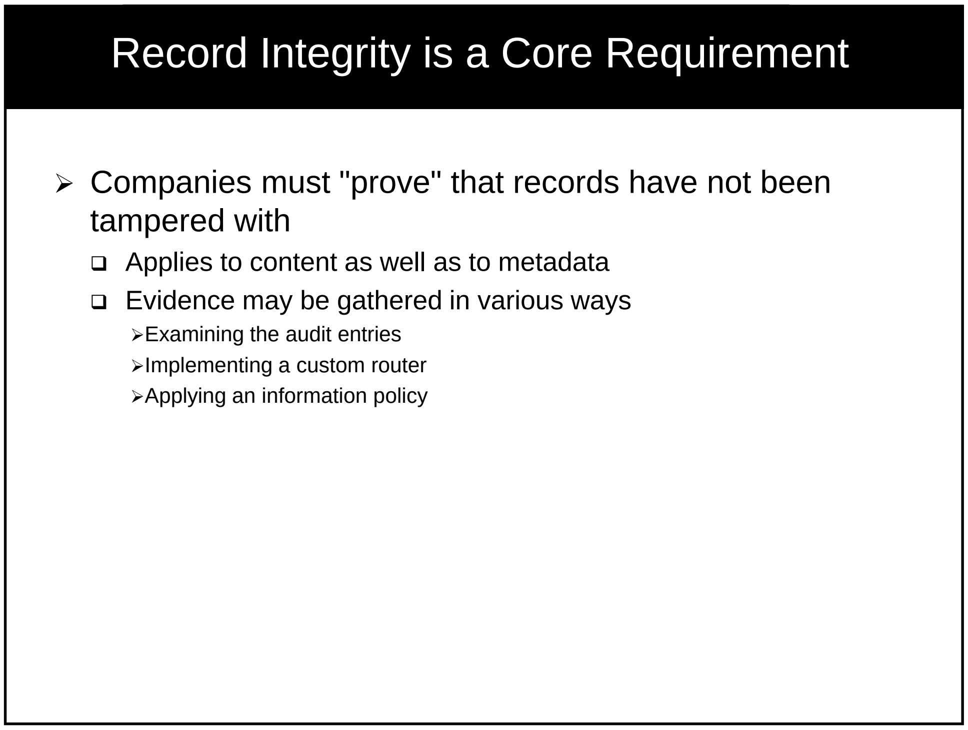
- Records management is driven by regulatory compliance
- An effective RM solution must meet core requirements of confidentiality, integrity, availability, adherence to policy and auditability
- MOSS includes a default implementation that meets these requirements

Summary

- The Official File web service provides a standards-based SOA platform for records management.
- MOSS implements an extensible routing framework for processing records.
- Information Management Policy enables governance of all aspects of enterprise content management.

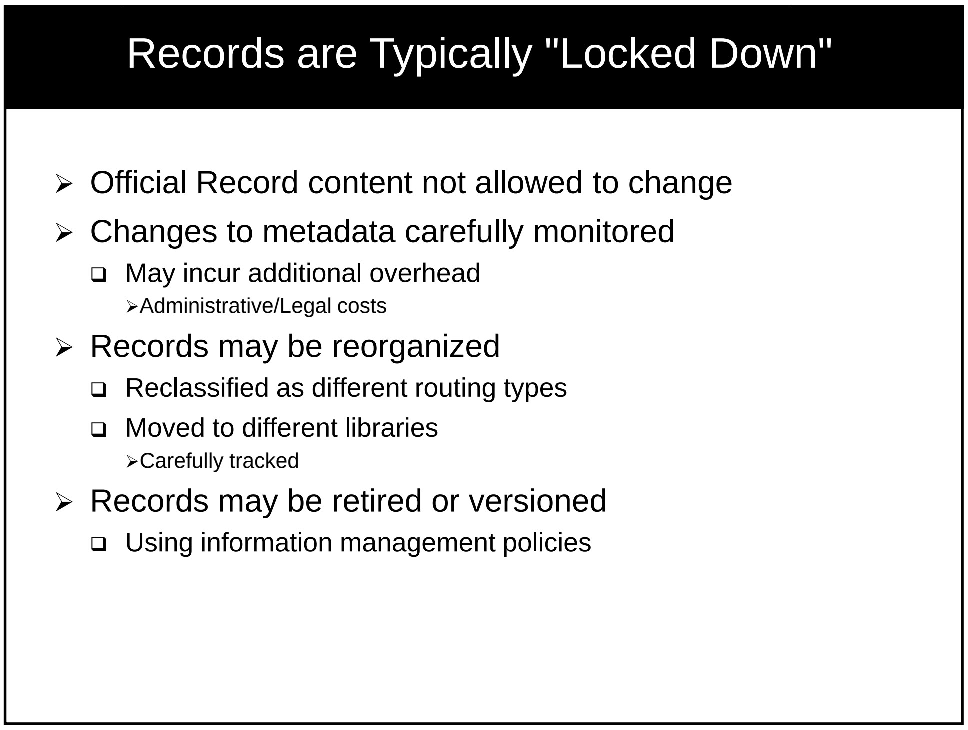


Maintaining Record Integrity



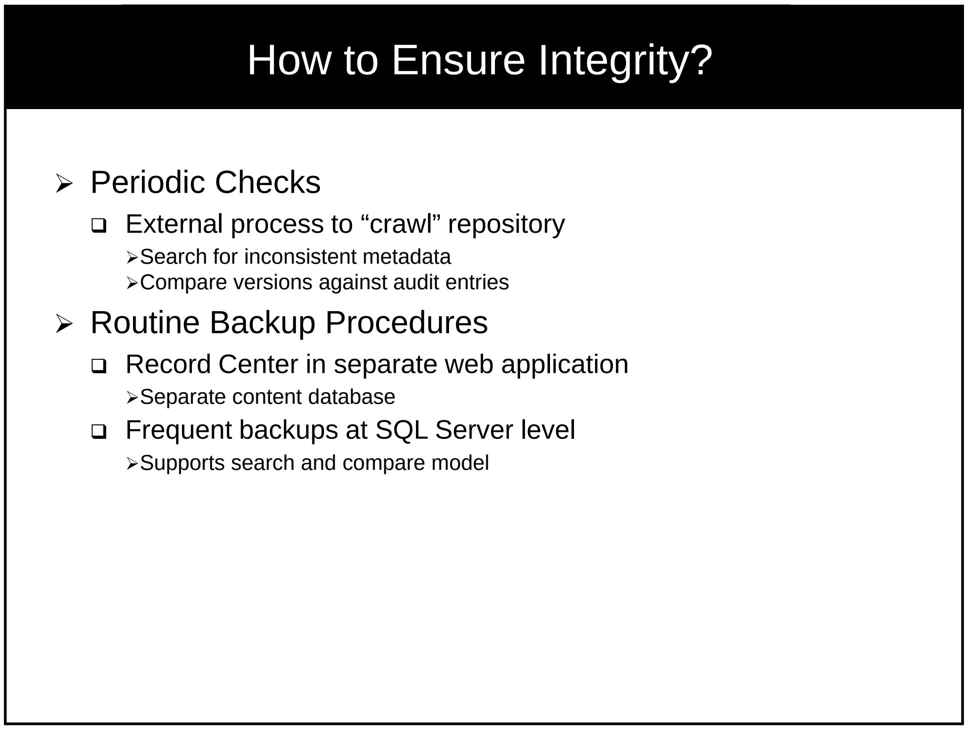
Record Integrity is a Core Requirement

- Companies must "prove" that records have not been tampered with
  - Applies to content as well as to metadata
  - Evidence may be gathered in various ways
    - Examining the audit entries
    - Implementing a custom router
    - Applying an information policy



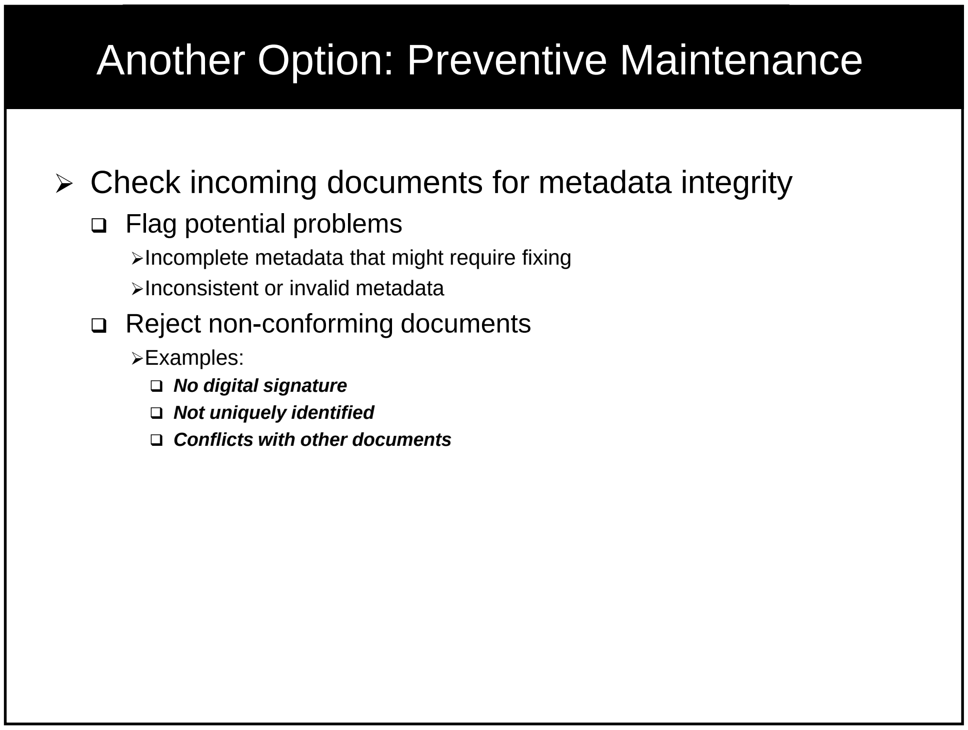
Records are Typically "Locked Down"

- Official Record content not allowed to change
- Changes to metadata carefully monitored
  - May incur additional overhead
    - Administrative/Legal costs
- Records may be reorganized
  - Reclassified as different routing types
  - Moved to different libraries
    - Carefully tracked
- Records may be retired or versioned
  - Using information management policies



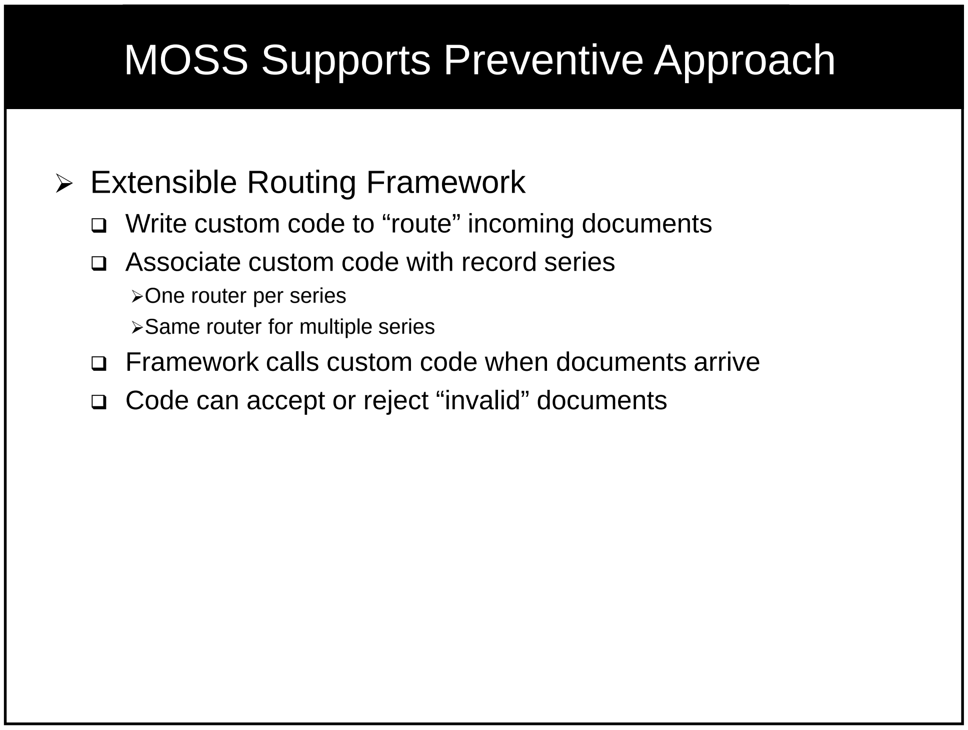
How to Ensure Integrity?

- Periodic Checks
  - External process to “crawl” repository
    - Search for inconsistent metadata
    - Compare versions against audit entries
- Routine Backup Procedures
  - Record Center in separate web application
    - Separate content database
  - Frequent backups at SQL Server level
    - Supports search and compare model



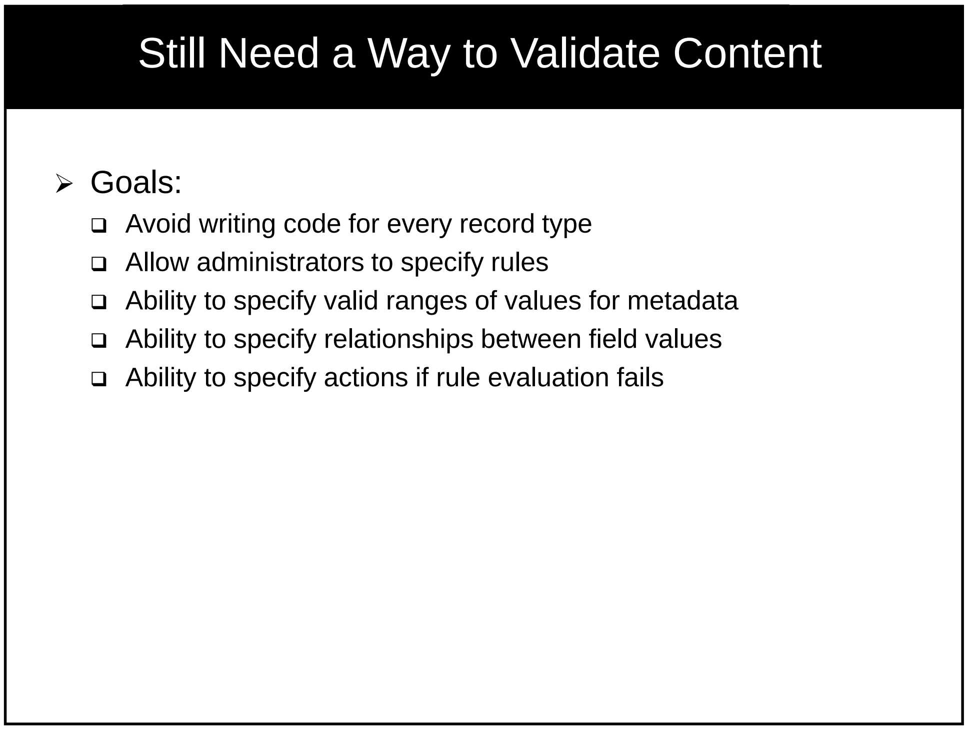
Another Option: Preventive Maintenance

- Check incoming documents for metadata integrity
  - Flag potential problems
    - Incomplete metadata that might require fixing
    - Inconsistent or invalid metadata
  - Reject non-conforming documents
    - Examples:
      - ***No digital signature***
      - ***Not uniquely identified***
      - ***Conflicts with other documents***



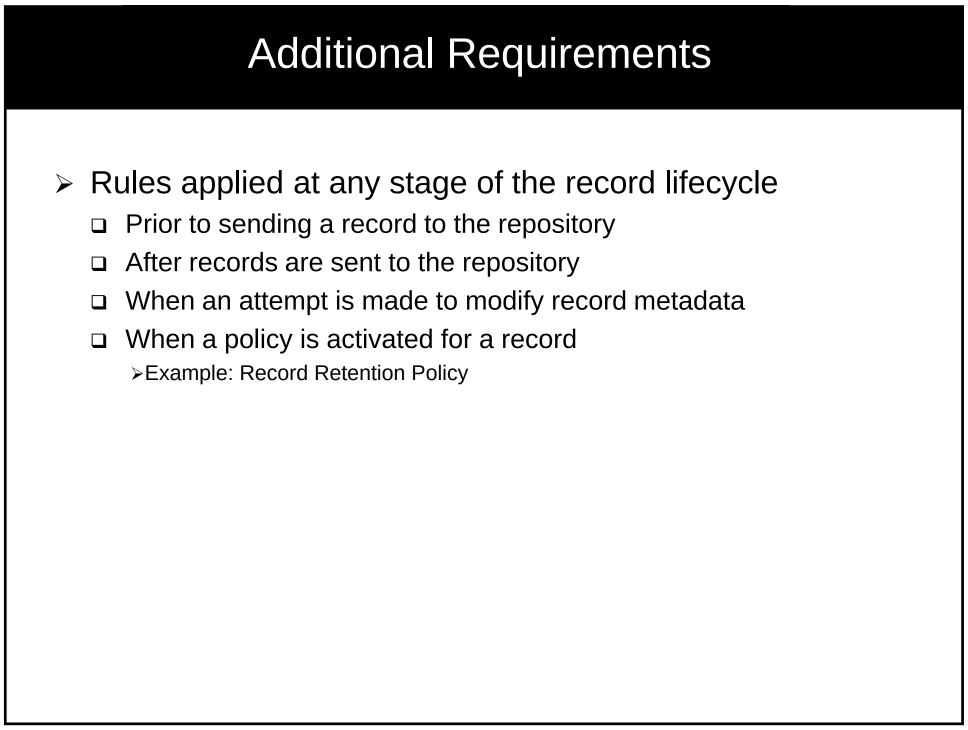
MOSS Supports Preventive Approach

- Extensible Routing Framework
  - Write custom code to “route” incoming documents
  - Associate custom code with record series
    - One router per series
    - Same router for multiple series
  - Framework calls custom code when documents arrive
  - Code can accept or reject “invalid” documents



Still Need a Way to Validate Content

- Goals:
  - Avoid writing code for every record type
  - Allow administrators to specify rules
  - Ability to specify valid ranges of values for metadata
  - Ability to specify relationships between field values
  - Ability to specify actions if rule evaluation fails



Additional Requirements

- Rules applied at any stage of the record lifecycle
  - Prior to sending a record to the repository
  - After records are sent to the repository
  - When an attempt is made to modify record metadata
  - When a policy is activated for a record
    - Example: Record Retention Policy

Key Observations

1. Records are based on content types
   - ❑ Record Series name mapped to Content Type name
2. Content Types support embedded XML
   - ❑ Rules can be attached to content type template
3. Rules can be schematized
   - ❑ Schema encoded using wrapper classes
4. Content Types have behavioral characteristics
   - ❑ Tightly coupled via event receivers
   - ❑ Loosely coupled via information policy features

Additional Observations

- ➢ Content type templates are attached to records
  - ❑ But only if sent to repository by WSS
  - ❑ Ensures that rules are consistently applied
  - ❑ Problem if sent via web service from non-WSS source
- ➢ Validation code can be called from custom router
  - ❑ Records series "name" identifies the content type
  - ❑ Router extracts attached content type template
  - ❑ Router retrieves embedded rule specification
  - ❑ Router evaluates rules against supplied document properties

Alternative Approach

- Rules added to document metadata directly
  - Default set of rules attached to content type
  - Policy feature adds hidden column to store rules
  - Policy feature copies default rules from content type to document instance when policy is applied
  - Code checks column to retrieve rules
  - Code evaluates rules against document properties

Enforcing Data Integrity using Custom Schemas

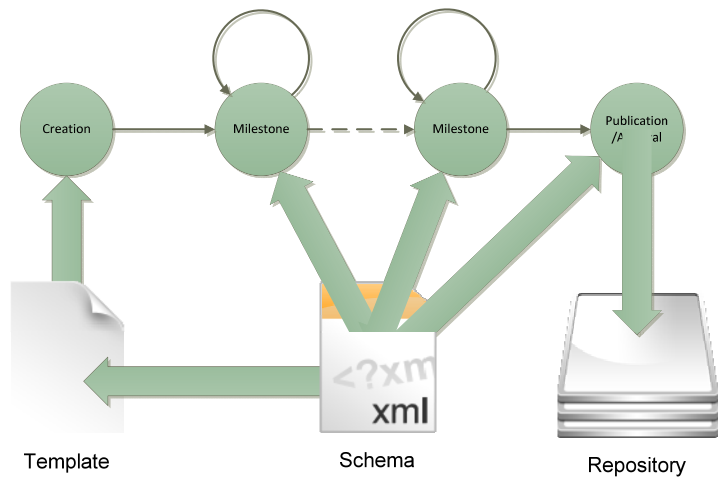
Managing Metadata via Rules

- Rules can be used to
  - Define valid ranges of metadata field values
  - Establish relationships among data fields
- Rules can be applied at any stage
  - To determine whether new content may be created
  - To determine whether existing content is “valid”

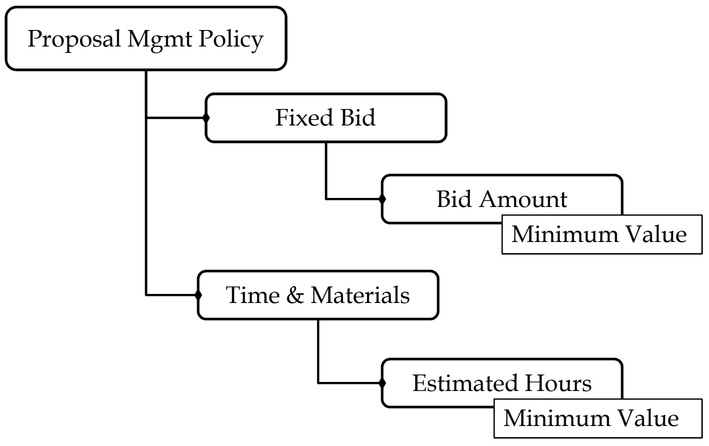
Content Validity

- Meaning depends on context
  - Validity may change as content moves through its lifecycle
  - Content may be valid for one purpose but invalid for another
- Implementation requires flexibility
  - Ideally, need to attach an “interface” instead of a “type”
  - SharePoint Content Types alone are insufficient
    - Need custom extension mechanism

Leveraging the Power of XML Schemas



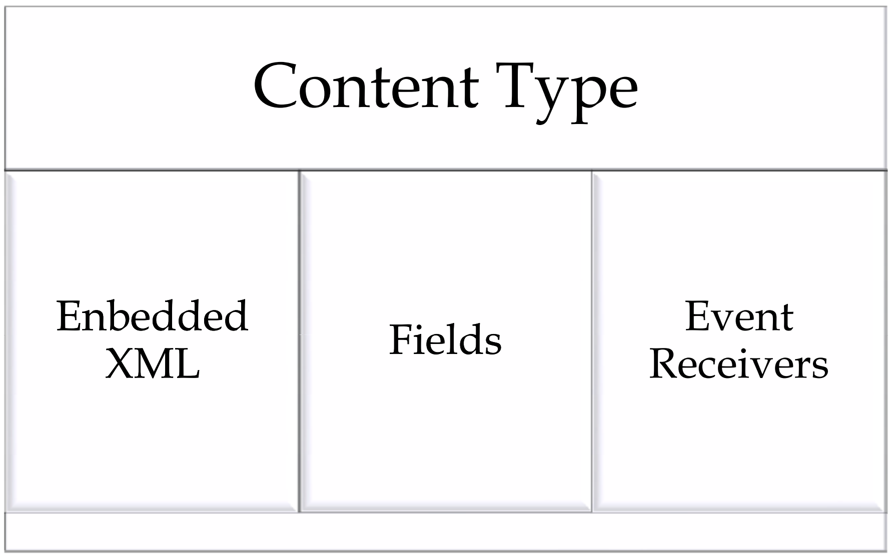
A Proposal Management Schema

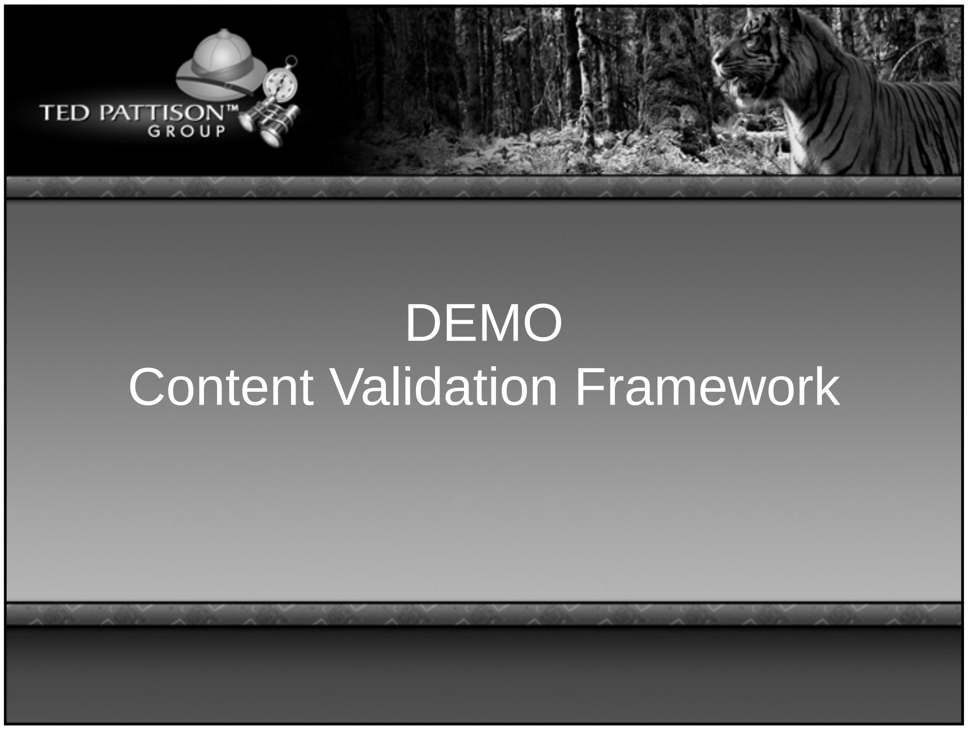


Extensible Validation via Rules

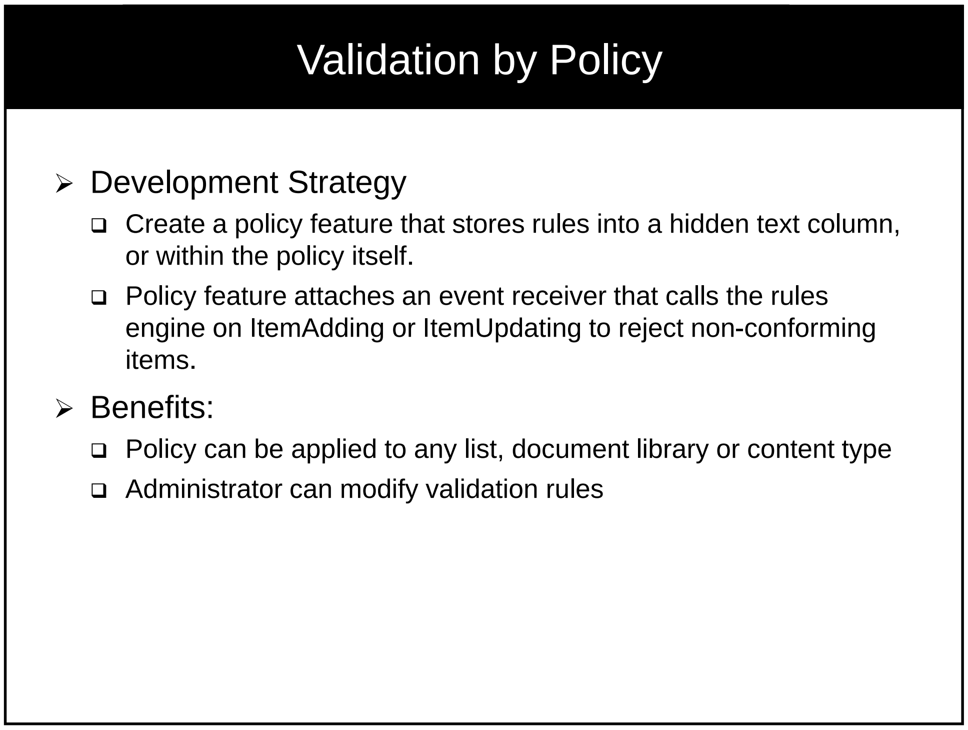
```
                    <Rule Name="MinimumBidAmount">
                          <Match>
Match                        <Eq>
Expression                      <FieldName>ProposalType</FieldName>
                                <Value Type="Text">FixedBid</Value>
                             </Eq>
                          </Match>
                          <Try>
                             <Geq>
Rule                            <FieldName>BidAmount</FieldName>
Expression                      <Value Type="double">5000</Value>
                             </Geq>
                          </Try>
Validation                <Catch>
Exception                    <Throw>The bid amount must be at least $5000</Throw>
                          </Catch>
                       </Rule>
```

Where to Store the Rules?





