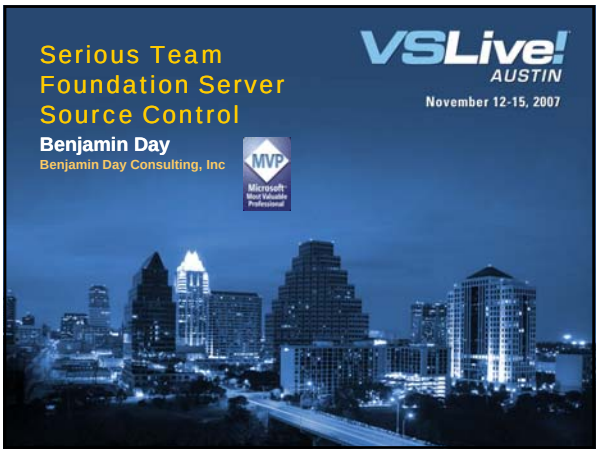


About the speaker

- Owner, Benjamin Day Consulting, Inc.
  - Email: benday@benday.com
  - Web: http://www.benday.com
  - Blog: http://blog.benday.com
- Trainer
  - Visual Studio Team System, Team Foundation Server
- Microsoft MVP for C#
- Microsoft VSTS/TFS Customer Advisory Council
- Leader of Beantown.NET INETA User