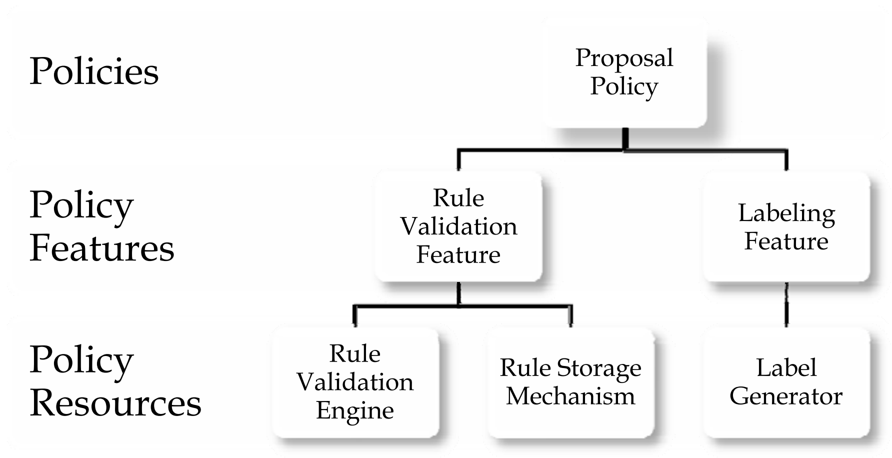
DEMO
Content Validation Framework



Validation by Policy

- Development Strategy
  - Create a policy feature that stores rules into a hidden text column, or within the policy itself.
  - Policy feature attaches an event receiver that calls the rules engine on ItemAdding or ItemUpdating to reject non-conforming items.
- Benefits:
  - Policy can be applied to any list, document library or content type
  - Administrator can modify validation rules

Information Policy

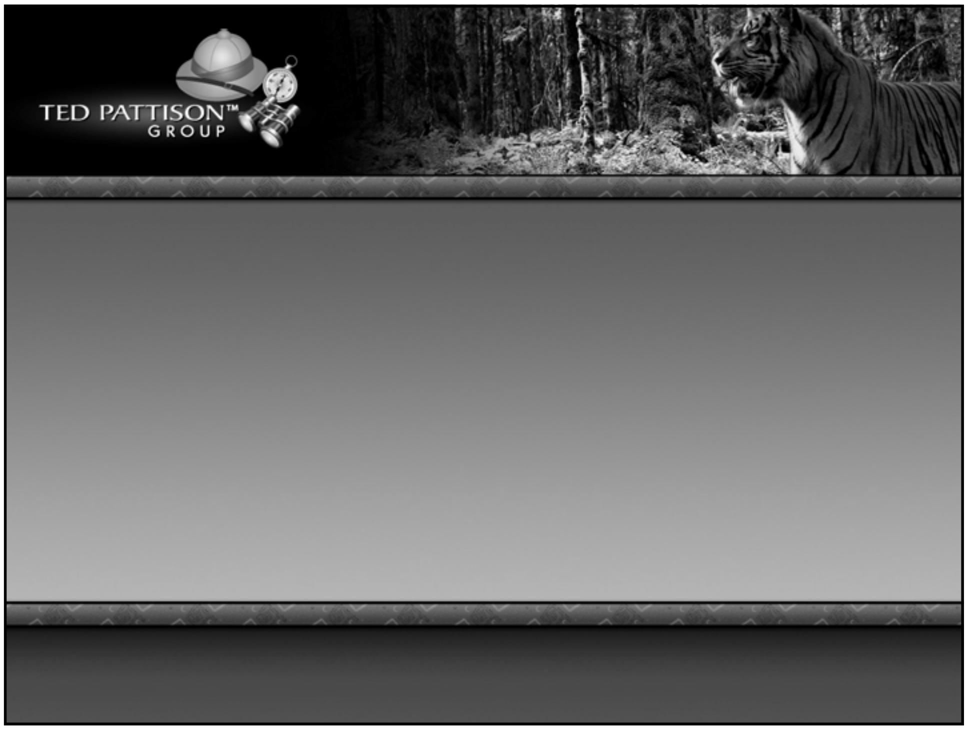


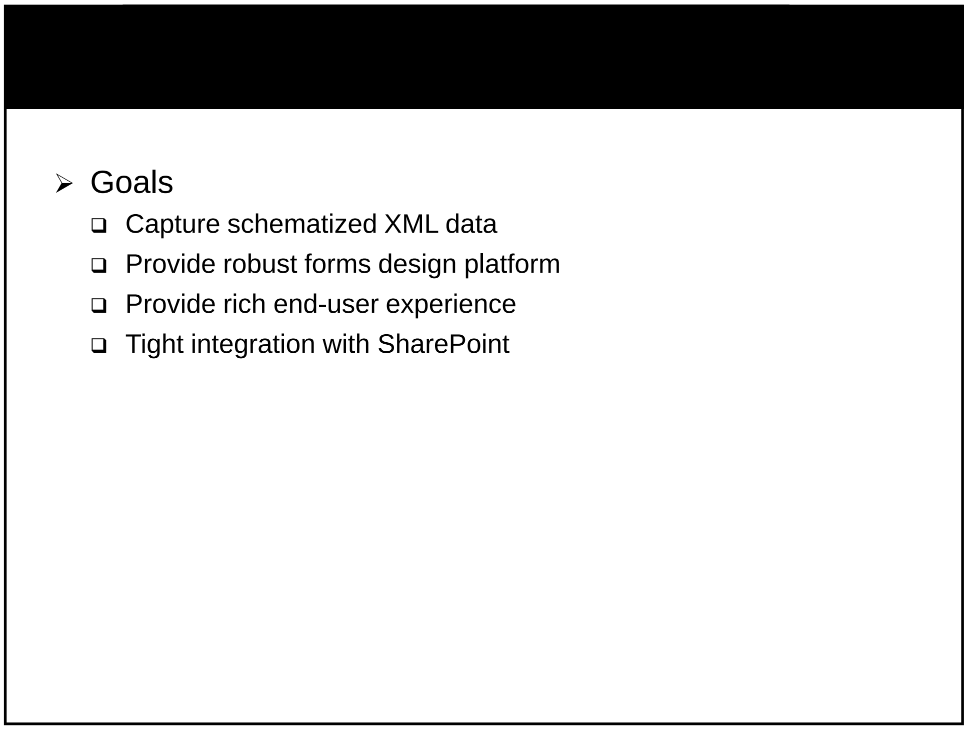
Validating Router

- Extensions to validation framework
  - Add support for RecordRepositoryProperties collection
    - Simplify implementation of IRouter.OnSubmitFile
  - Add to abstract base class
- Associate validation rules with destination library
  - Fits with the record repository architecture
  - Can still apply to content types (record series)

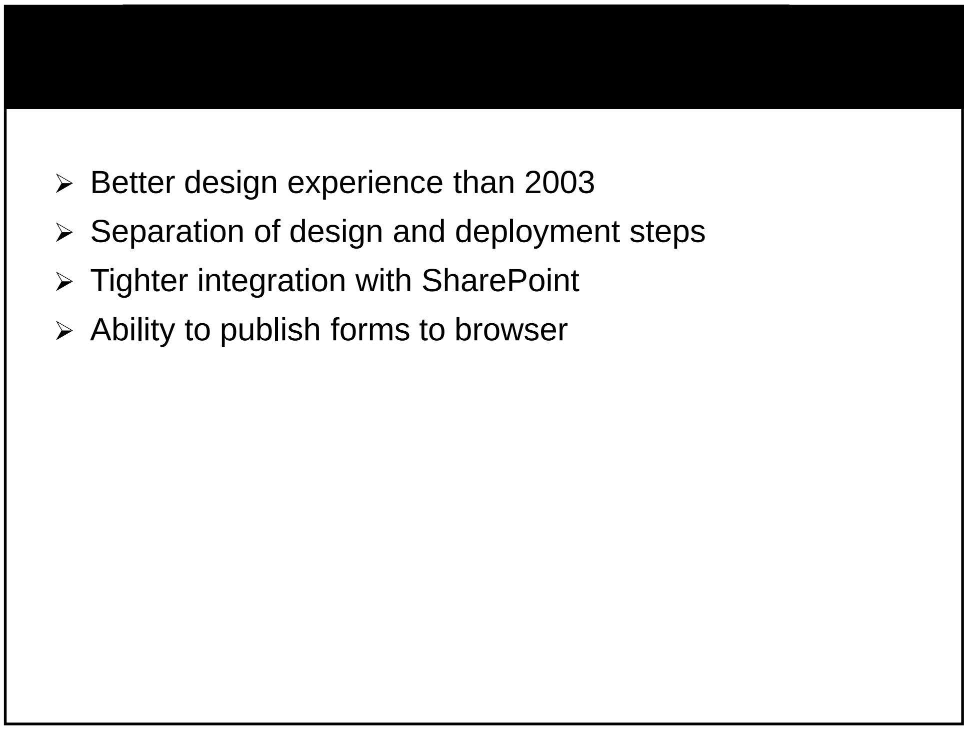
Summary

- Record integrity is a core requirement for records management, because records managers must not only ensure the integrity of documents but must prove no tampering has occurred.
- Record integrity checks can occur at any stage in the content lifecycle, therefore flexible mechanisms are necessary to avoid downtime associated with implementing an integrity solution strategy.
- Information policy in conjunction with a rules-based approach can provide the needed flexibility by de-coupling the validation engine from the document content and the storage mechanism.

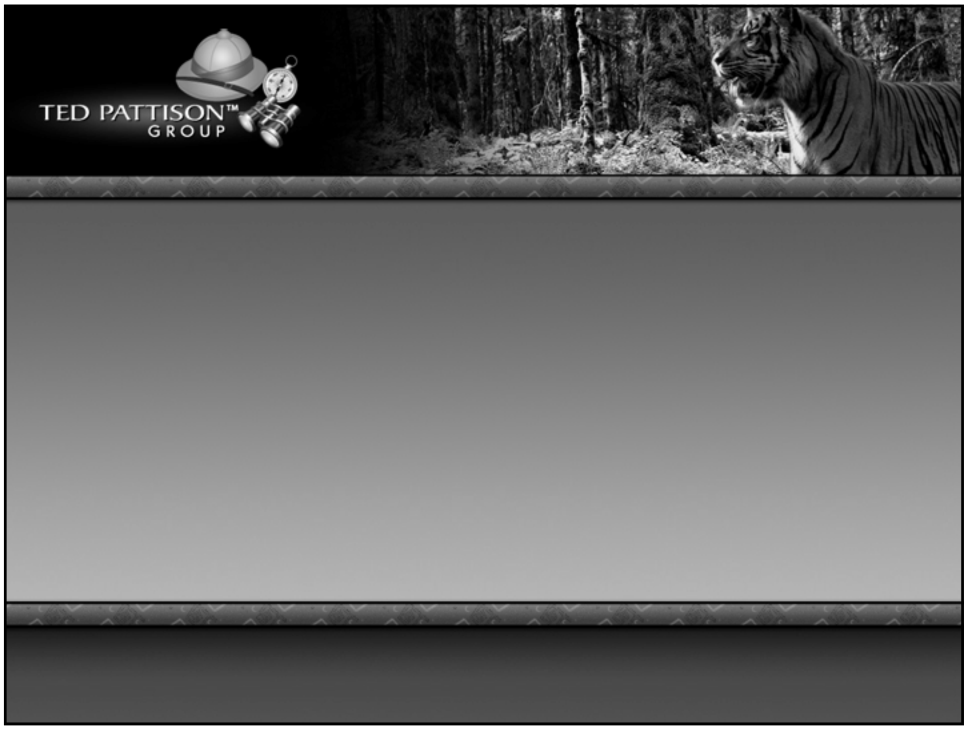




- Goals
  - Capture schematized XML data
  - Provide robust forms design platform
  - Provide rich end-user experience
  - Tight integration with SharePoint



- Better design experience than 2003
- Separation of design and deployment steps
- Tighter integration with SharePoint
- Ability to publish forms to browser

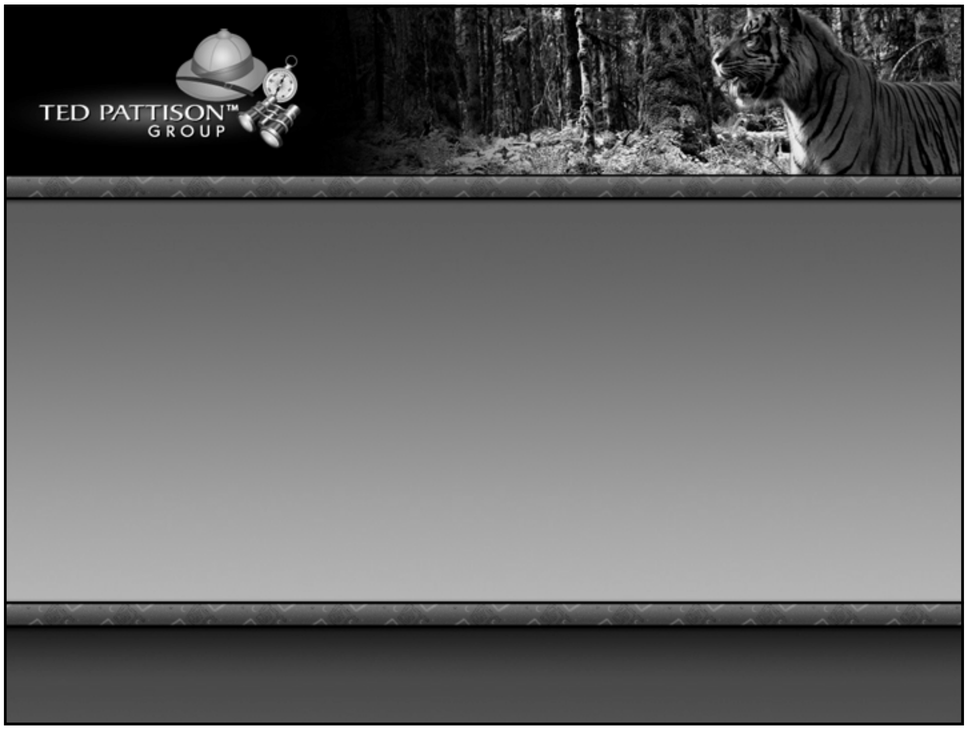


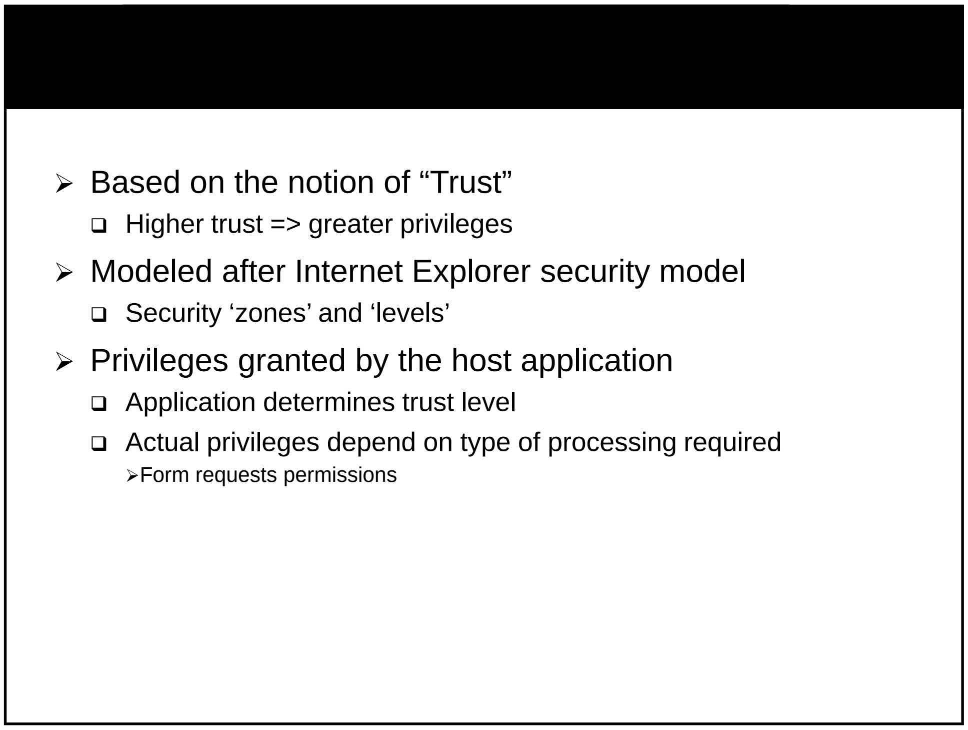
- Stored as a CAB file (.xsn)
  - Embedded form manifest (xsf) describes the form
  - XML schema (xsd) describes the form data
  - XML style sheets (xsl) describe the views
  - XML data files contain form data

- Form Libraries
  - Specialization of document library
    - Document template is a .xsn file
  - Content Types
    - Created by 'publishing' the form
      - ***Document template is a .xsn file***

- Visual Studio Tools for Applications (VSTA)
  - A way to write managed code for any kind of application (managed or unmanaged)
  - Provides a Visual Studio IDE embedded within the InfoPath forms designer
- Visual Studio Tools for Office (VSTO)
  - A subset of VSTA geared toward Office applications
  - Embeds the InfoPath forms designer within the Visual Studio IDE

TED PATTISON™ GROUP

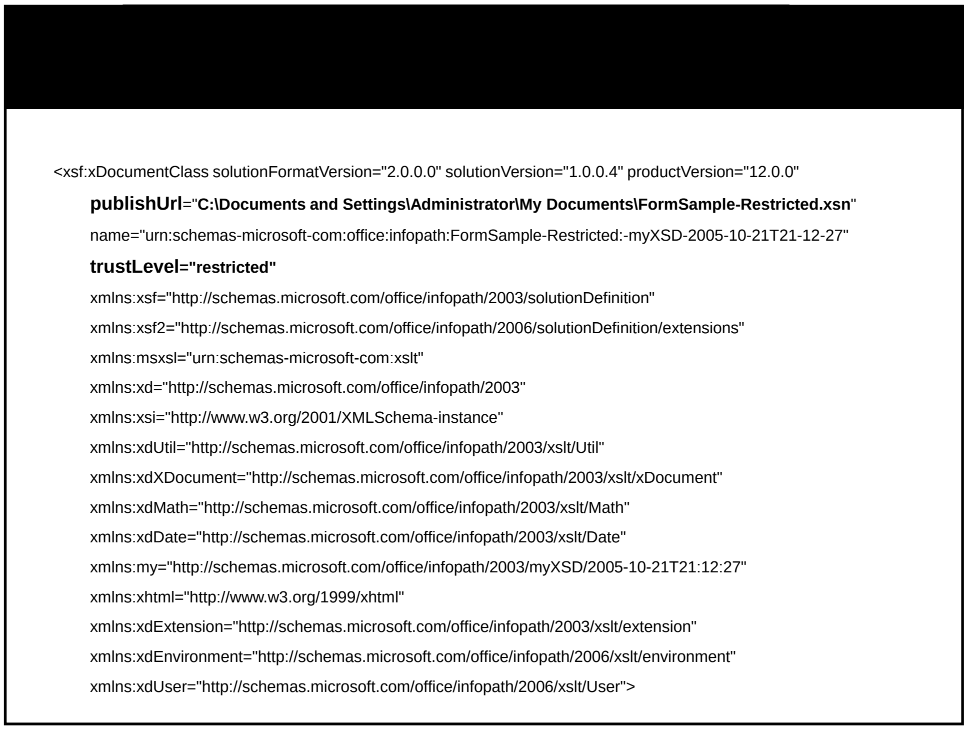




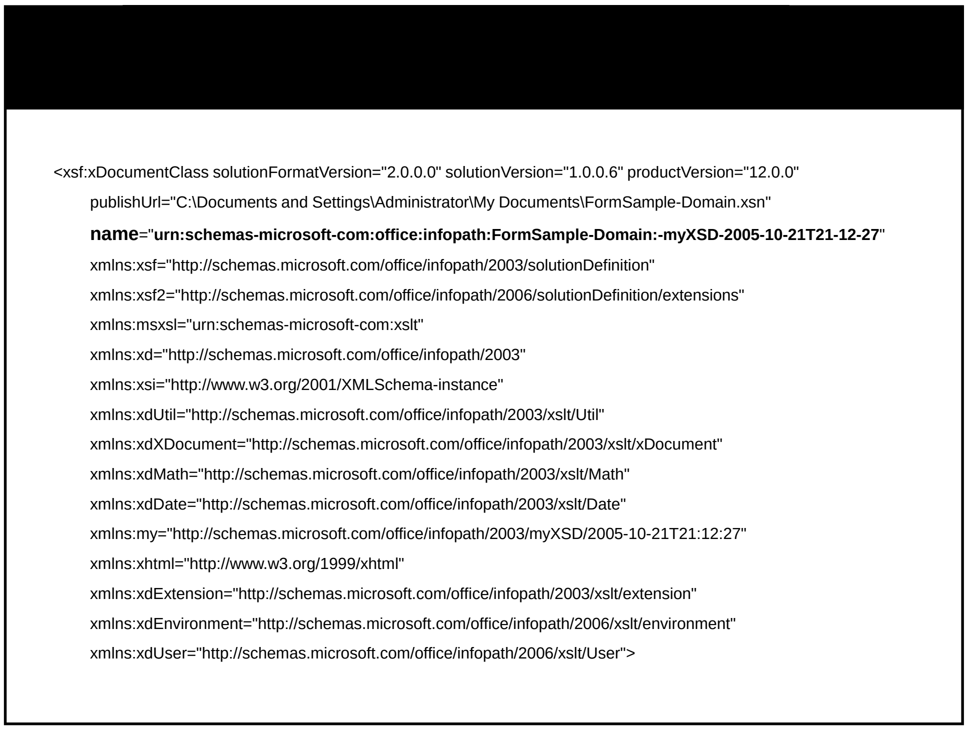
- Based on the notion of “Trust”
  - Higher trust => greater privileges
- Modeled after Internet Explorer security model
  - Security ‘zones’ and ‘levels’
- Privileges granted by the host application
  - Application determines trust level
  - Actual privileges depend on type of processing required
    - Form requests permissions

- URL based (sandboxed)
  - User opens form from the URL where form is published
  - URL is also embedded within the form
  - Similar to IE security for a web page
- URN based
  - Uniform Resource Name (URN) embedded within the form
  - Domain-level security by default
  - Form can request full trust using special tag within the form
  - Must be registered on client or must be digitally signed

- Restricted
  - URL based (sandboxed)
  - Inherits permissions from the IE zone associated with the URL
- Domain
  - URN based (no requireFullTrust attribute)
  - Inherits permissions from the Local Computer zone
- Full
  - URN based (includes requireFullTrust attribute)
  - Must be digitally signed or installed in register
  - User can control whether installed forms are trusted



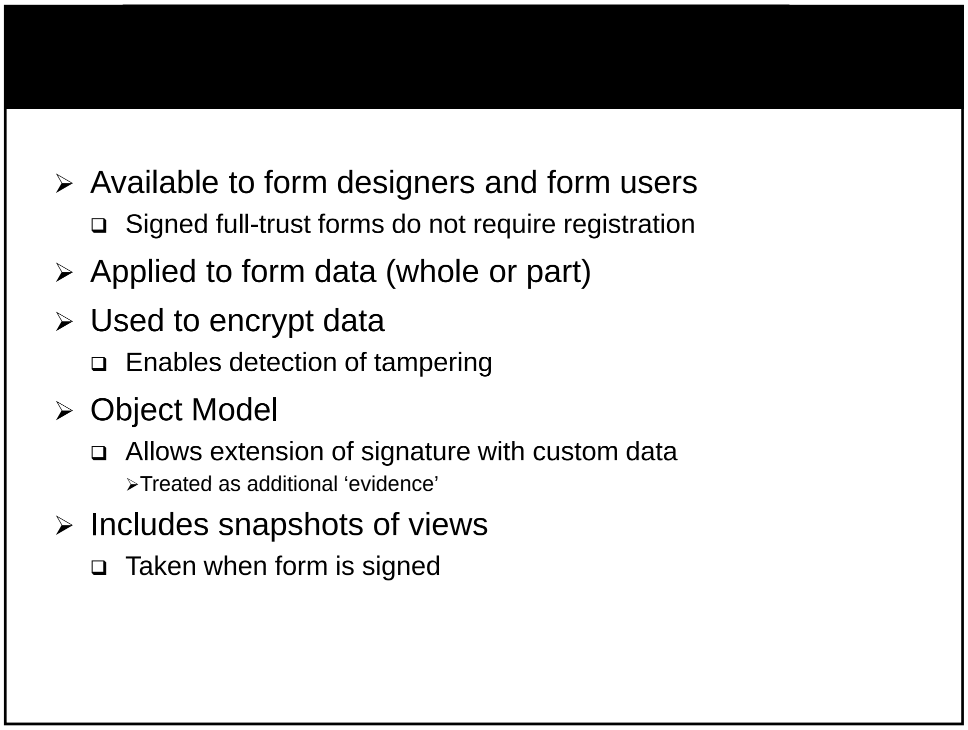
```
<xsf:xDocumentClass solutionFormatVersion="2.0.0.0" solutionVersion="1.0.0.4" productVersion="12.0.0"
    publishUrl="C:\Documents and Settings\Administrator\My Documents\FormSample-Restricted.xsn"
    name="urn:schemas-microsoft-com:office:infopath:FormSample-Restricted:-myXSD-2005-10-21T21-12-27"
    trustLevel="restricted"
    xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
    xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt"
    xmlns:xd="http://schemas.microsoft.com/office/infopath/2003"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xdUtil="http://schemas.microsoft.com/office/infopath/2003/xslt/Util"
    xmlns:xdXDocument="http://schemas.microsoft.com/office/infopath/2003/xslt/xDocument"
    xmlns:xdMath="http://schemas.microsoft.com/office/infopath/2003/xslt/Math"
    xmlns:xdDate="http://schemas.microsoft.com/office/infopath/2003/xslt/Date"
    xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
    xmlns:xdExtension="http://schemas.microsoft.com/office/infopath/2003/xslt/extension"
    xmlns:xdEnvironment="http://schemas.microsoft.com/office/infopath/2006/xslt/environment"
    xmlns:xdUser="http://schemas.microsoft.com/office/infopath/2006/xslt/User">
```



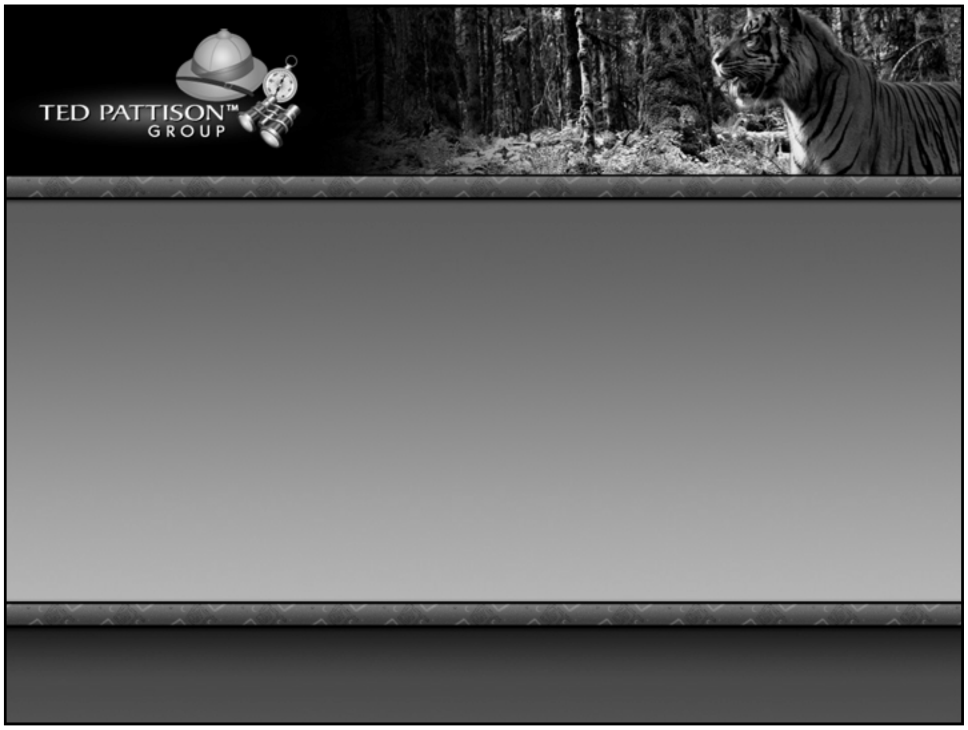
```
<xsf:xDocumentClass solutionFormatVersion="2.0.0.0" solutionVersion="1.0.0.6" productVersion="12.0.0"
    publishUrl="C:\Documents and Settings\Administrator\My Documents\FormSample-Domain.xsn"
    name="urn:schemas-microsoft-com:office:infopath:FormSample-Domain:-myXSD-2005-10-21T21-12-27"
    xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
    xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt"
    xmlns:xd="http://schemas.microsoft.com/office/infopath/2003"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xdUtil="http://schemas.microsoft.com/office/infopath/2003/xslt/Util"
    xmlns:xdXDocument="http://schemas.microsoft.com/office/infopath/2003/xslt/xDocument"
    xmlns:xdMath="http://schemas.microsoft.com/office/infopath/2003/xslt/Math"
    xmlns:xdDate="http://schemas.microsoft.com/office/infopath/2003/xslt/Date"
    xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
    xmlns:xdExtension="http://schemas.microsoft.com/office/infopath/2003/xslt/extension"
    xmlns:xdEnvironment="http://schemas.microsoft.com/office/infopath/2006/xslt/environment"
    xmlns:xdUser="http://schemas.microsoft.com/office/infopath/2006/xslt/User">
```

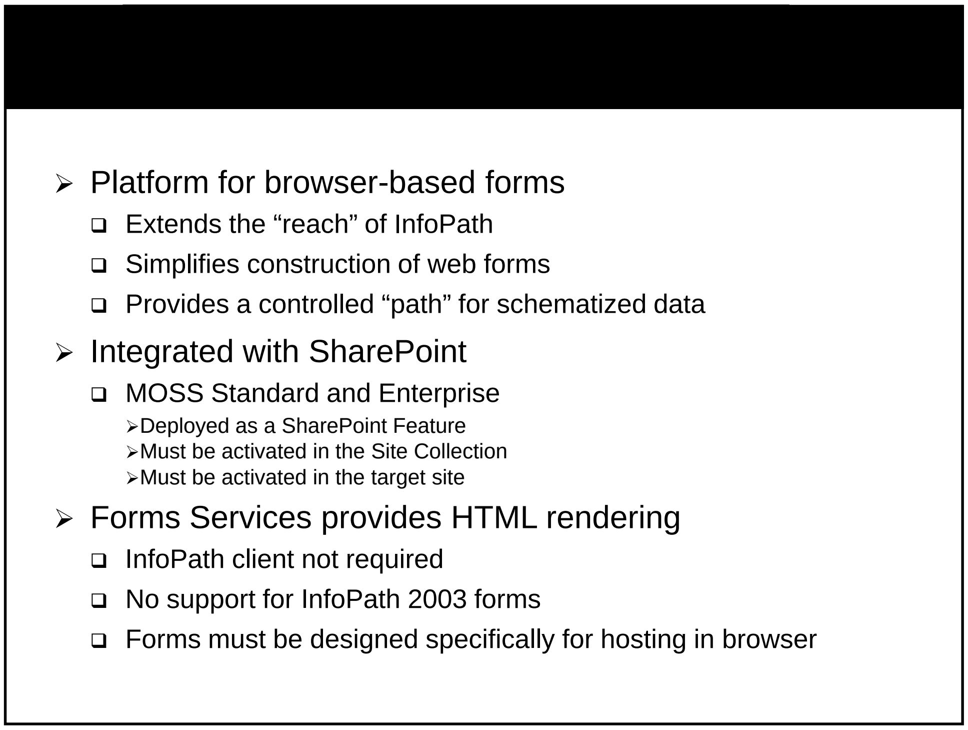
```
<xsf:xDocumentClass solutionFormatVersion="2.0.0.0" solutionVersion="1.0.0.8" productVersion="12.0.0"
    publishUrl="C:\Documents and Settings\Administrator\My Documents\FormSample-Full.xsn"
    name="urn:schemas-microsoft-com:office:infopath:FormSample-Full:-myXSD-2005-10-21T21-12-27"
    requireFullTrust="yes"
    xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutionDefinition"
    xmlns:xsf2="http://schemas.microsoft.com/office/infopath/2006/solutionDefinition/extensions"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt"
    xmlns:xd="http://schemas.microsoft.com/office/infopath/2003"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xdUtil="http://schemas.microsoft.com/office/infopath/2003/xslt/Util"
    xmlns:xdXDocument="http://schemas.microsoft.com/office/infopath/2003/xslt/xDocument"
    xmlns:xdMath="http://schemas.microsoft.com/office/infopath/2003/xslt/Math"
    xmlns:xdDate="http://schemas.microsoft.com/office/infopath/2003/xslt/Date"
    mlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2005-10-21T21:12:27"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
    xmlns:xdExtension="http://schemas.microsoft.com/office/infopath/2003/xslt/extension"
    xmlns:xdEnvironment="http://schemas.microsoft.com/office/infopath/2006/xslt/environment"
    xmlns:xdUser="http://schemas.microsoft.com/office/infopath/2006/xslt/User">
```

- Key areas of risk identified:
  - Accessing data across domains
    - Using form user credentials to access data
  - Using the InfoPath task pane
    - May include web pages
      - ***Must be from same domain as the form, or***
      - ***Cross domain access must be enabled for zone***
  - Embedding ActiveX controls
    - Disabled in form views
  - Calling the InfoPath object model
    - Controlled by Code Access Security (CAS)

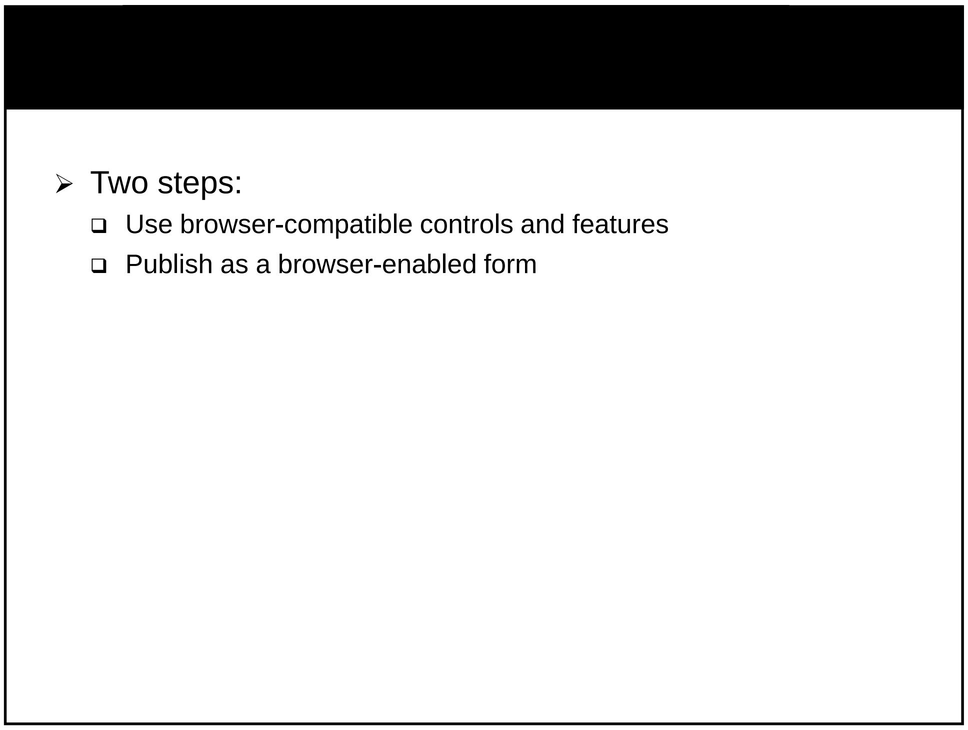


- Available to form designers and form users
  - Signed full-trust forms do not require registration
- Applied to form data (whole or part)
- Used to encrypt data
  - Enables detection of tampering
- Object Model
  - Allows extension of signature with custom data
    - Treated as additional ‘evidence’
- Includes snapshots of views
  - Taken when form is signed

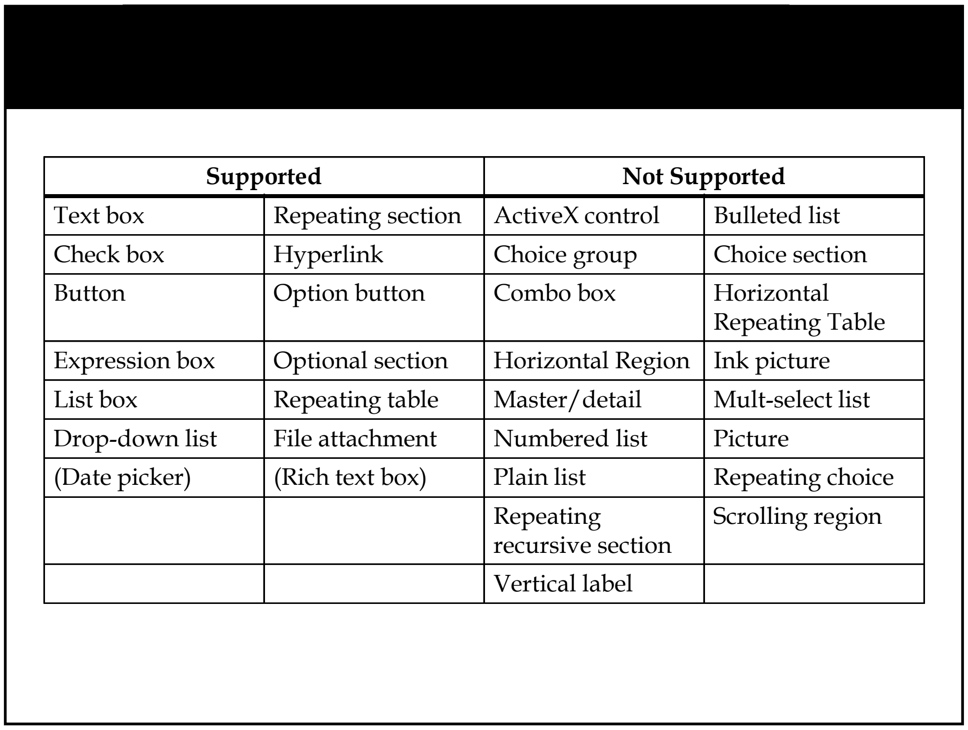




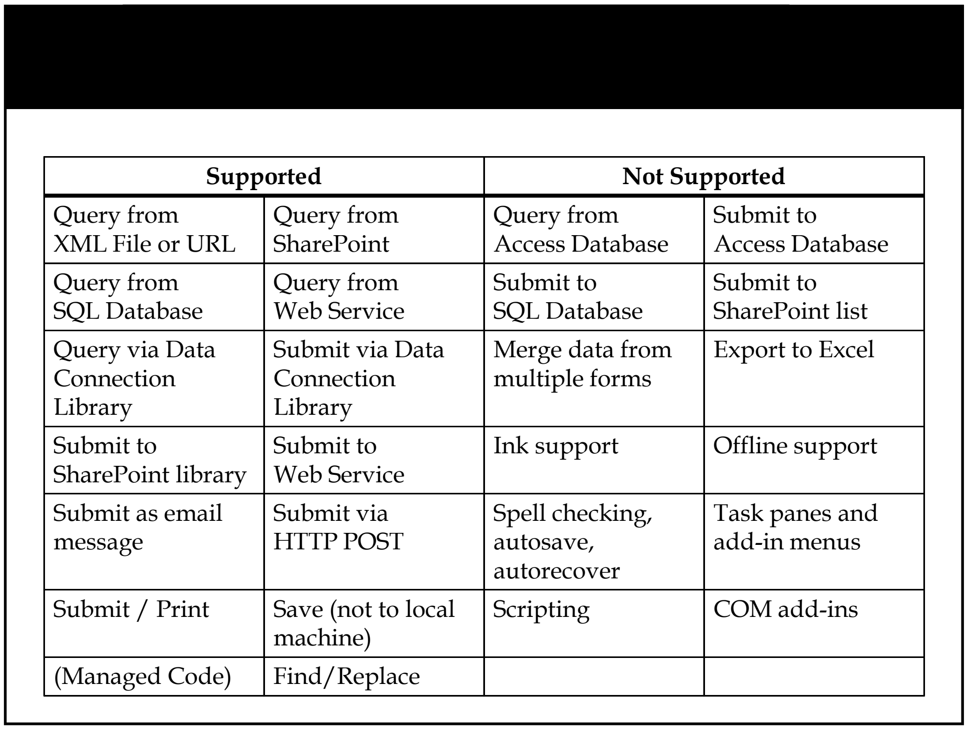
- Platform for browser-based forms
  - Extends the “reach” of InfoPath
  - Simplifies construction of web forms
  - Provides a controlled “path” for schematized data
- Integrated with SharePoint
  - MOSS Standard and Enterprise
    - Deployed as a SharePoint Feature
    - Must be activated in the Site Collection
    - Must be activated in the target site
- Forms Services provides HTML rendering
  - InfoPath client not required
  - No support for InfoPath 2003 forms
  - Forms must be designed specifically for hosting in browser



- Two steps:
  - Use browser-compatible controls and features
  - Publish as a browser-enabled form

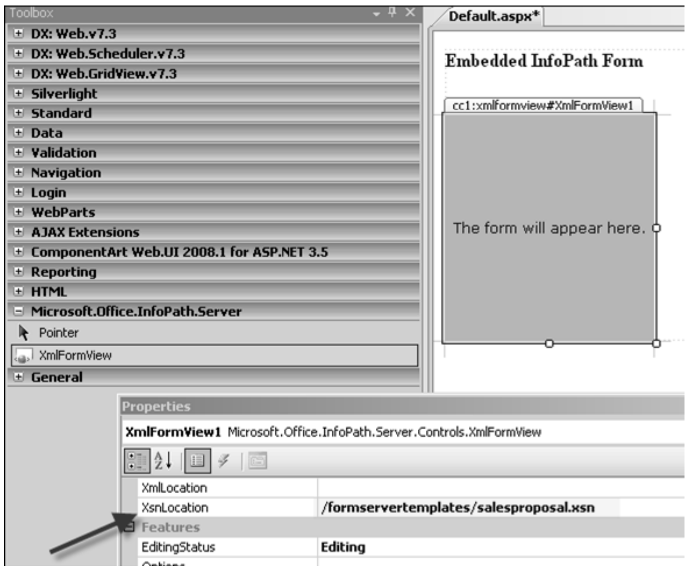


Supported		Not Supported	
Text box	Repeating section	ActiveX control	Bulleted list
Check box	Hyperlink	Choice group	Choice section
Button	Option button	Combo box	Horizontal Repeating Table
Expression box	Optional section	Horizontal Region	Ink picture
List box	Repeating table	Master/detail	Mult-select list
Drop-down list	File attachment	Numbered list	Picture
(Date picker)	(Rich text box)	Plain list	Repeating choice
		Repeating recursive section	Scrolling region
		Vertical label	



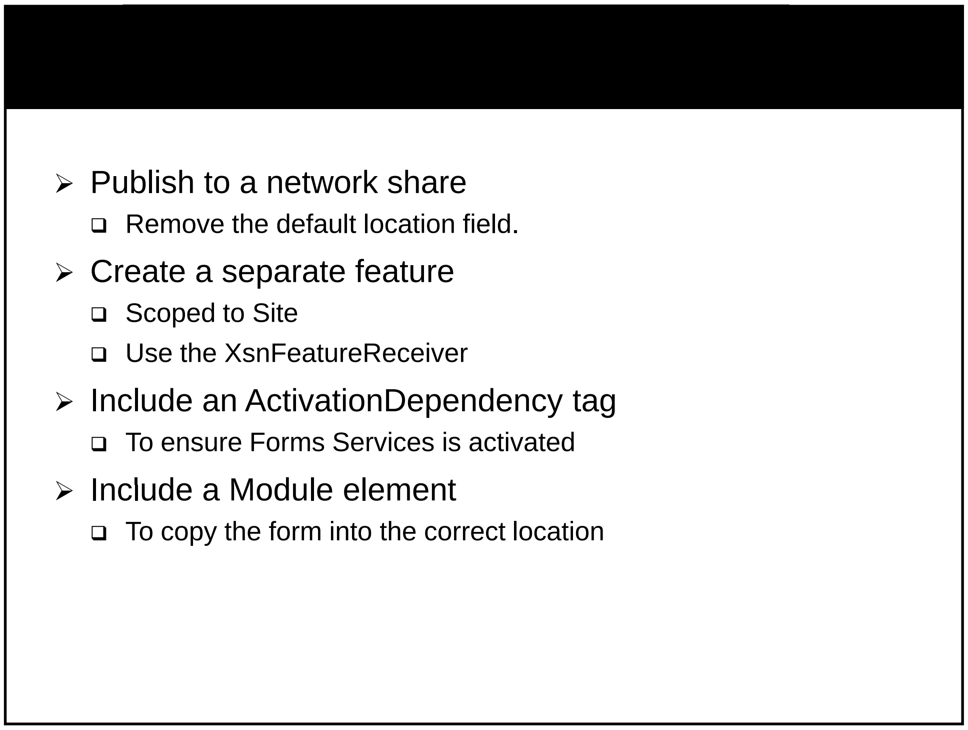
Supported		Not Supported	
Query from XML File or URL	Query from SharePoint	Query from Access Database	Submit to Access Database
Query from SQL Database	Query from Web Service	Submit to SQL Database	Submit to SharePoint list
Query via Data Connection Library	Submit via Data Connection Library	Merge data from multiple forms	Export to Excel
Submit to SharePoint library	Submit to Web Service	Ink support	Offline support
Submit as email message	Submit via HTTP POST	Spell checking, autosave, autorecover	Task panes and add-in menus
Submit / Print	Save (not to local machine)	Scripting	COM add-ins
(Managed Code)	Find/Replace		

- Based on DHTML and Javascript
- Tested with IE, FireFox, Netscape & Mobile Devices
- Implemented as a server control
  - XMLFormView Control

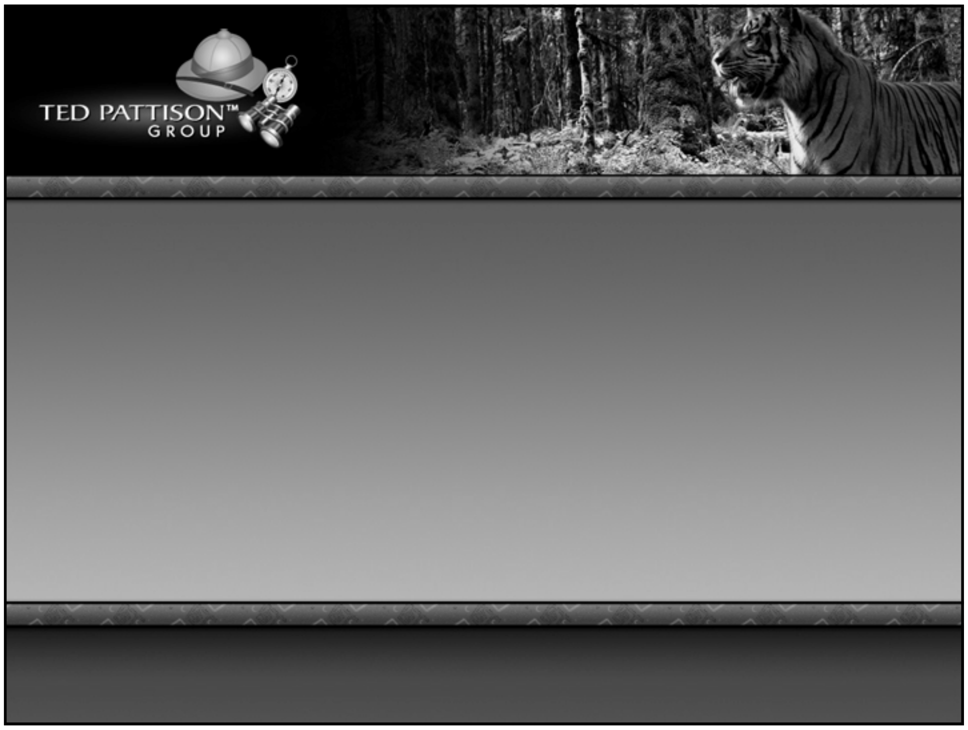


XmlFormView1 Microsoft.Office.InfoPath.Server.Controls.XmlFormView

XmlLocation	
XsnLocation	/formservertemplates/salesproposal.xsn
EditingStatus	Editing



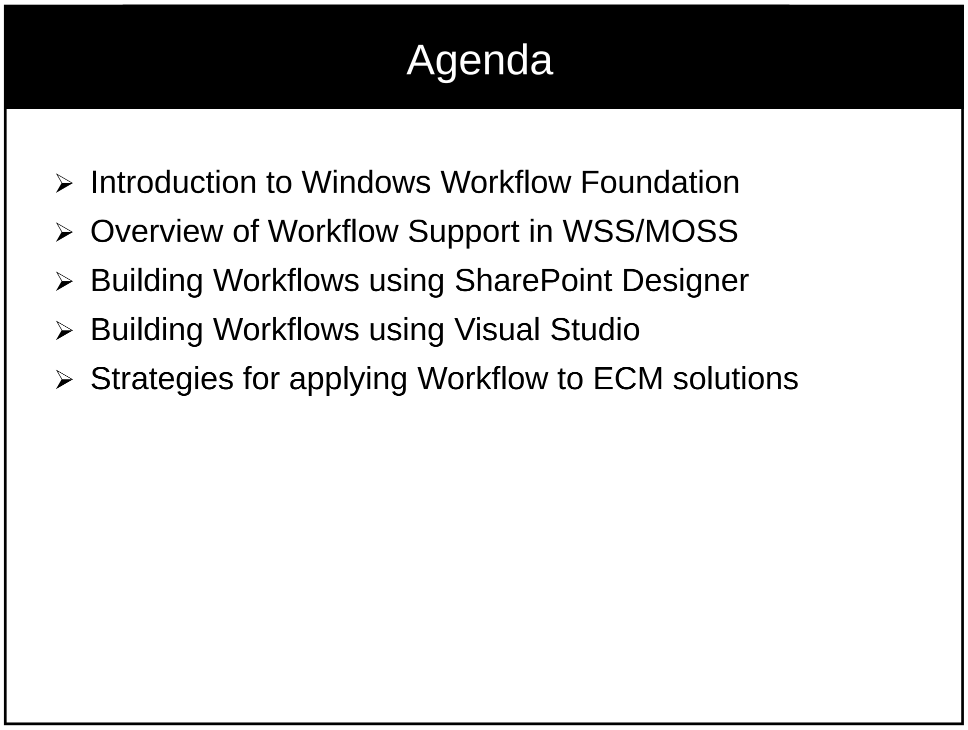
- Publish to a network share
  - Remove the default location field.
- Create a separate feature
  - Scoped to Site
  - Use the XsnFeatureReceiver
- Include an ActivationDependency tag
  - To ensure Forms Services is activated
- Include a Module element
  - To copy the form into the correct location



- SharePoint Central Administration
  - Upload form templates
  - Manage data connections
  - Manage web service proxy

- Manually uploaded by administrator
  - Forms with managed code
  - Forms with data connections
- Installed by features
  - Workflow forms

Using Workflow to Manage Enterprise Content



Agenda

- Introduction to Windows Workflow Foundation
- Overview of Workflow Support in WSS/MOSS
- Building Workflows using SharePoint Designer
- Building Workflows using Visual Studio
- Strategies for applying Workflow to ECM solutions

Introduction

- What is a Workflow?
- A long-running process
  - Has built-in support for persistence
  - Can be suspended or resumed by the WF runtime
- Composed from a series of discrete “activities”
  - Atomic operations for a given domain
  - Activities may contain other activities
- Sequential or state-machine

Workflows and Activities

What is a Workflow Activity?

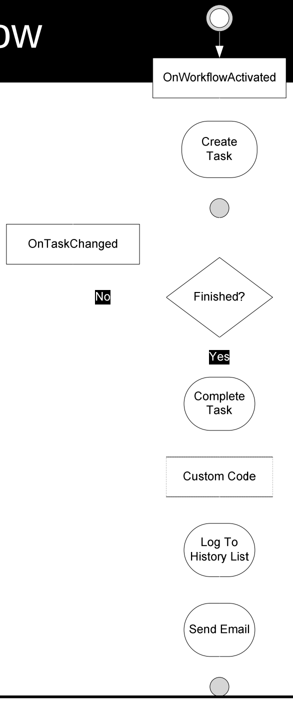
- A .NET class
  - Written in managed code
  - Compiled into a .NET assembly
- Exposes properties and methods
  - Can be bound via reflection to other activities
- Exposes interfaces
  - Can be called by other activities
- Handles events
  - Events generated by the WF runtime

Types of Activities

- Generic Activities
  - Code Activity
  - Composite Activity
  - If-Else, While
- SharePoint-Specific Activities
  - Create Task
  - Update Task
  - OnTaskChanged
  - …

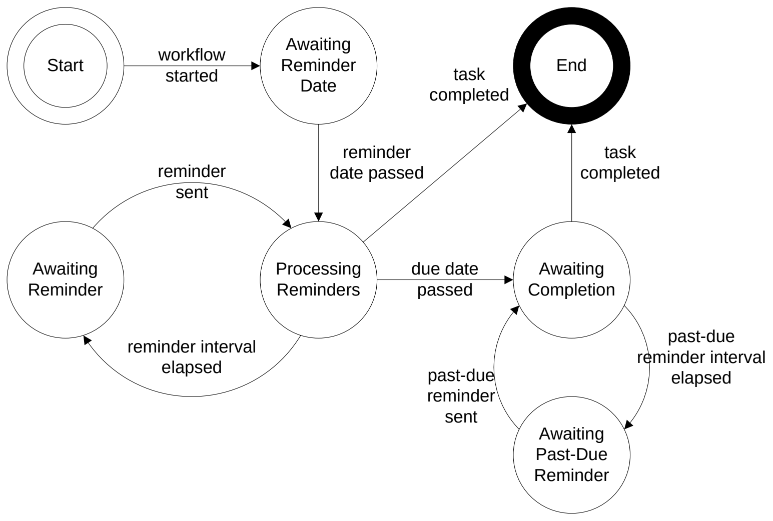
Sequential Workflow

- Similar to a flow-chart
- Harder to model complex workflows



State Machine Workflow

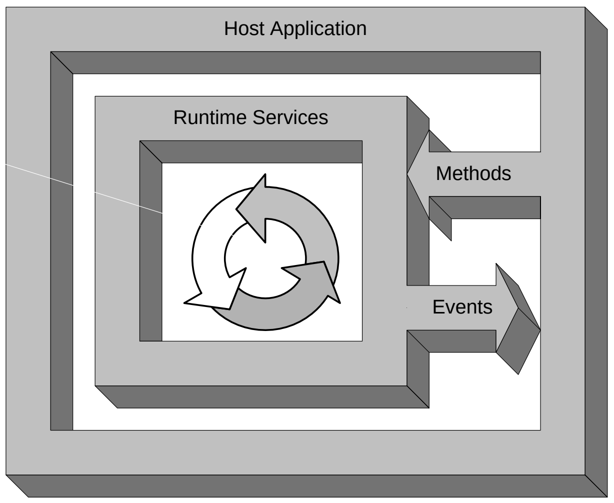
Task Reminder Workflow



Windows Workflow Foundation

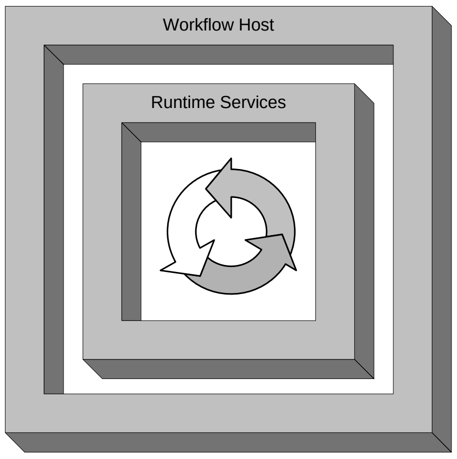
- Hosted Workflow Engine
  - Can be hosted in any .NET application
  - Custom hosting layer in WSS
- Fully Integrated into .NET 3.0
  - Integrated into the OS
  - Required for running workflows

WF Architecture

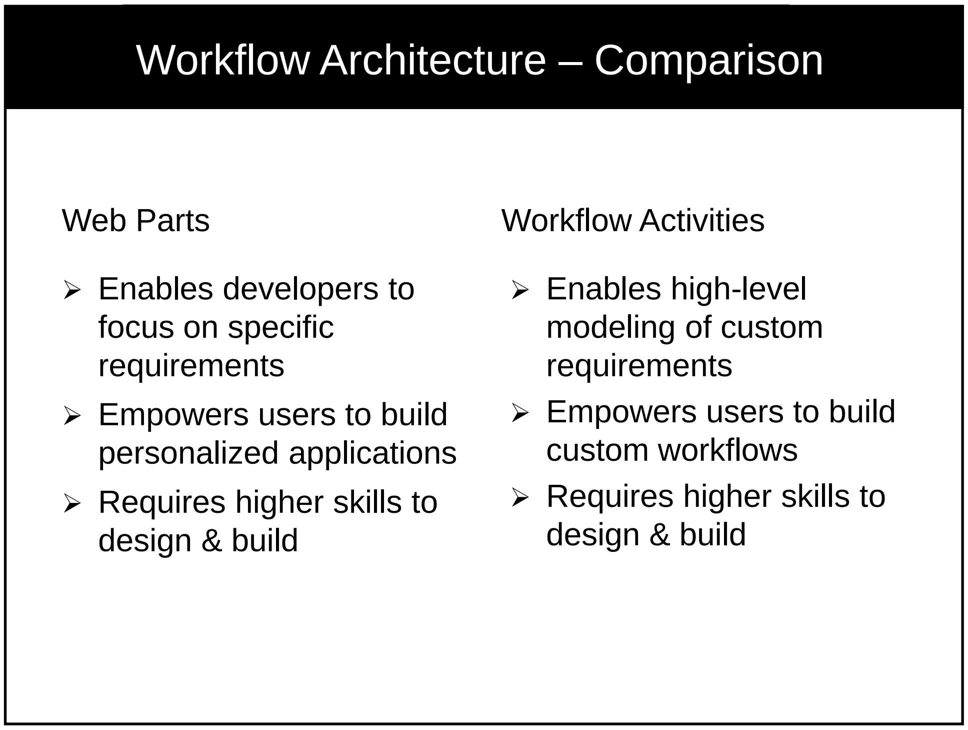


WF Architecture

Base Activities



Custom Activities



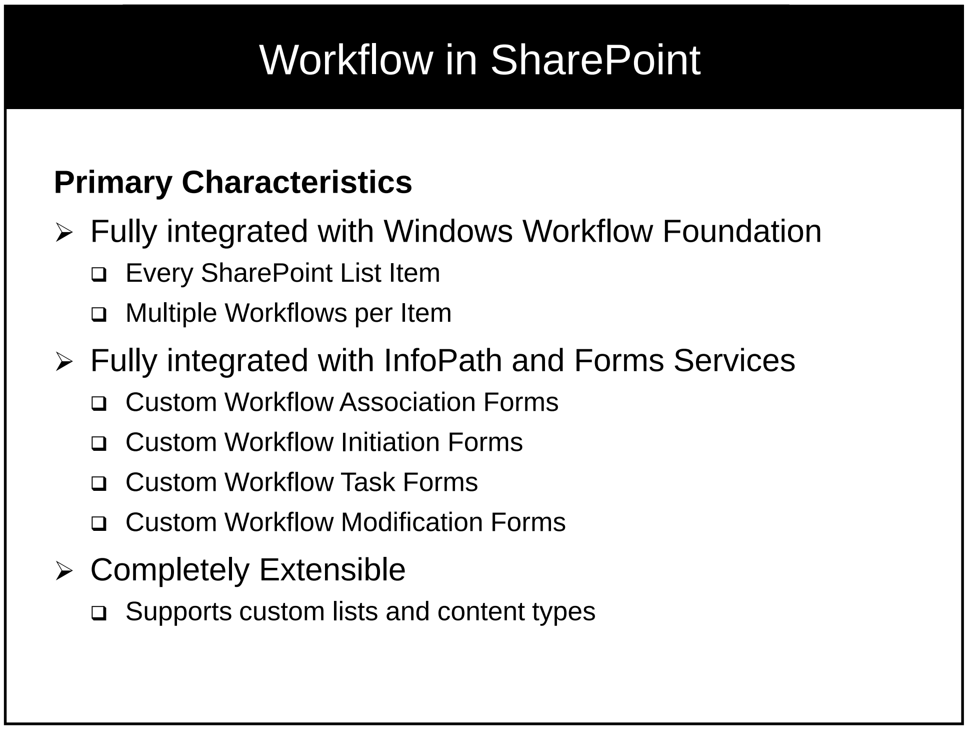
Workflow Architecture – Comparison

Web Parts

- Enables developers to focus on specific requirements
- Empowers users to build personalized applications
- Requires higher skills to design & build

Workflow Activities

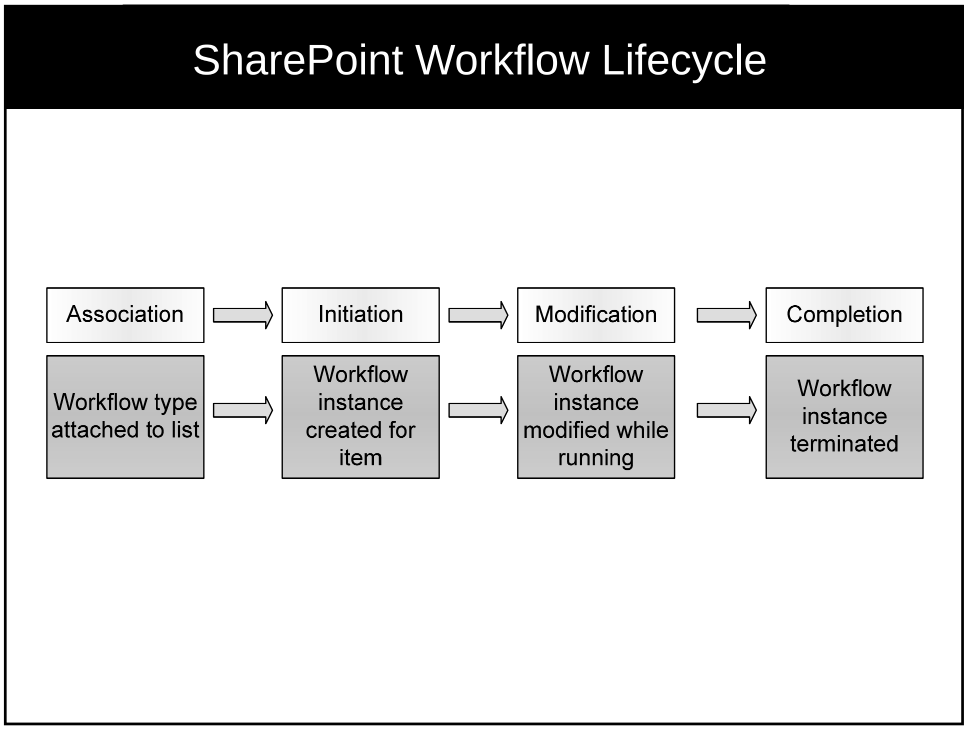
- Enables high-level modeling of custom requirements
- Empowers users to build custom workflows
- Requires higher skills to design & build



Workflow in SharePoint

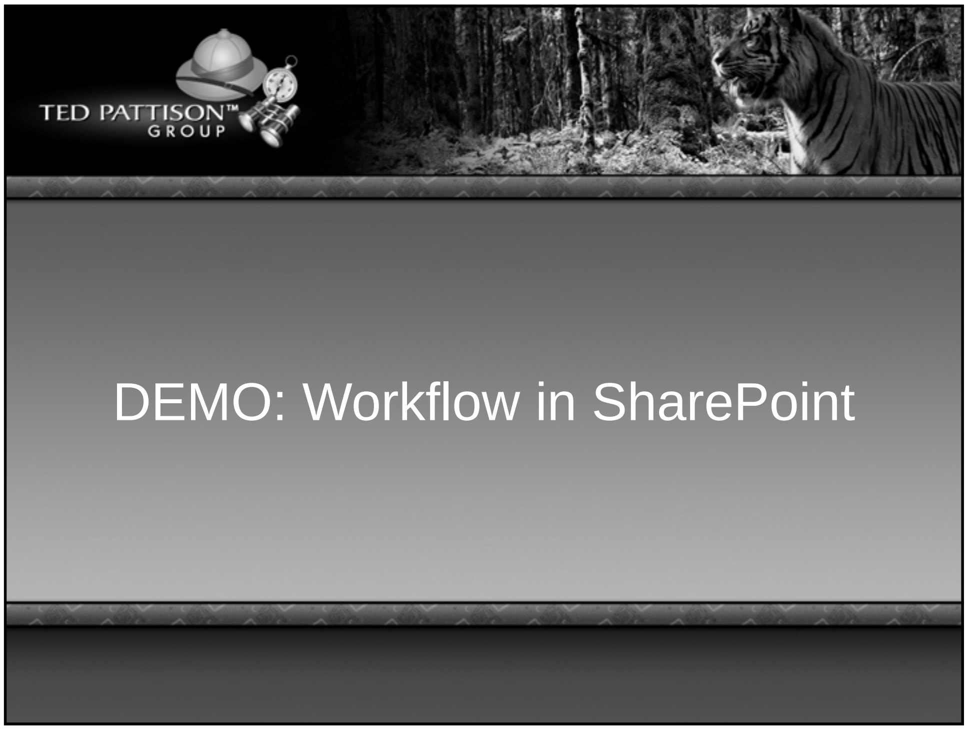
Primary Characteristics

- Fully integrated with Windows Workflow Foundation
  - Every SharePoint List Item
  - Multiple Workflows per Item
- Fully integrated with InfoPath and Forms Services
  - Custom Workflow Association Forms
  - Custom Workflow Initiation Forms
  - Custom Workflow Task Forms
  - Custom Workflow Modification Forms
- Completely Extensible
  - Supports custom lists and content types



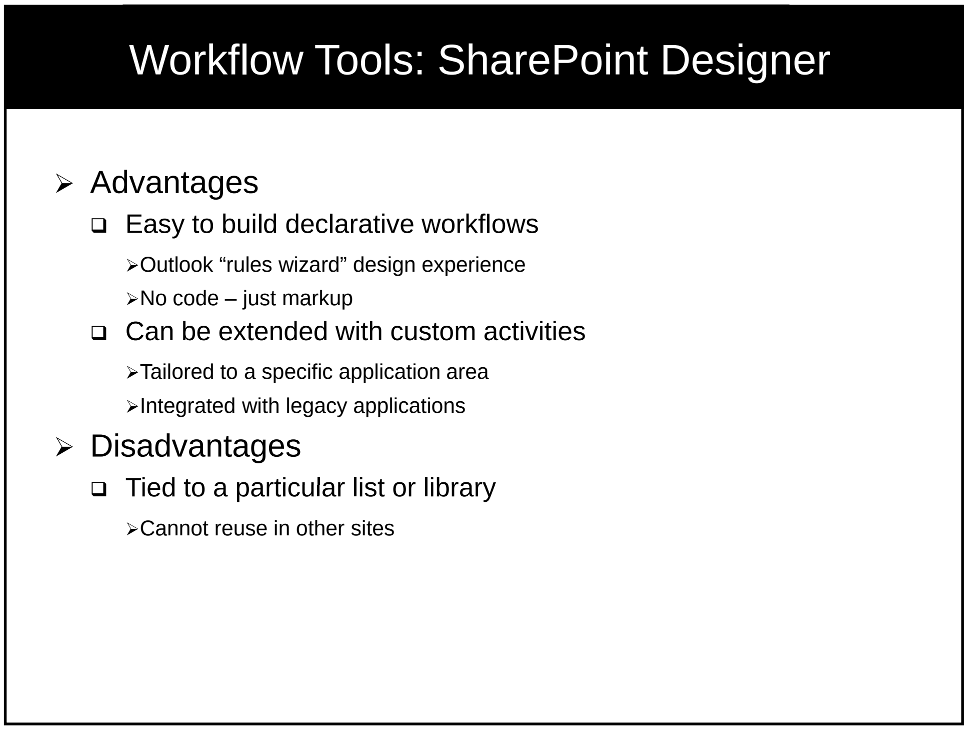
SharePoint Workflow Lifecycle

$$\begin{tikzcd}
\text{Association} \arrow[r] & \text{Initiation} \arrow[r] & \text{Modification} \arrow[r] & \text{Completion} \\
\text{Workflow type attached to list} \arrow[r] & \text{Workflow instance created for item} \arrow[r] & \text{Workflow instance modified while running} \arrow[r] & \text{Workflow instance terminated}
\end{tikzcd}$$



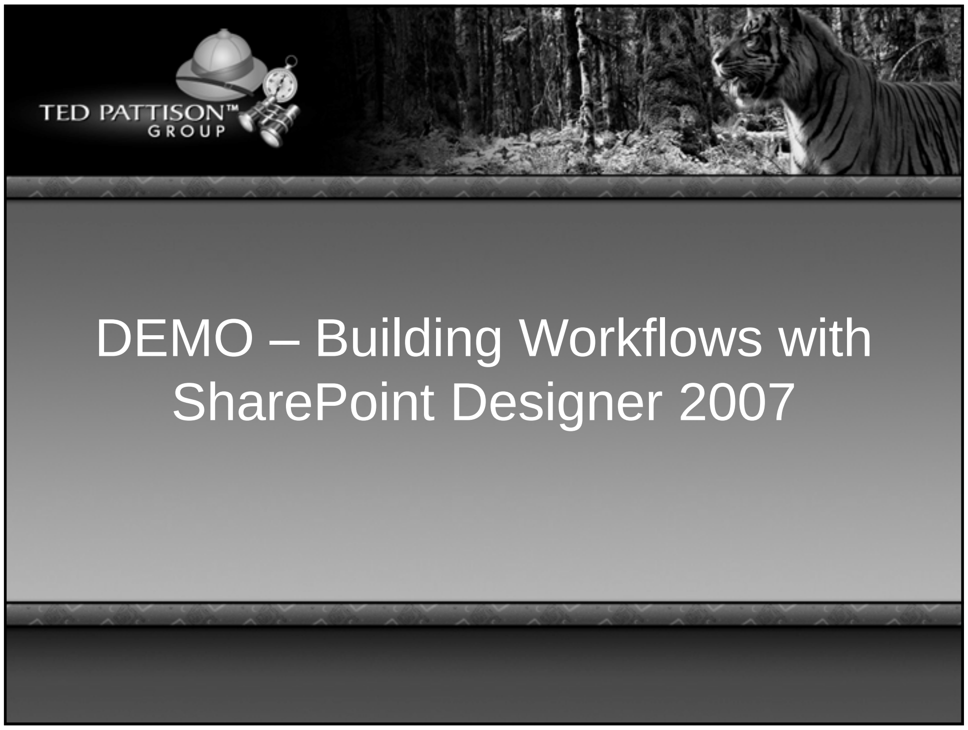
TED PATTISON™
GROUP

DEMO: Workflow in SharePoint



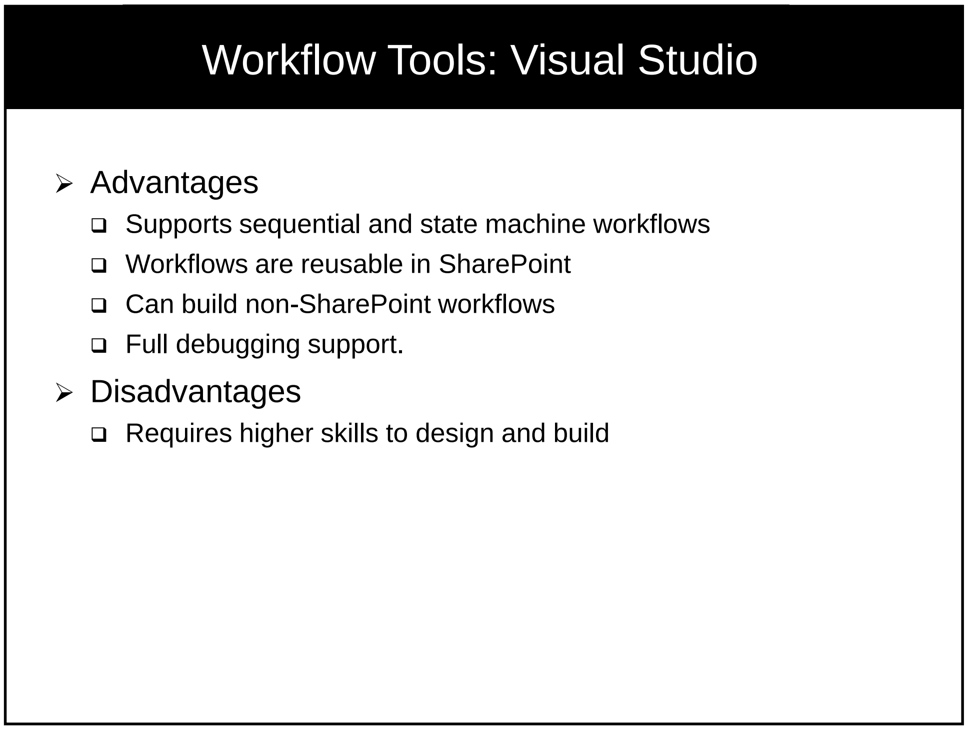
Workflow Tools: SharePoint Designer

- Advantages
  - Easy to build declarative workflows
    - Outlook “rules wizard” design experience
    - No code – just markup
  - Can be extended with custom activities
    - Tailored to a specific application area
    - Integrated with legacy applications
- Disadvantages
  - Tied to a particular list or library
    - Cannot reuse in other sites



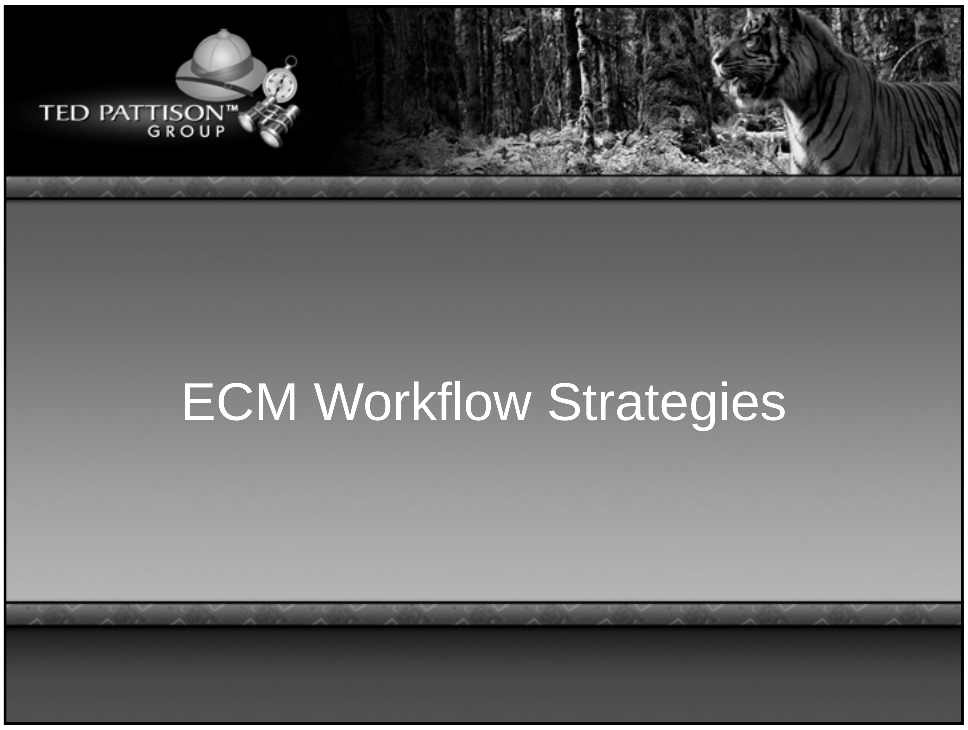
TED PATTISON™ GROUP

DEMO – Building Workflows with SharePoint Designer 2007



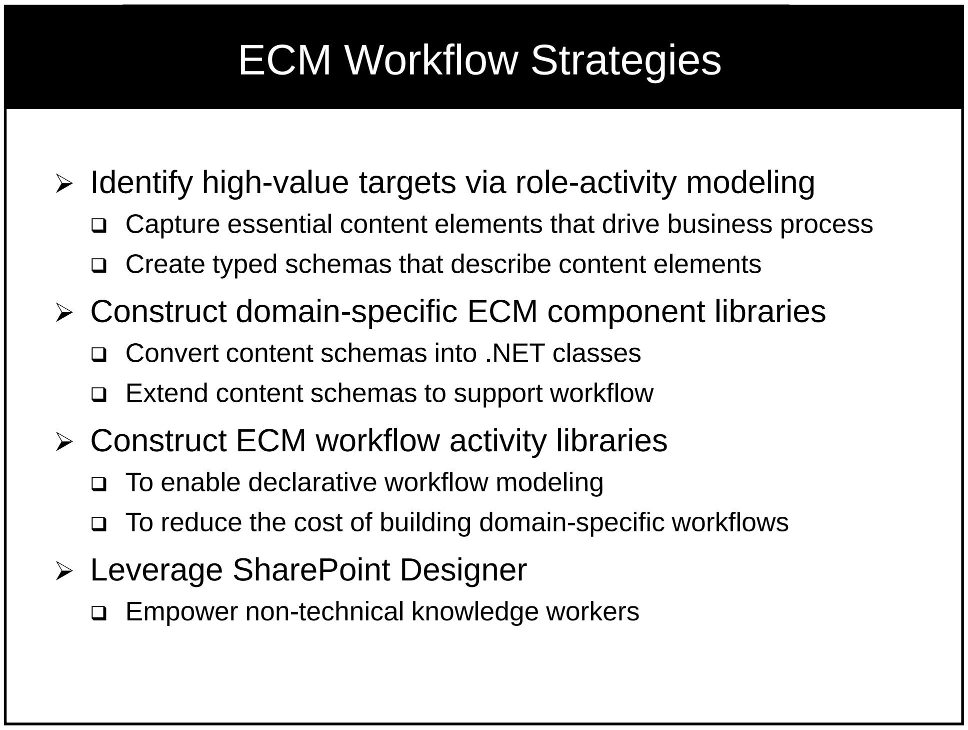
Workflow Tools: Visual Studio

- Advantages
  - Supports sequential and state machine workflows
  - Workflows are reusable in SharePoint
  - Can build non-SharePoint workflows
  - Full debugging support.
- Disadvantages
  - Requires higher skills to design and build



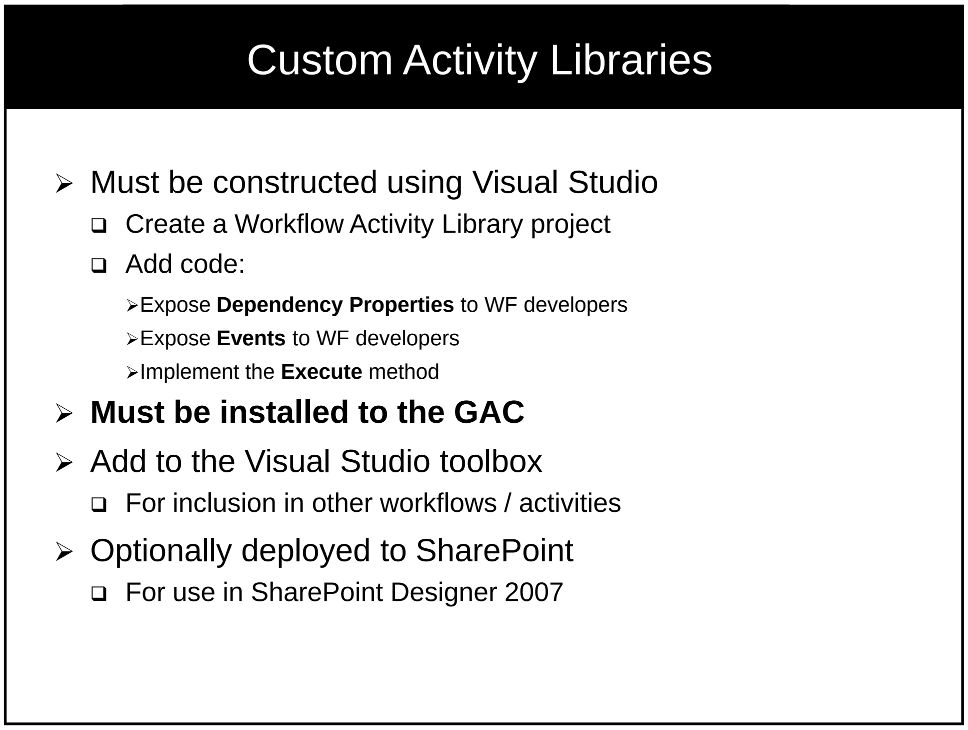
TED PATTISON™ GROUP

ECM Workflow Strategies



ECM Workflow Strategies

- Identify high-value targets via role-activity modeling
  - Capture essential content elements that drive business process
  - Create typed schemas that describe content elements
- Construct domain-specific ECM component libraries
  - Convert content schemas into .NET classes
  - Extend content schemas to support workflow
- Construct ECM workflow activity libraries
  - To enable declarative workflow modeling
  - To reduce the cost of building domain-specific workflows
- Leverage SharePoint Designer
  - Empower non-technical knowledge workers



Custom Activity Libraries

- Must be constructed using Visual Studio
  - Create a Workflow Activity Library project
  - Add code:
    - Expose **Dependency Properties** to WF developers
    - Expose **Events** to WF developers
    - Implement the **Execute** method
- **Must be installed to the GAC**
- Add to the Visual Studio toolbox
  - For inclusion in other workflows / activities
- Optionally deployed to SharePoint
  - For use in SharePoint Designer 2007

Extending SharePoint Designer

- Steps:
  1. Create custom activity assembly
  2. Sign and deploy assembly to the GAC
  3. Configure SharePoint to recognize the activity
  4. Create a .ACTIONS file for SharePoint Designer

Configuring Trusted Assemblies

- Add a special web.config entry

```
<System.Workflow.ComponentModel.WorkflowCompiler>
    <authorizedType Assembly=“MyAssembly, Version=1.0.0.0,
                Culture=neutral, PublicKeyToken=0b97b340d4a71524”
                namespace=“MyCustomActivity” TypeName=“*”
                Authorized=“True”/>
```

Creating a .ACTIONS file

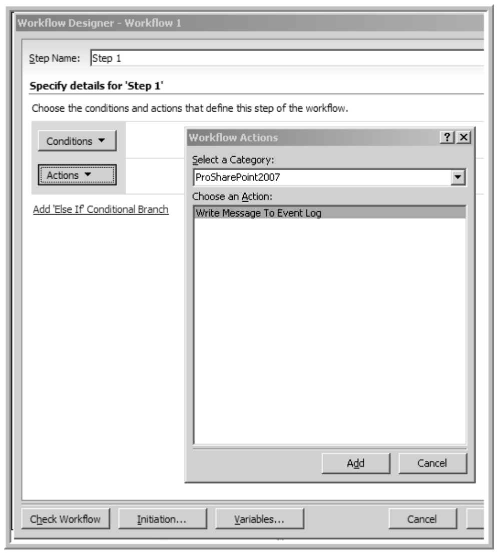
```
<?xml version="1.0" encoding="utf-8" ?>
<WorkflowInfo>
<Actions Sequential="then" Parallel="and">
    <Action Name="Write Message To Event Log"
            ClassName="JohnHolliday.Workflow.EventLogger“
            Assembly="JohnHolliday.Workflow.EventLoggerActivity, Version=1.0.0.0, Culture=neutral,
                      PublicKeyToken=0b97b340d4a71524“
            AppliesTo="all"
            Category="MyCustomActivities">
            <RuleDesigner Sentence="Write '%1' to the event log">
                <FieldBind Field="Message" DesignerType="TextArea" Id="1"/>
            </RuleDesigner>

            <Parameters>
                <Parameter Name="Message" Type="System.String, mscorlib" Direction="In"/>
            </Parameters>
    </Action>
</Actions>
</WorkflowInfo>
```

Shows up in SPD Workflow Designer

Binds to the dependency property in the custom activity

Tells SPD how to invoke the activity

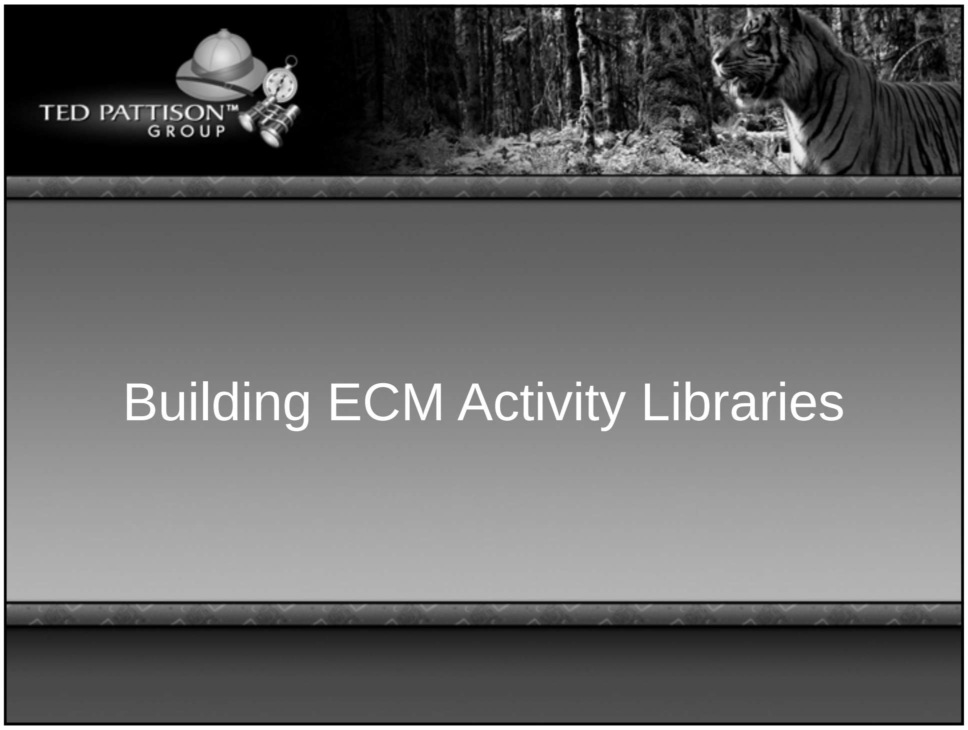
Custom Activity in SPD


Workflow Designer - Workflow 1
Specify details for 'Step 1'
Choose the conditions and actions that define this step of the workflow.
Conditions
Actions
Add 'Else If' Conditional Branch
Workflow Actions
Add
Cancel
Check Workflow
Initiation...
Variables...
Cancel

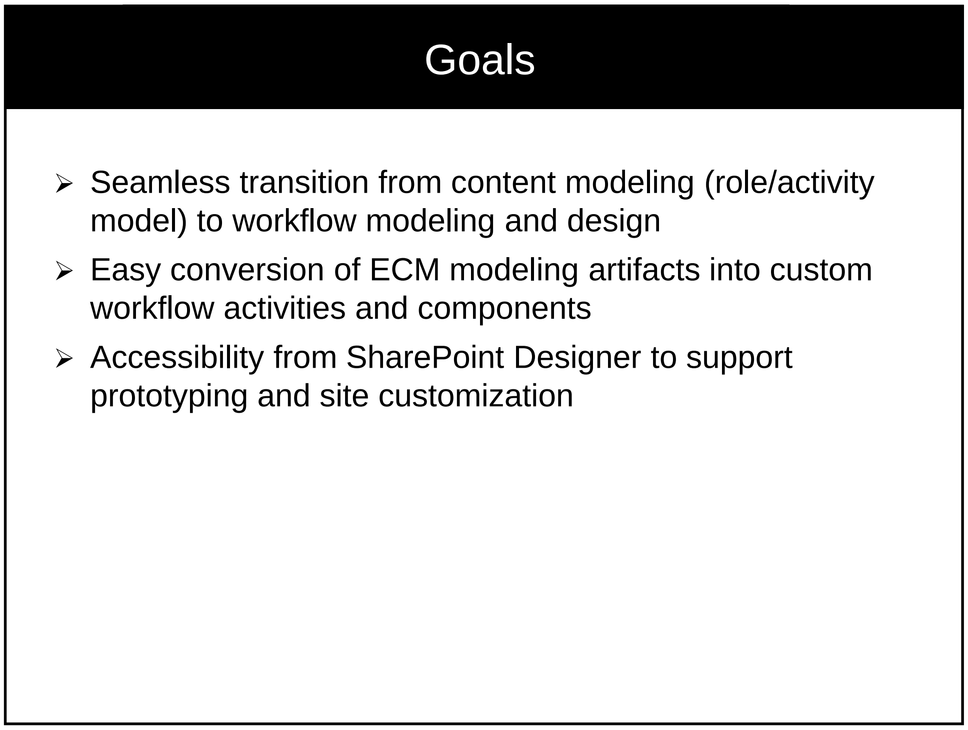
Step Name: Step 1

Select a Category: ProSharePoint2007

Choose an Action: Write Message To Event Log



Building ECM Activity Libraries

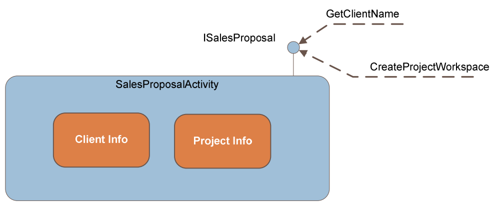


Goals

- ➢ Seamless transition from content modeling (role/activity model) to workflow modeling and design
- ➢ Easy conversion of ECM modeling artifacts into custom workflow activities and components
- ➢ Accessibility from SharePoint Designer to support prototyping and site customization

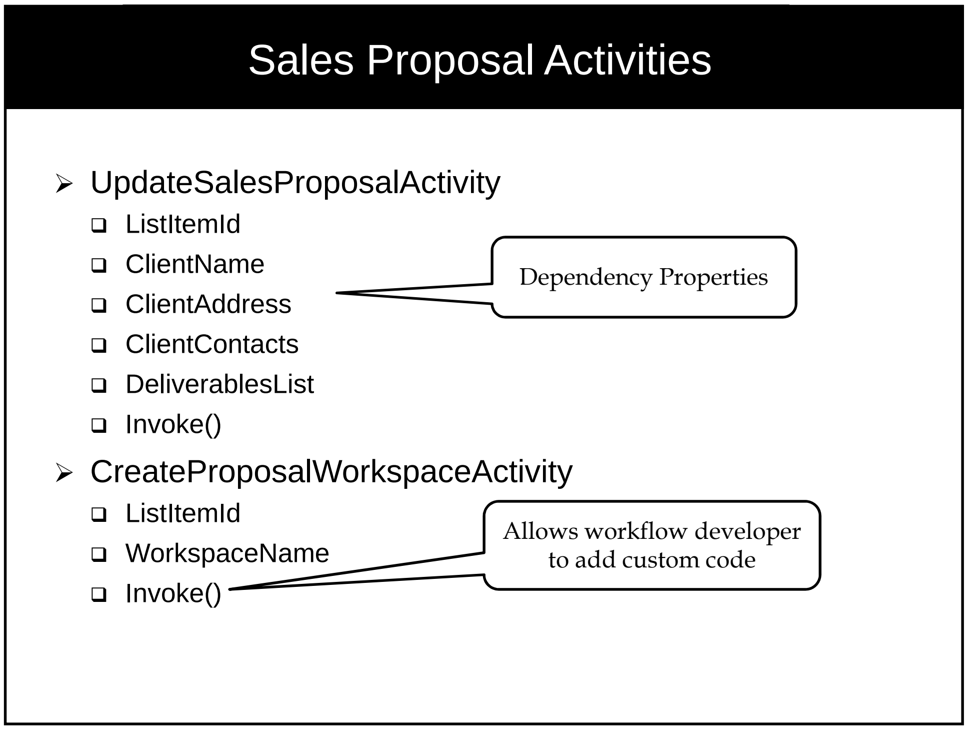
Identifying Useful Activities

- Get/Set structured data
  - Access XML form data from internal schema
- Perform High-Level Operations



Activities + Schema

- Create separate classes for related activities
- Expose dependency properties in the activity
- Using the XML serializer to retrieve ListItem data
- Implement dependency properties

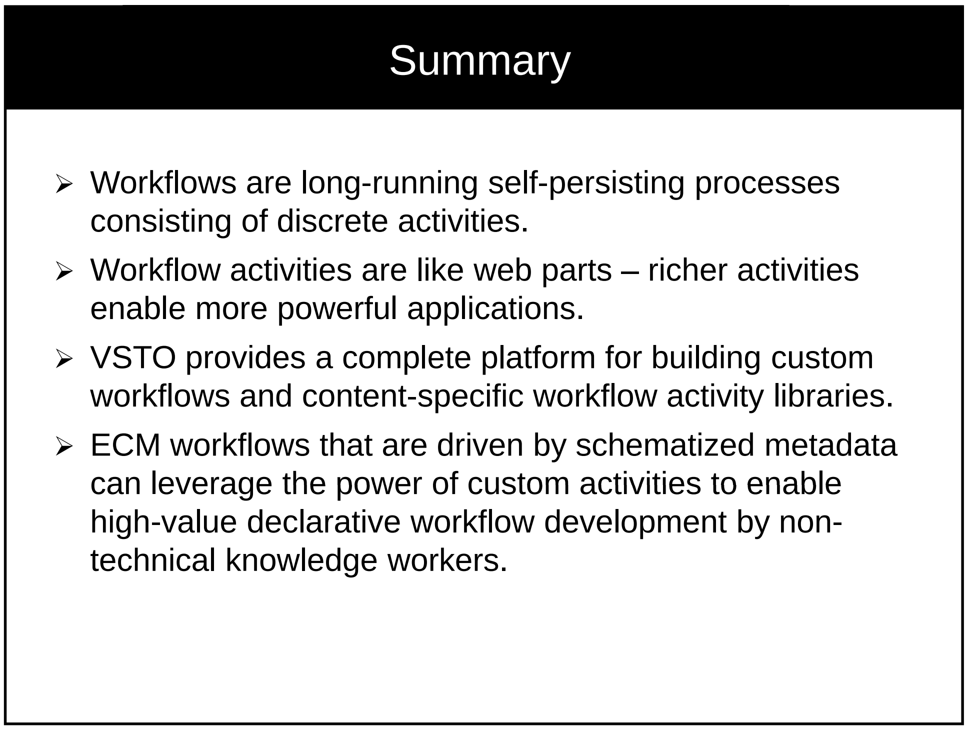


Sales Proposal Activities

- UpdateSalesProposalActivity
  - ListItemId
  - ClientName
  - ClientAddress
  - ClientContacts
  - DeliverablesList
  - Invoke()
- CreateProposalWorkspaceActivity
  - ListItemId
  - WorkspaceName
  - Invoke()

Dependency Properties

Allows workflow developer to add custom code

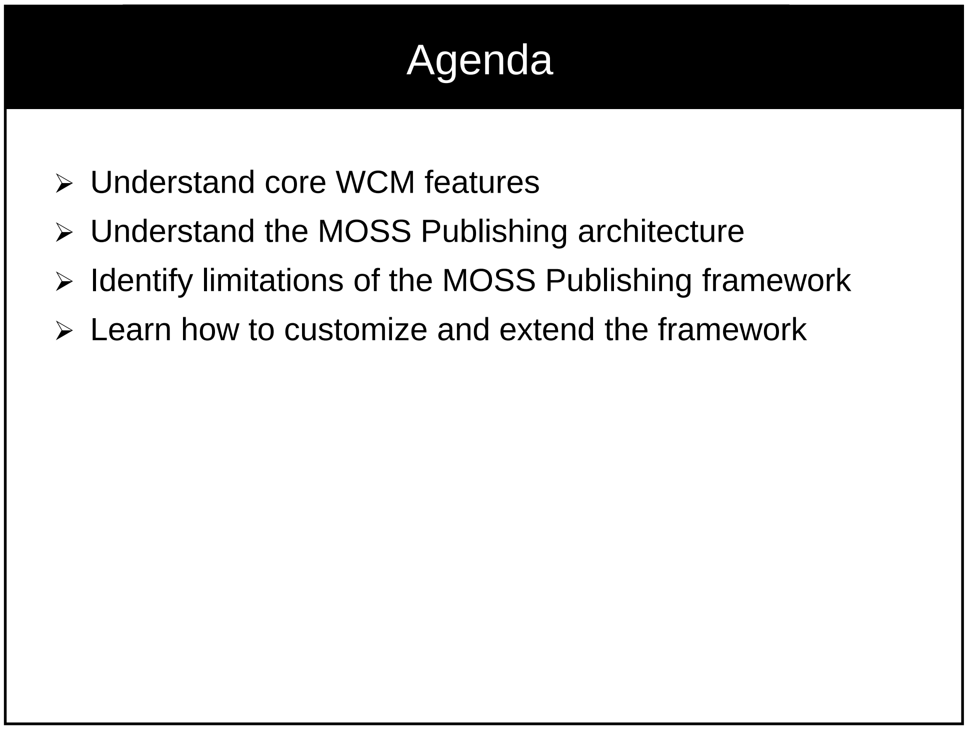


Summary

- Workflows are long-running self-persisting processes consisting of discrete activities.
- Workflow activities are like web parts – richer activities enable more powerful applications.
- VSTO provides a complete platform for building custom workflows and content-specific workflow activity libraries.
- ECM workflows that are driven by schematized metadata can leverage the power of custom activities to enable high-value declarative workflow development by non-technical knowledge workers.

TED PATTISON™ GROUP

Web Content Management in Office SharePoint Server 2007

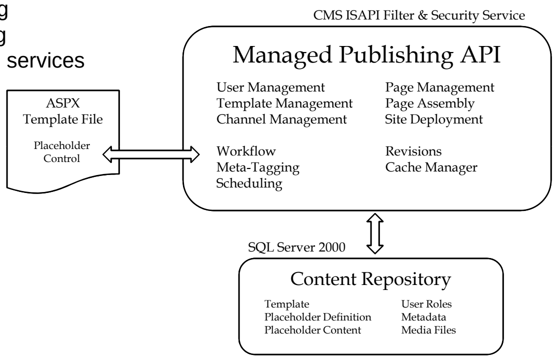


Agenda

- Understand core WCM features
- Understand the MOSS Publishing architecture
- Identify limitations of the MOSS Publishing framework
- Learn how to customize and extend the framework

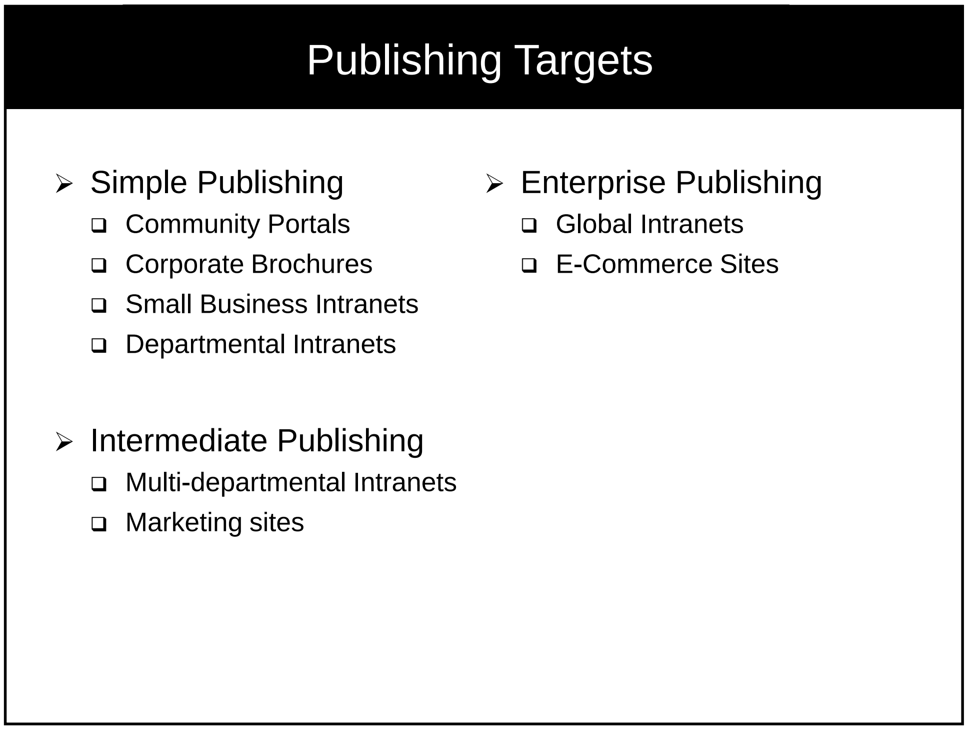
Content Management Server 2002

- Previous Microsoft Publishing Solution ~ 40K
  - Enabled users to create, publish and manage content
  - Out-of-the-box features:
    - Scheduled content refresh
    - Integrated publishing workflow
    - Revision tracking
    - Content indexing
    - XML-based web services



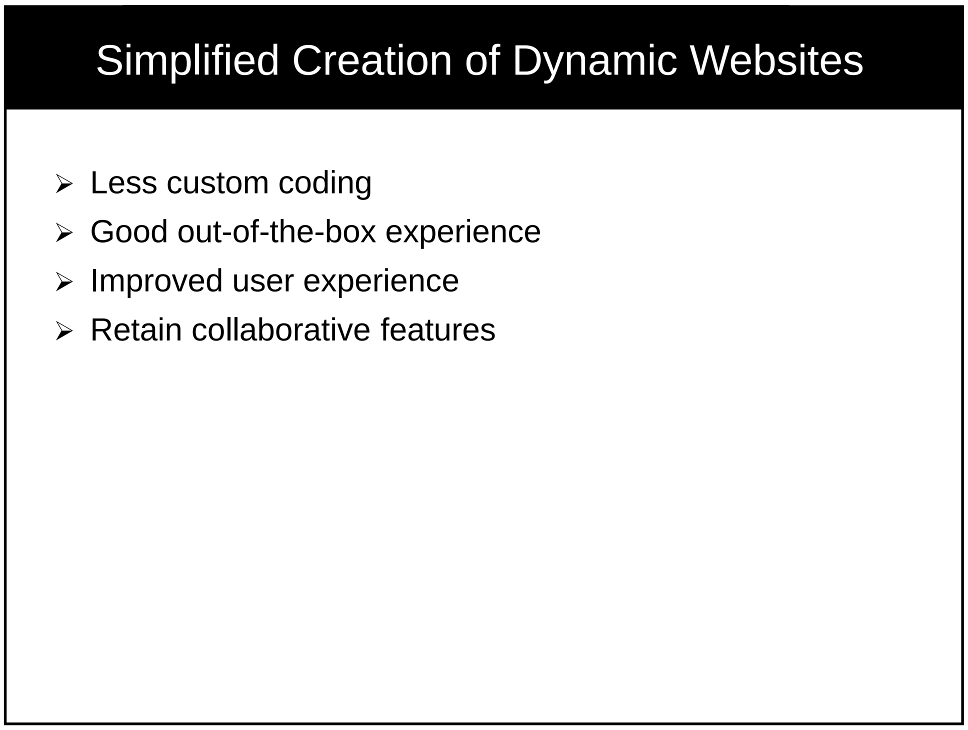
MOSS Publishing Design Goals

- Eliminate the perceived choice between CMS and SharePoint
- Implement the Content Management Server product as an example of a SharePoint 3.0 application
- Provide an integrated ECM solution development platform
  - Simplify the creation of dynamic websites
- Provide support for three levels of publishing:
  - Simple Publishing
  - Intermediate Publishing
  - Enterprise Publishing



Publishing Targets

- Simple Publishing
  - Community Portals
  - Corporate Brochures
  - Small Business Intranets
  - Departmental Intranets
- Intermediate Publishing
  - Multi-departmental Intranets
  - Marketing sites
- Enterprise Publishing
  - Global Intranets
  - E-Commerce Sites



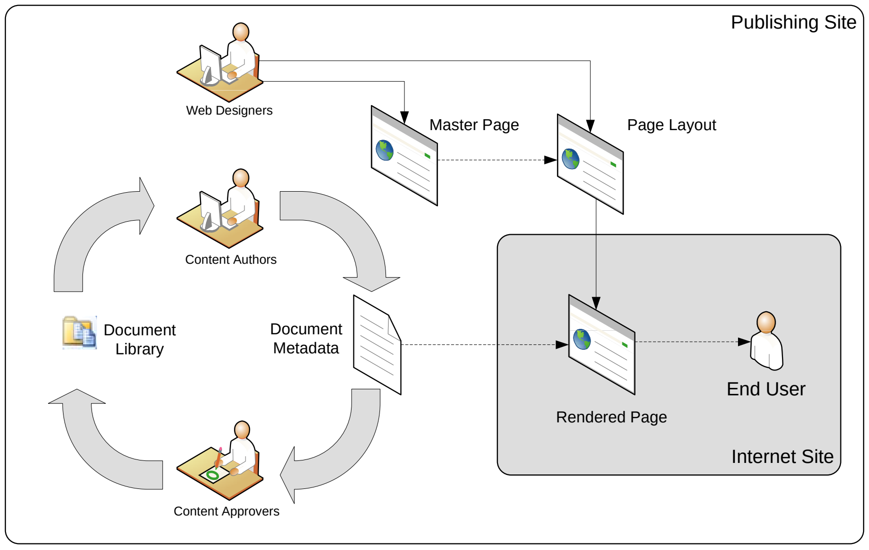
Simplified Creation of Dynamic Websites

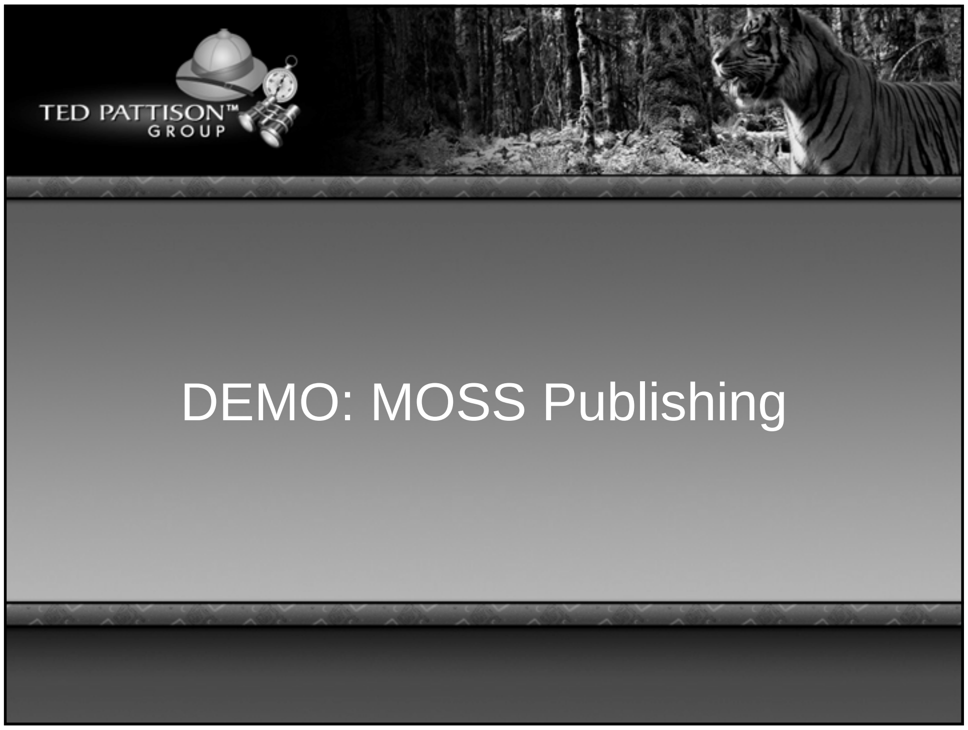
- Less custom coding
- Good out-of-the-box experience
- Improved user experience
- Retain collaborative features

MOSS Publishing Features

- Flexible branding and navigation
  - Leverages ASP.NET 2.0 features
- Decentralized authoring
  - Support for offline editing and multiple authors
- Workflow and scheduling
  - Content is approved before publication
- Parallel content structure via “variations”

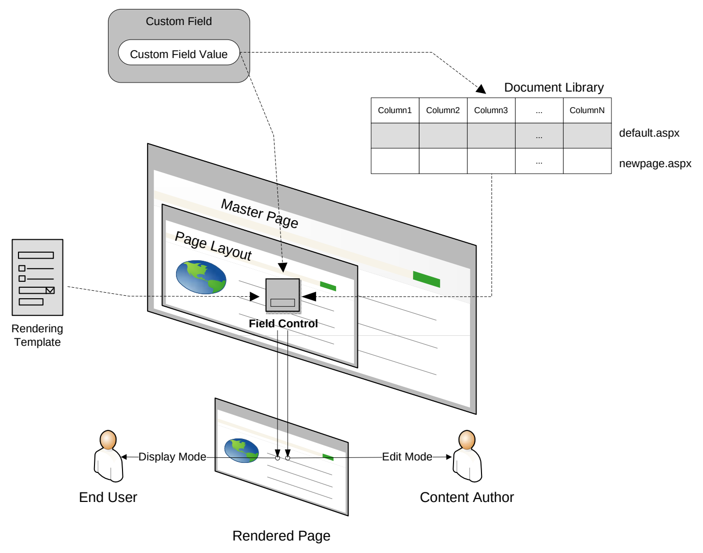
The MOSS Publishing Workflow

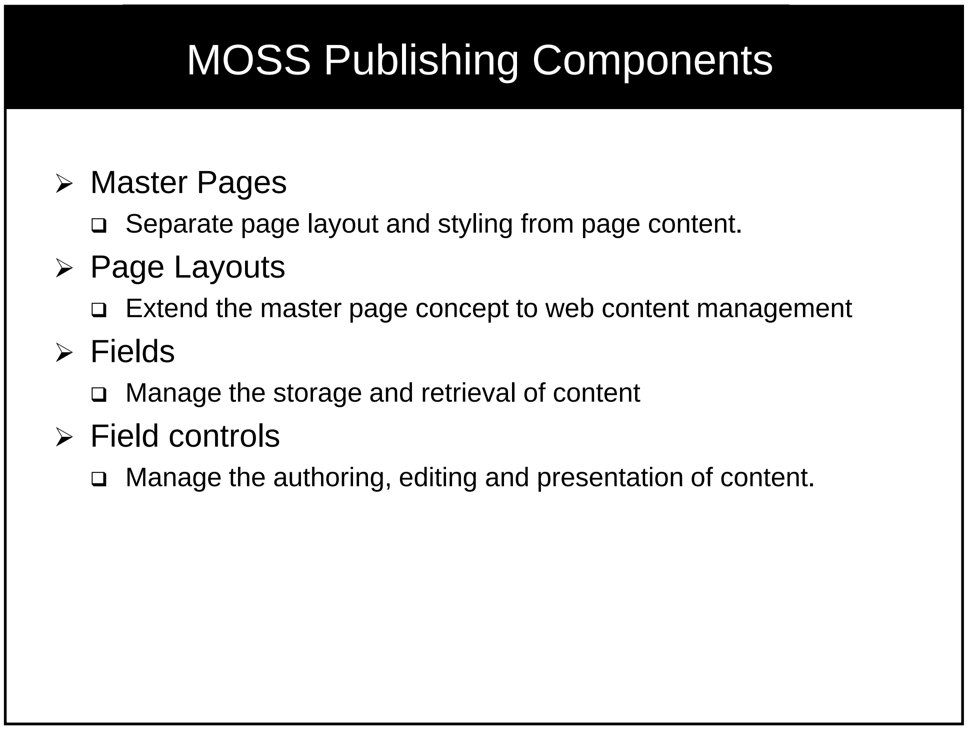




DEMO: MOSS Publishing

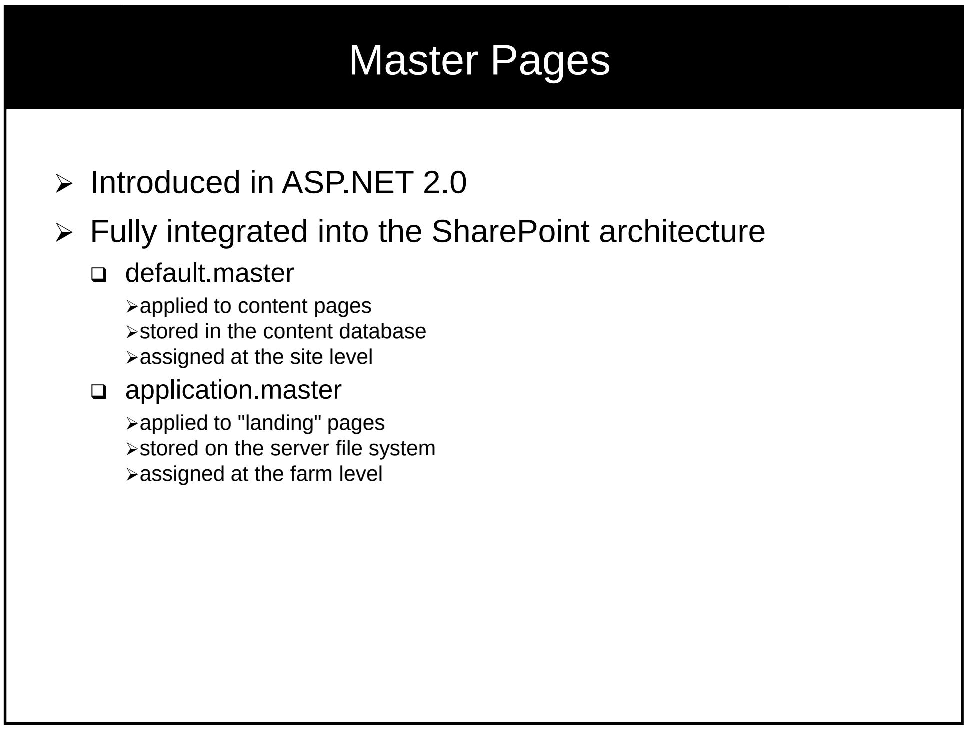
The MOSS Publishing Architecture





MOSS Publishing Components

- Master Pages
  - ❑ Separate page layout and styling from page content.
- Page Layouts
  - ❑ Extend the master page concept to web content management
- Fields
  - ❑ Manage the storage and retrieval of content
- Field controls
  - ❑ Manage the authoring, editing and presentation of content.



Master Pages

- Introduced in ASP.NET 2.0
- Fully integrated into the SharePoint architecture
  - ❑ default.master
    - applied to content pages
    - stored in the content database
    - assigned at the site level
  - ❑ application.master
    - applied to "landing" pages
    - stored on the server file system
    - assigned at the farm level

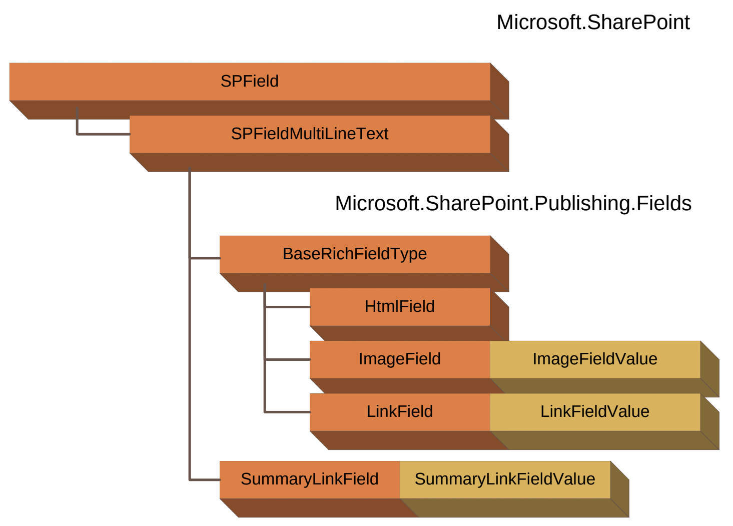
Page Layouts

- Operate like Master Pages for WCM
  - Separate control layout and configuration from content
  - Stored in the master page gallery of publishing site
  - Used as template for page instances
  - Merged with content stored in list columns
- Implemented as ASP.NET content pages
  - Linked to a master page
  - Include static content and ASP.NET controls
  - Include field controls for WCM content

Fields and Field Controls

- Fields and Field Controls are tightly bound
- Fields provide structure for content
  - Control where content is stored
    - Single column
    - Multiple columns
  - Control how content is stored
    - Raw format
    - Structured or delimited format
  - Can transform content in transit
    - Calculated values

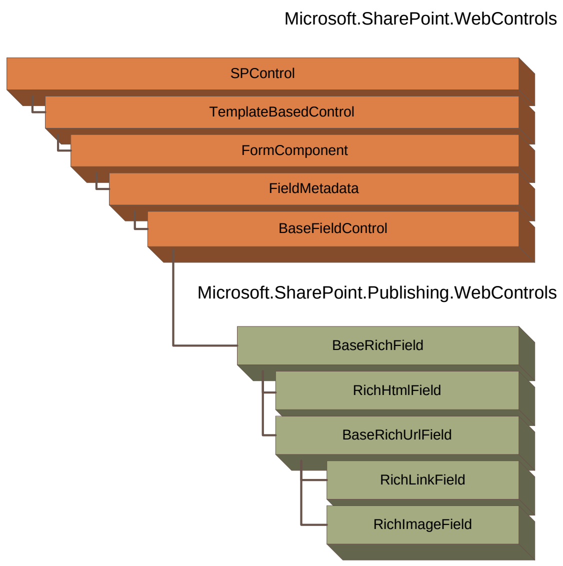
SharePoint Field Classes



Field Controls

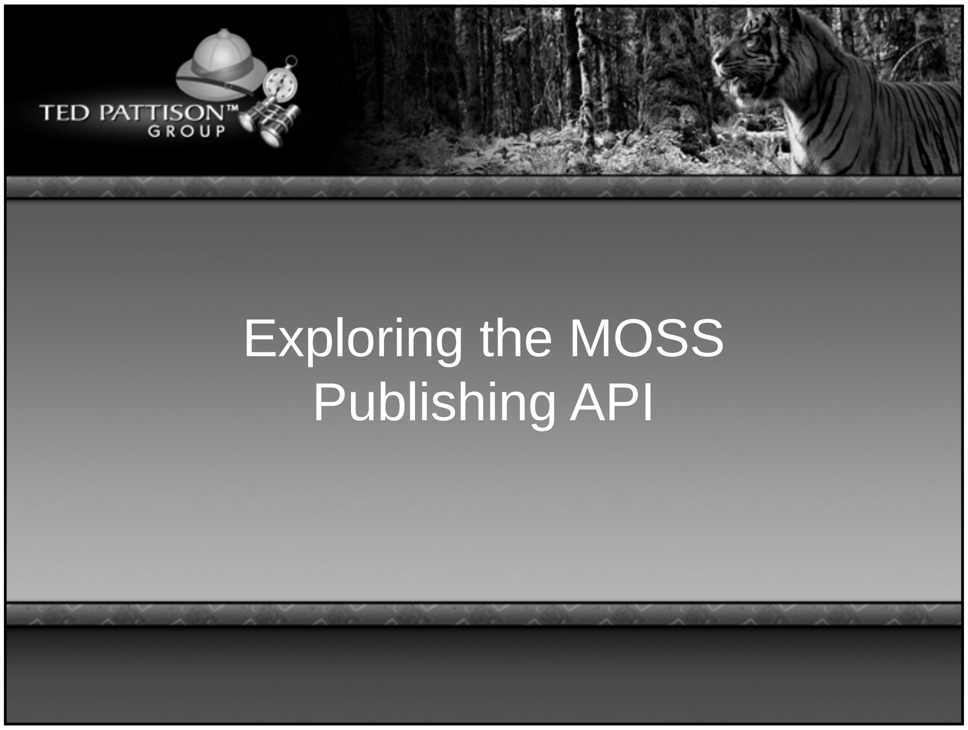
- Provide presentation for managed content
  - Takes advantage of internal knowledge of the associated field.
  - Generally developed in tandem with the associated field.

SharePoint Field Control Classes

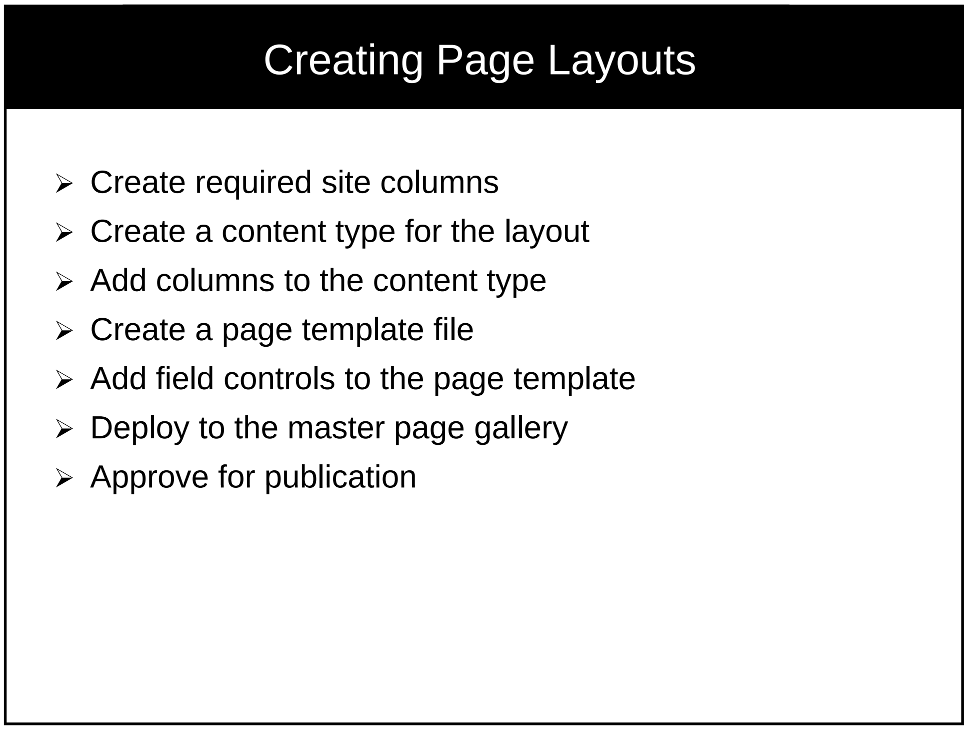


Field Controls vs Web Parts

	Field Control	**Web Part**
Data Storage	Field in the page list item	Web part data associated with the page
Location in Page	Fixed as a control in the page layout	Fixed inside of a web part zone
Versioning	Versioned with the page	Versioned with the page, but without history.
Personalization	No	Yes
Standard Uses	Used to display content stored as page metadata.	Used to display the result of queries or views of external content
Example	Rich HTML Field Image Field Summary Links	Content Query Web Part TOC Web Part



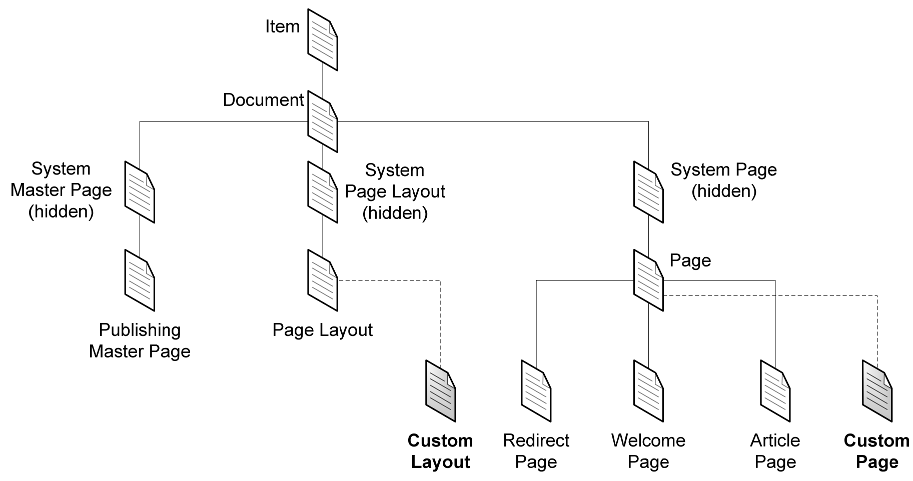
Exploring the MOSS Publishing API



Creating Page Layouts

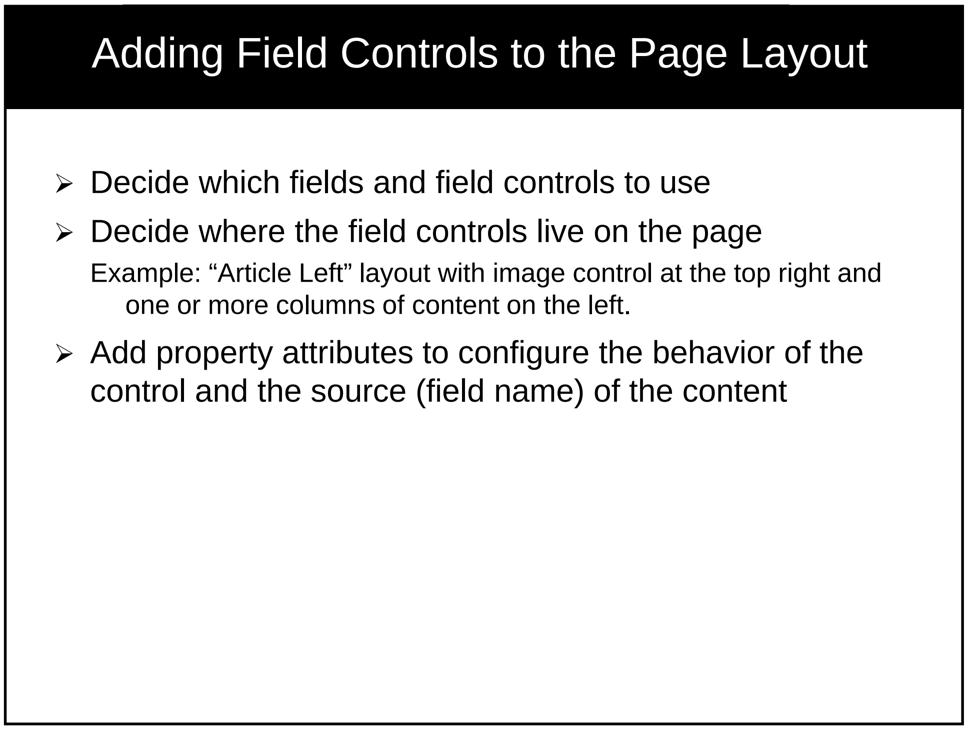
- Create required site columns
- Create a content type for the layout
- Add columns to the content type
- Create a page template file
- Add field controls to the page template
- Deploy to the master page gallery
- Approve for publication

Page Layout Content Types



Creating a Page Layout template file

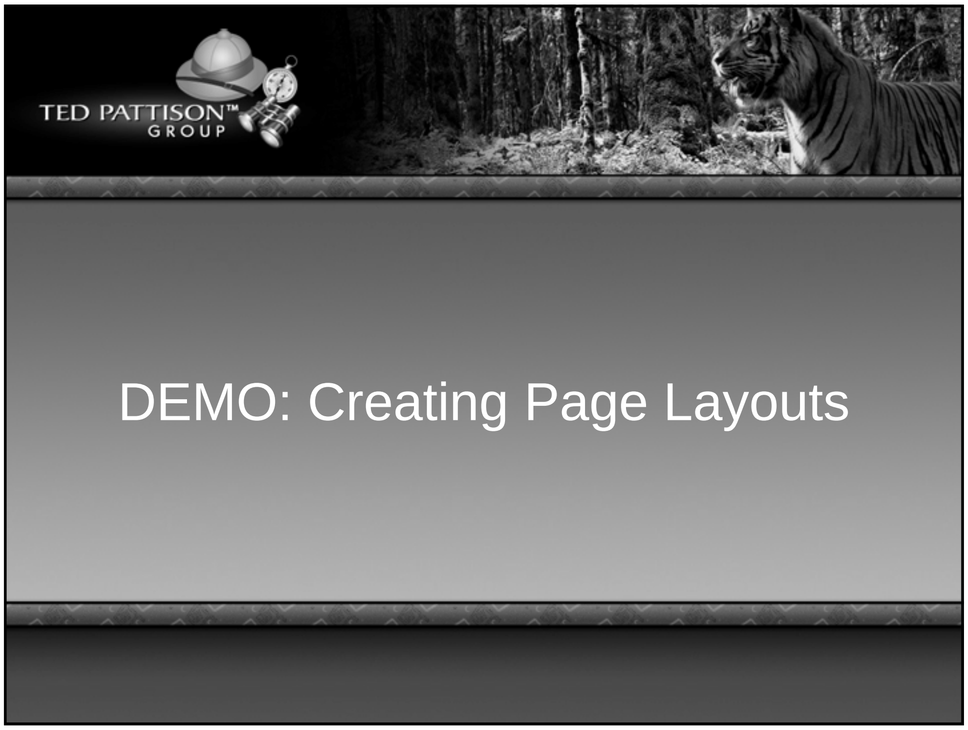
- Using SharePoint Designer
  - Create directly in gallery
  - Check out for editing
- Using Visual Studio
  - Create in web page designer
  - Save to file system for packaging in solution



Adding Field Controls to the Page Layout

- Decide which fields and field controls to use
- Decide where the field controls live on the page

  Example: “Article Left” layout with image control at the top right and one or more columns of content on the left.
- Add property attributes to configure the behavior of the control and the source (field name) of the content



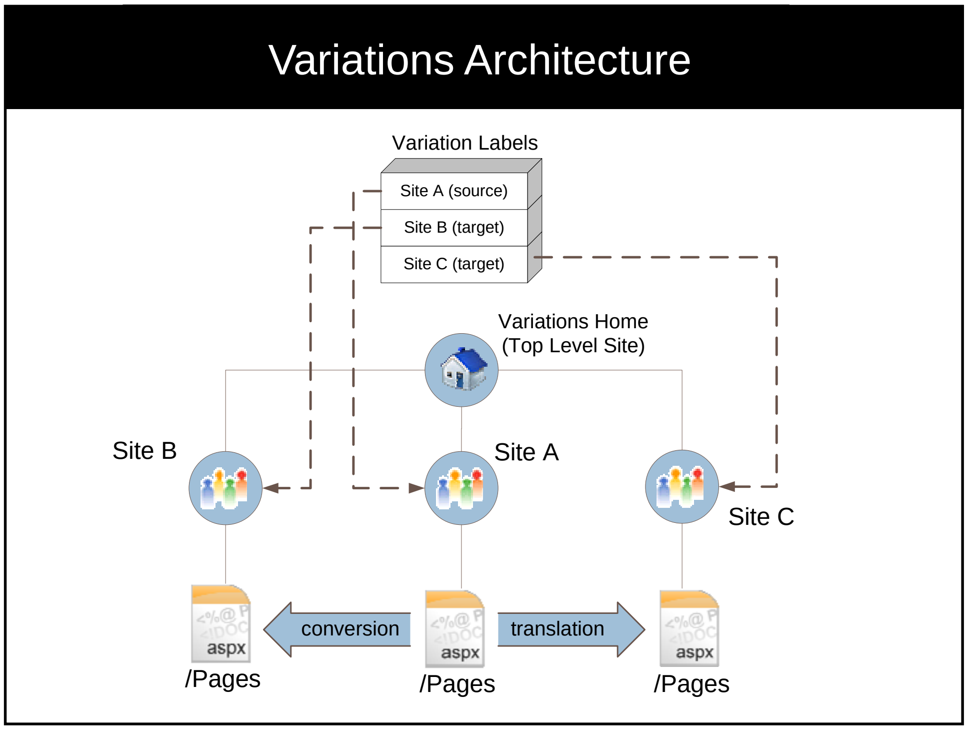
DEMO: Creating Page Layouts

Deploying Page Templates

- Deploy to the master page gallery
  - Using SharePoint Designer
    - Check in to the gallery
    - Approve and accept changes
  - Using Visual Studio
    - Add to SharePoint solution package
    - Add Module element to copy into gallery

Variations

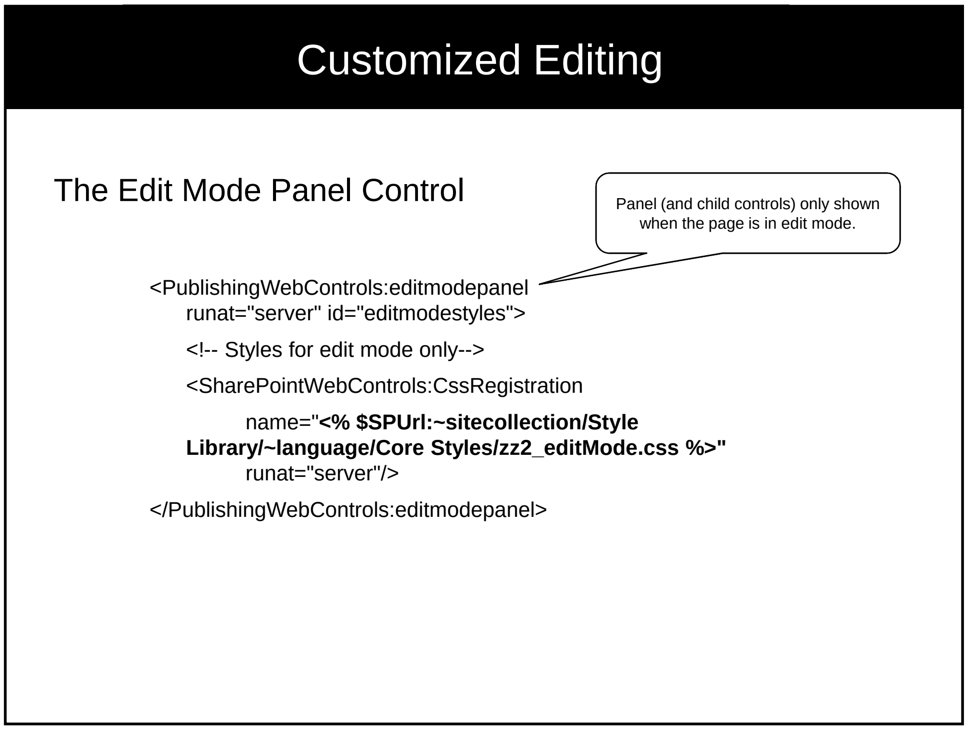
- Enables parallel content structure
  - Administrator creates labels
  - System replicates containers and items when a new label is created
  - System maintains parallel structure as containers and items are created and removed
- Can be used to store content in multiple languages, form factors, brands, etc.



Variations Architecture

Extending the Publishing Framework

TED PATTISON™ GROUP

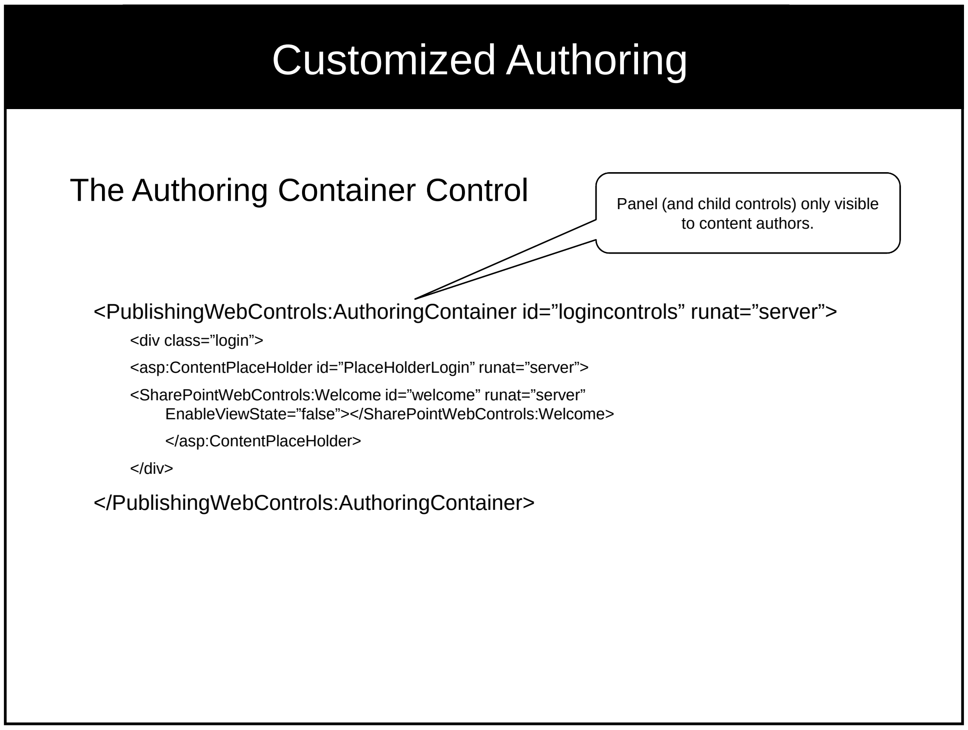


Customized Editing

The Edit Mode Panel Control

Panel (and child controls) only shown when the page is in edit mode.

```
<PublishingWebControls:editmodepanel
    runat="server" id="editmodestyles">
    <!-- Styles for edit mode only-->
    <SharePointWebControls:CssRegistration
          name="<% $SPUrl:~sitecollection/Style
    Library/~language/Core Styles/zz2_editMode.css %>"
          runat="server"/>
</PublishingWebControls:editmodepanel>
```



Customized Authoring

The Authoring Container Control

Panel (and child controls) only visible to content authors.

```
<PublishingWebControls:AuthoringContainer id=”logincontrols” runat=”server”>
    <div class=”login”>
    <asp:ContentPlaceHolder id=”PlaceHolderLogin” runat=”server”>
    <SharePointWebControls:Welcome id=”welcome” runat=”server”
        EnableViewState=”false”></SharePointWebControls:Welcome>
        </asp:ContentPlaceHolder>
    </div>
</PublishingWebControls:AuthoringContainer>
```

Client-Based Authoring

- Using Microsoft Office Word as a publishing tool
  - Needs server-side support => document conversion
- Document Converters
  - Specially written .EXE
  - Runs on the server – installed by administrator
  - Runs in a sandboxed application domain
  - Command-line parameters specify conversion operation

Building a Document Converter

- Implementing a custom document converter requires the following steps:

  1. Create an EXE that performs the conversion.
  2. Register the converter in the SharePoint environment.
  3. Optionally create a custom ASPX page to allow the user to provide converter-specific settings.

- The converter EXE file must accept the following parameters:

  -in infile

  The input file containing the document to be converted.

  -out outfile.html

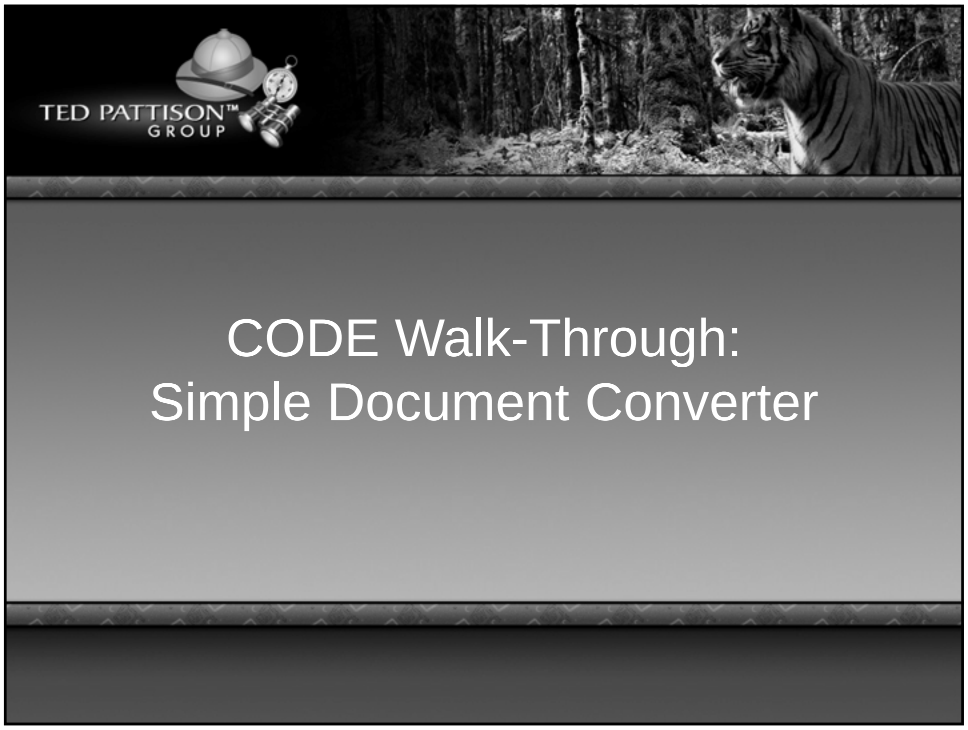
  The fully qualified path to the output HTML file.

  [-config config.xml]

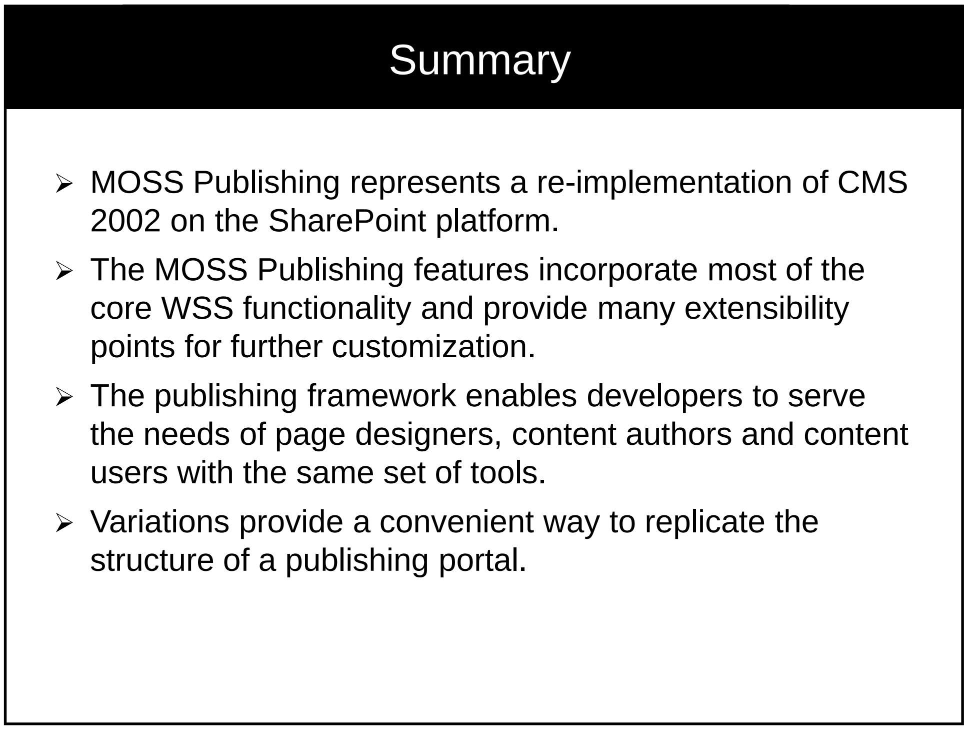
  (optional) path to the configuration settings

  [-log logfile.log]

  (optional) path to the log file

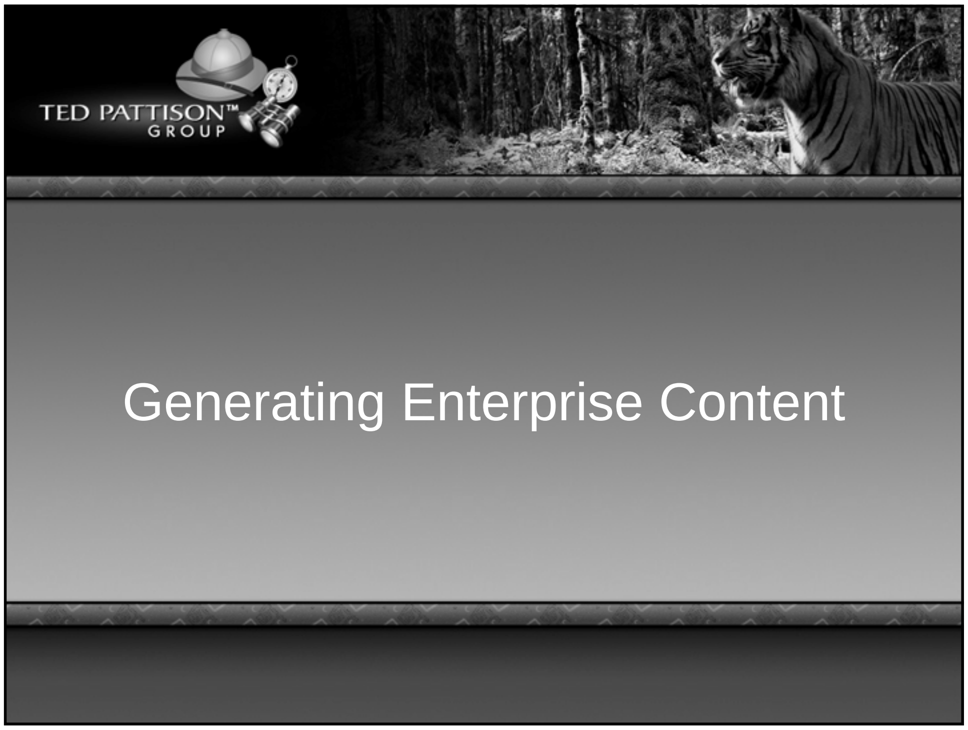


CODE Walk-Through: Simple Document Converter

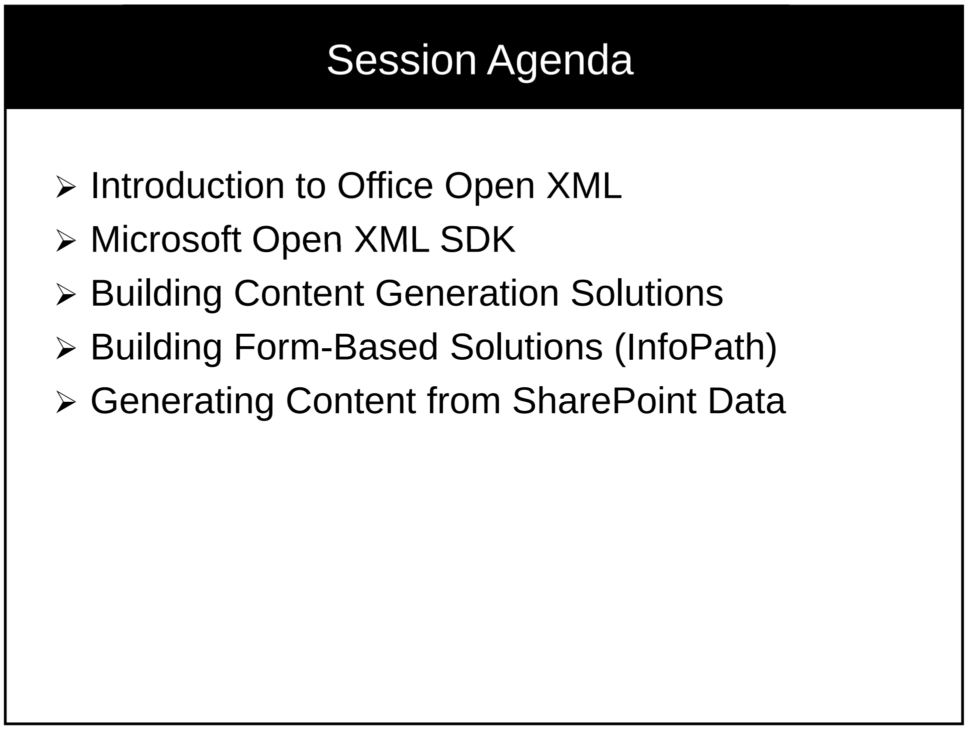


Summary

- MOSS Publishing represents a re-implementation of CMS 2002 on the SharePoint platform.
- The MOSS Publishing features incorporate most of the core WSS functionality and provide many extensibility points for further customization.
- The publishing framework enables developers to serve the needs of page designers, content authors and content users with the same set of tools.
- Variations provide a convenient way to replicate the structure of a publishing portal.



Generating Enterprise Content



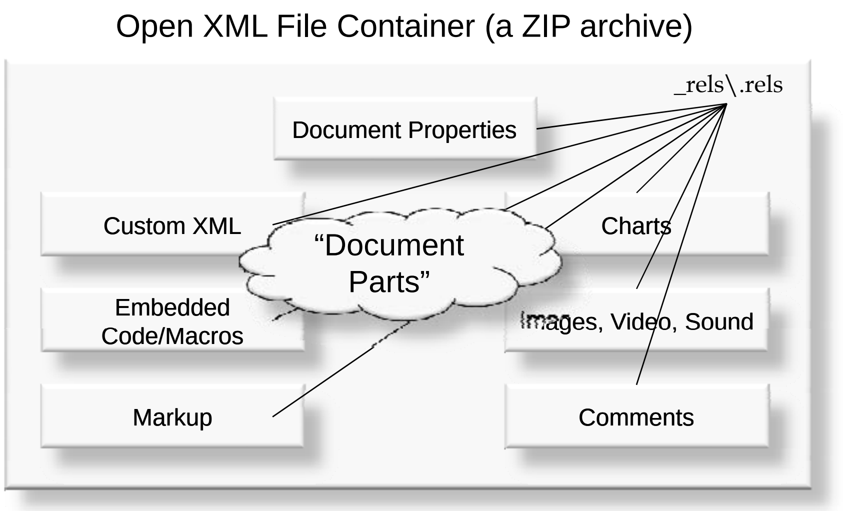
Session Agenda

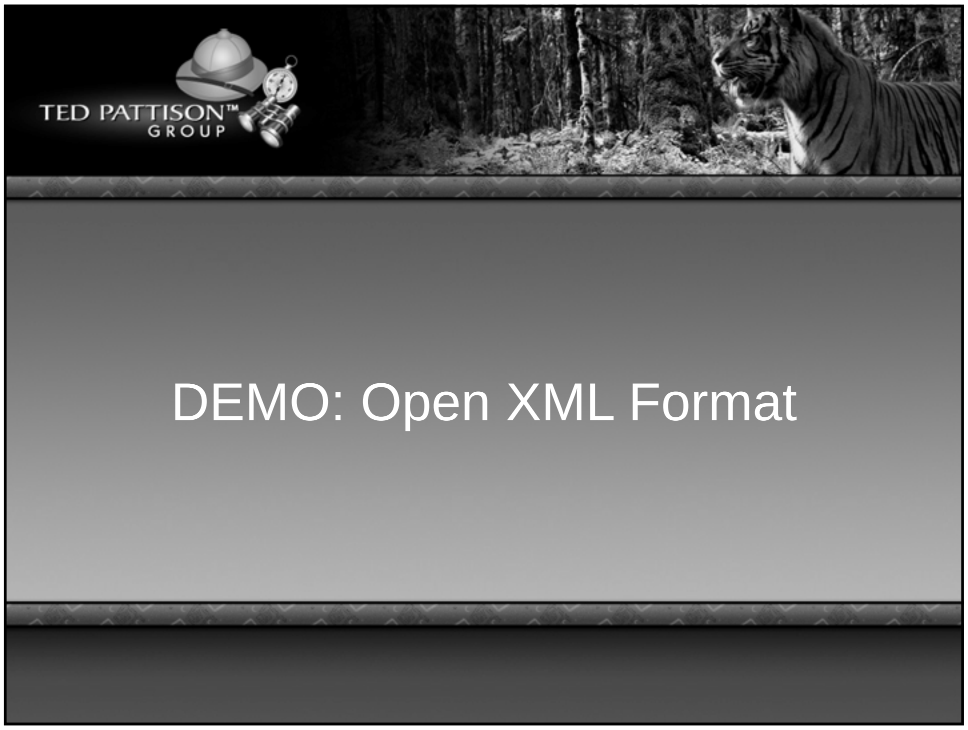
- Introduction to Office Open XML
- Microsoft Open XML SDK
- Building Content Generation Solutions
- Building Form-Based Solutions (InfoPath)
- Generating Content from SharePoint Data

Office Open XML?

- ECMA Standard XML-Based Format
- Replaces existing Office file formats
  - Different schema for each product
    - WordProcessingML, SpreadsheetML, etc.
- Can be used to:
  - Validate documents
  - Search for unstructured data
  - Modify data within documents
  - Generate new documents

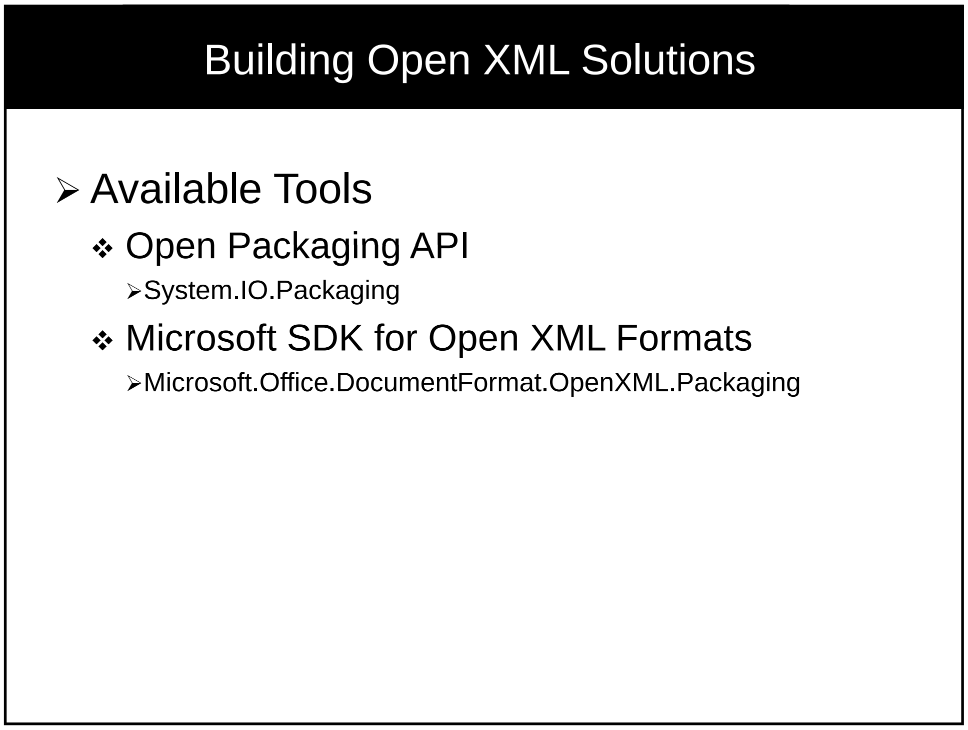
General Architecture





TED PATTISON™ GROUP

DEMO: Open XML Format



Building Open XML Solutions

- ➢ Available Tools
  - ❖ Open Packaging API
    - ➢System.IO.Packaging
  - ❖ Microsoft SDK for Open XML Formats
    - ➢Microsoft.Office.DocumentFormat.OpenXML.Packaging

Open Packaging API

- System.IO.Packaging
  - Applies to all Office applications
  - Lots of steps to create real documents

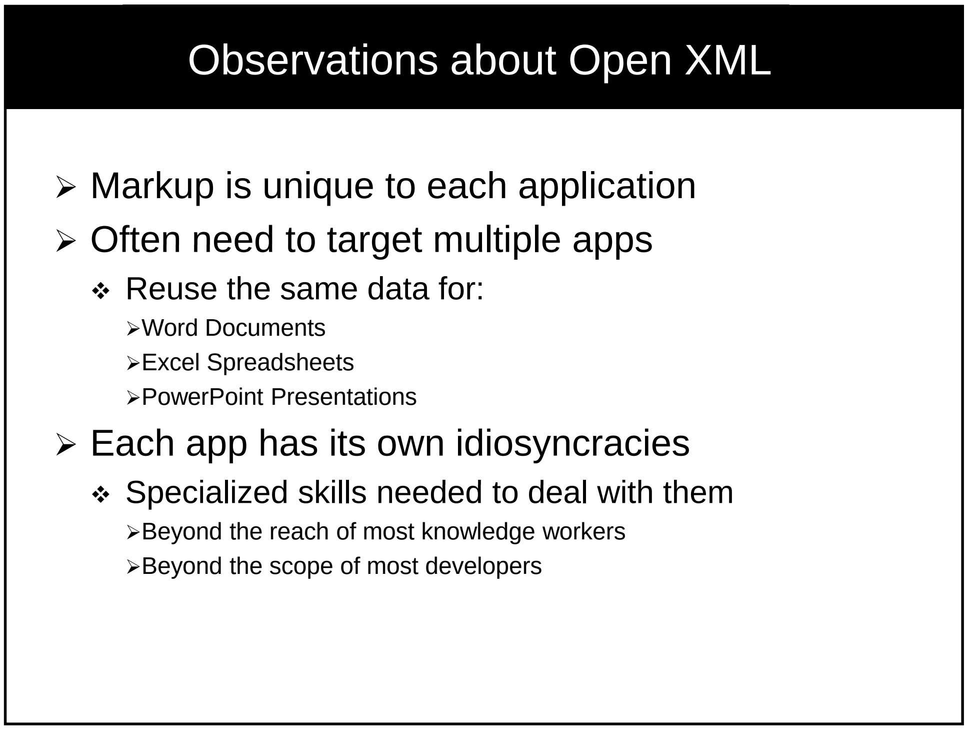
MYDOCUMENT.DOCX **= ZIP Archive**

```
Package pkg = Package.Open(“MyDocument.docx”,
FileMode.Create);
```

Microsoft SDK For Open XML Formats

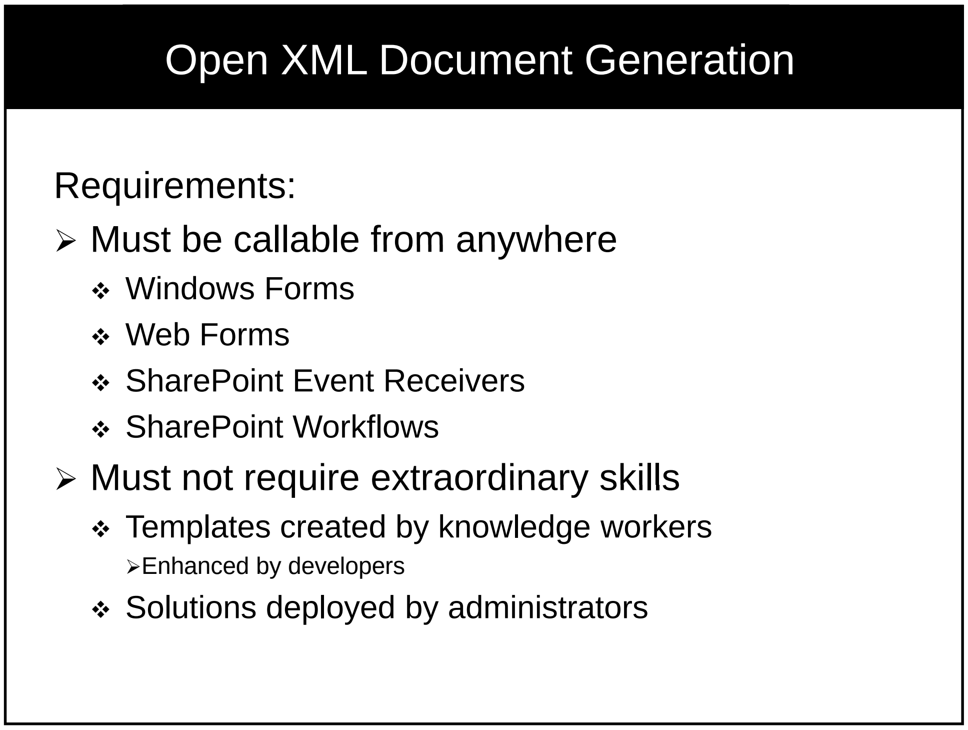
- Microsoft.Office.DocumentFormat.OpenXML.Packaging
  - Built on top of System.IO.Packaging
  - Easier than the raw packaging API
  - Wrapper classes for all Office document types
  - Wrapper classes for all known document parts

Markup	**Class**
WordProcessingML	WordProcessingDocument
SpreadsheetML	SpreadsheetDocument
PresentationML	PresentationDocument



Observations about Open XML

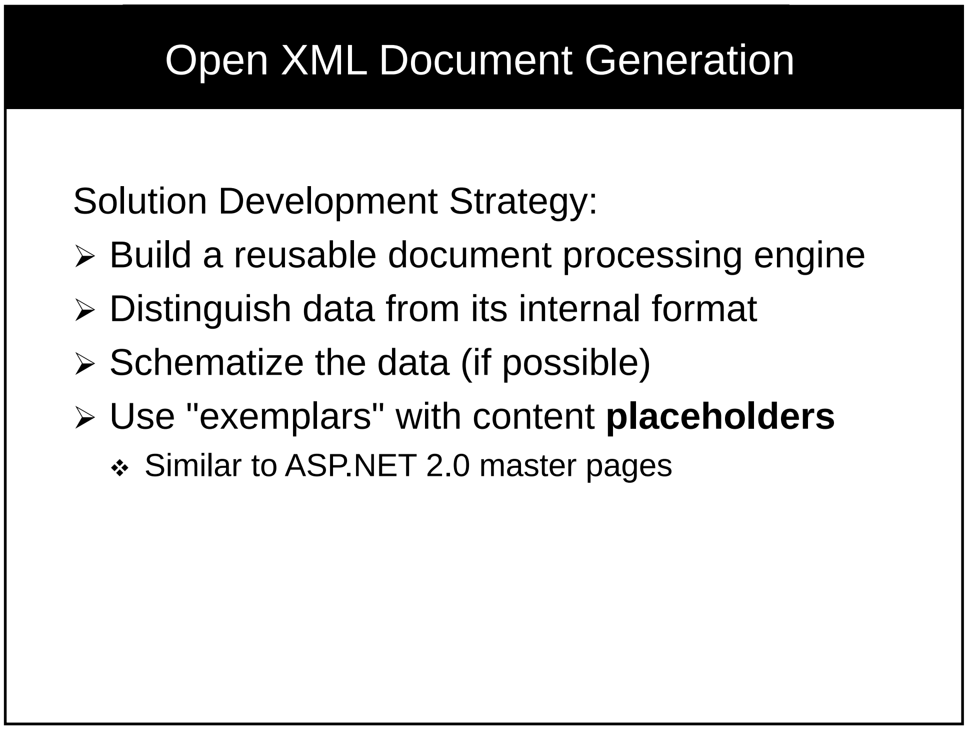
- Markup is unique to each application
- Often need to target multiple apps
  - Reuse the same data for:
    - Word Documents
    - Excel Spreadsheets
    - PowerPoint Presentations
- Each app has its own idiosyncracies
  - Specialized skills needed to deal with them
    - Beyond the reach of most knowledge workers
    - Beyond the scope of most developers



Open XML Document Generation

Requirements:

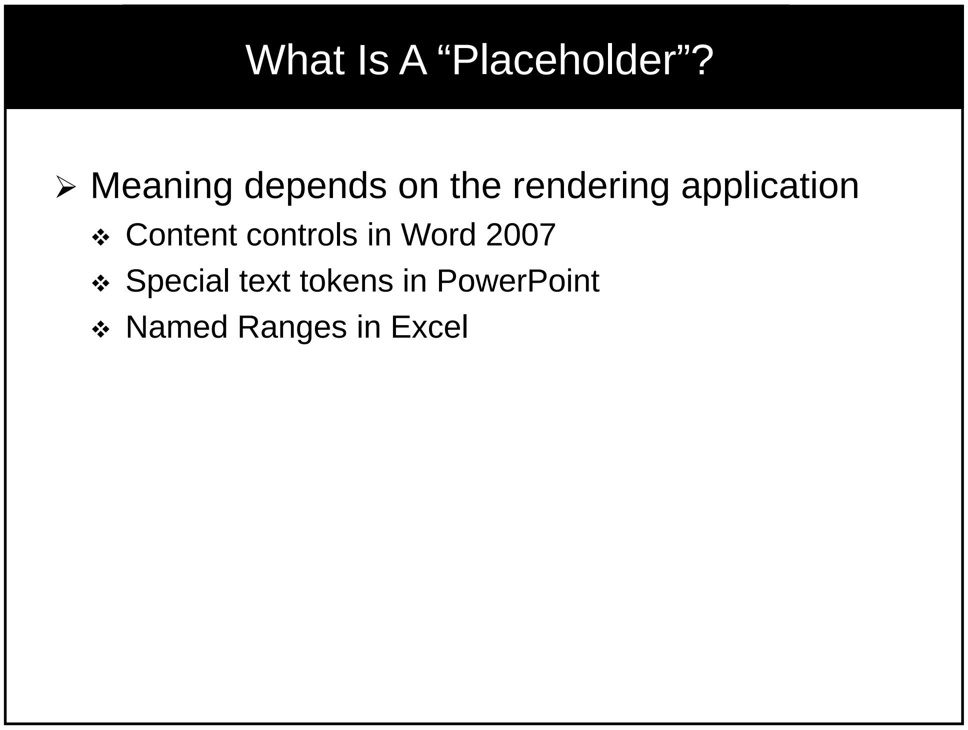
- Must be callable from anywhere
  - Windows Forms
  - Web Forms
  - SharePoint Event Receivers
  - SharePoint Workflows
- Must not require extraordinary skills
  - Templates created by knowledge workers
    - Enhanced by developers
  - Solutions deployed by administrators



Open XML Document Generation

Solution Development Strategy:

- ➢ Build a reusable document processing engine
- ➢ Distinguish data from its internal format
- ➢ Schematize the data (if possible)
- ➢ Use "exemplars" with content **placeholders**
  - ❖ Similar to ASP.NET 2.0 master pages

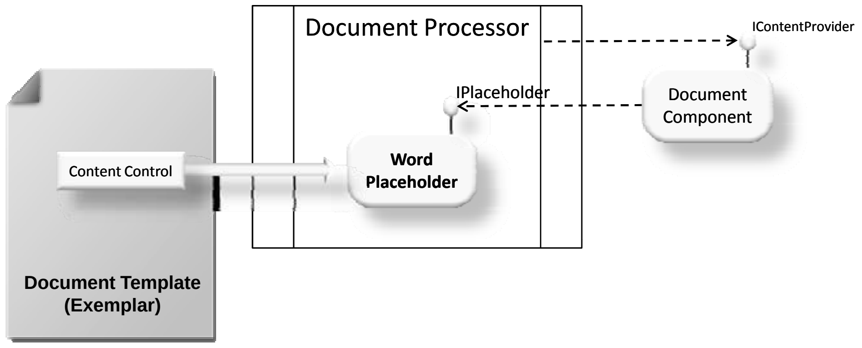


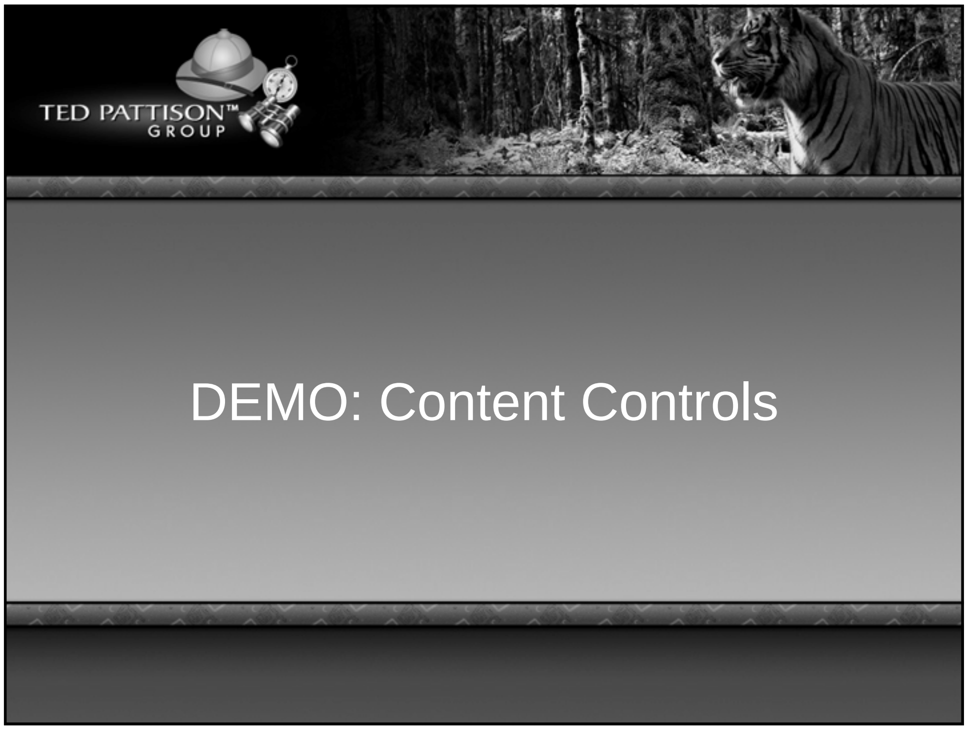
What Is A “Placeholder”?

- ➢ Meaning depends on the rendering application
  - ❖ Content controls in Word 2007
  - ❖ Special text tokens in PowerPoint
  - ❖ Named Ranges in Excel

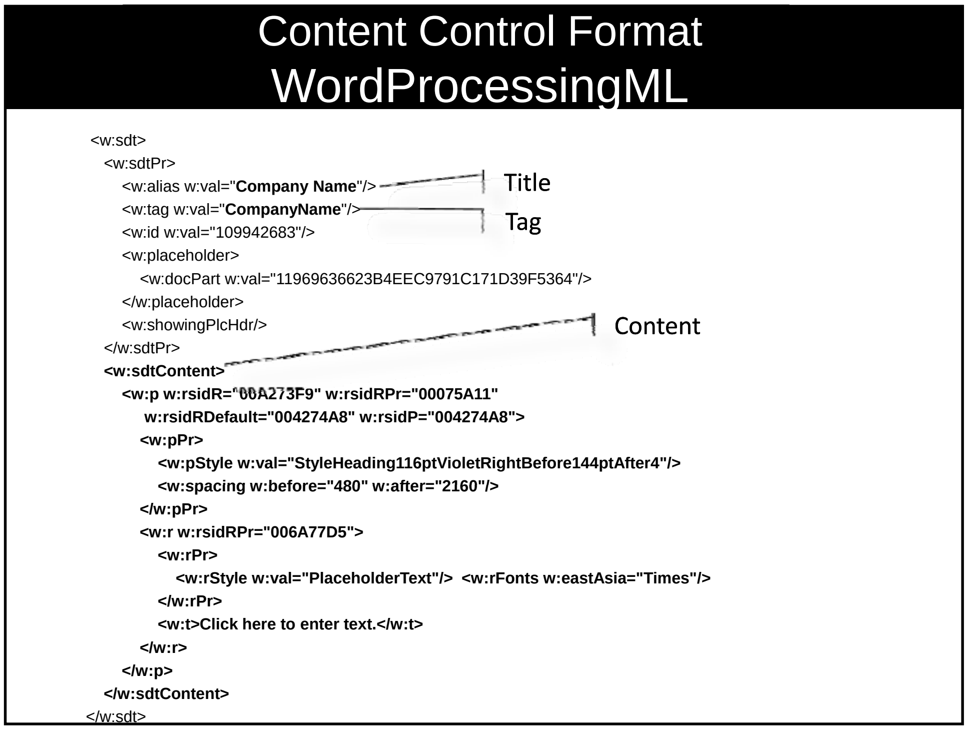
Document Processing: Word 2007

Word 2007 Content Controls as Placeholders





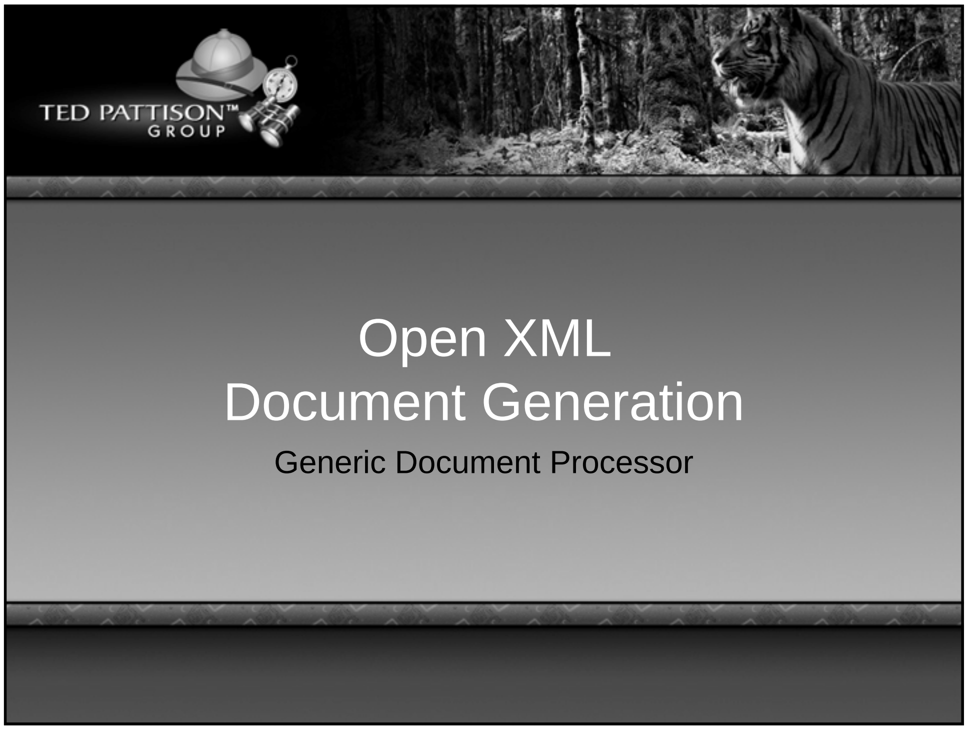
DEMO: Content Controls



Content Control Format WordProcessingML

```
<w:sdt>
  <w:sdtPr>
    <w:alias w:val="Company Name"/>
    <w:tag w:val="CompanyName"/>
    <w:id w:val="109942683"/>
    <w:placeholder>
      <w:docPart w:val="11969636623B4EEC9791C171D39F5364"/>
    </w:placeholder>
    <w:showingPlcHdr/>
  </w:sdtPr>
  <w:sdtContent>
    <w:p w:rsidR="00A273F9" w:rsidRPr="00075A11"
         w:rsidRDefault="004274A8" w:rsidP="004274A8">
      <w:pPr>
        <w:pStyle w:val="StyleHeading116ptVioletRightBefore144ptAfter4"/>
        <w:spacing w:before="480" w:after="2160"/>
      </w:pPr>
      <w:r w:rsidRPr="006A77D5">
        <w:rPr>
          <w:rStyle w:val="PlaceholderText"/>  <w:rFonts w:eastAsia="Times"/>
        </w:rPr>
        <w:t>Click here to enter text.</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
```

Title
Tag
Content

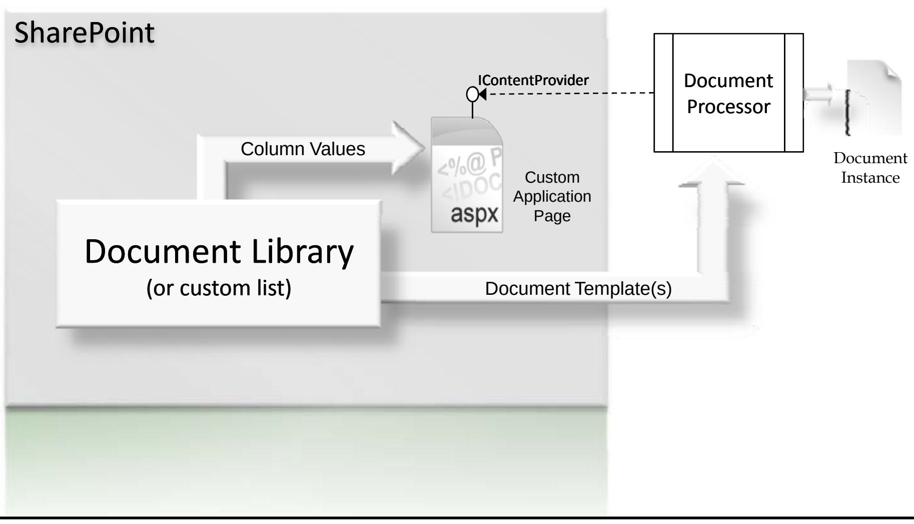


Open XML Document Generation

Generic Document Processor

TED PATTISON™ GROUP

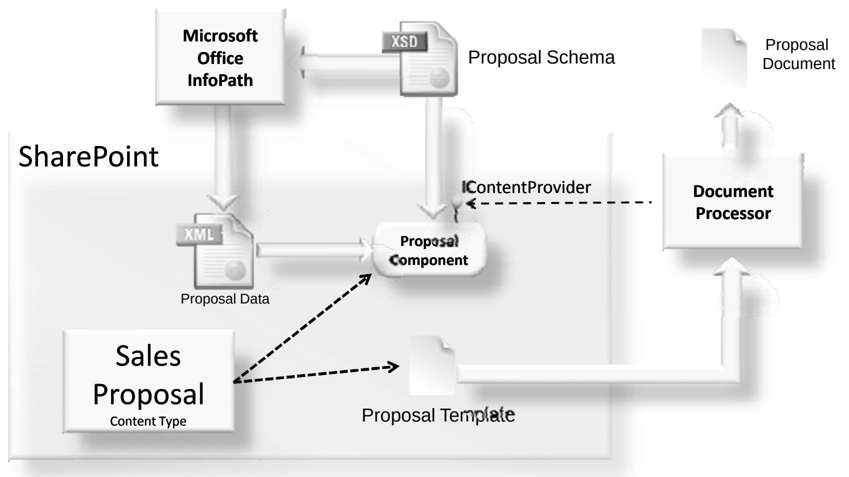
Document Processing Framework
Merging SharePoint List Data



SharePoint Document Generation

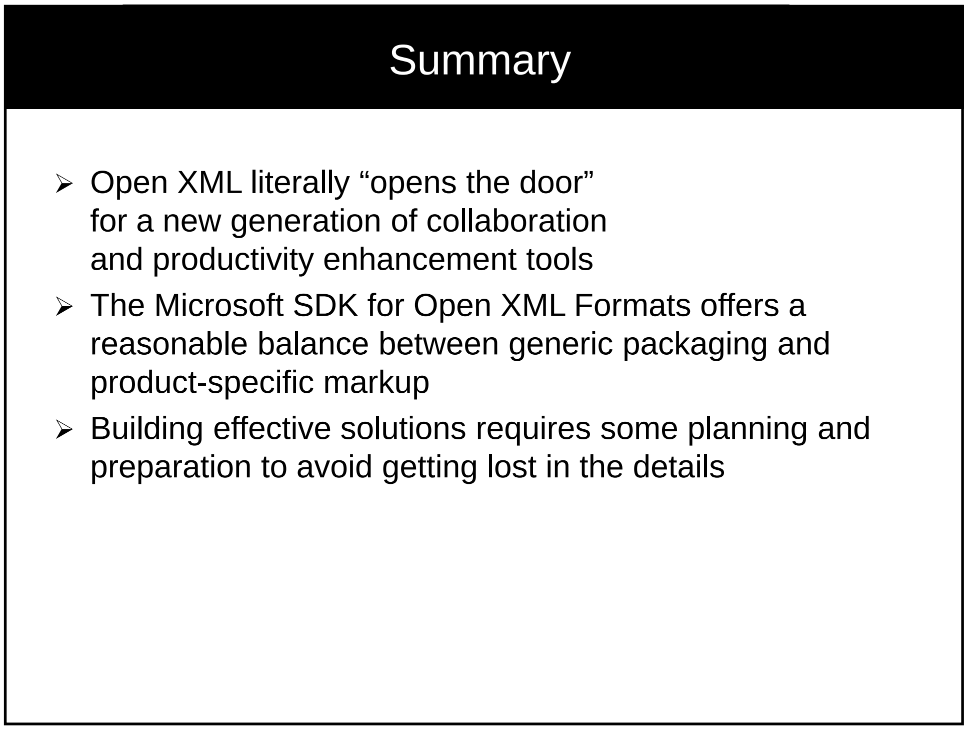
Proposal Builder

Document Processing Framework
Leveraging InfoPath and XML Schemas



SharePoint + Word + InfoPath

Proposal Generation Solution



Summary

- Open XML literally “opens the door” for a new generation of collaboration and productivity enhancement tools
- The Microsoft SDK for Open XML Formats offers a reasonable balance between generic packaging and product-specific markup
- Building effective solutions requires some planning and preparation to avoid getting lost in the details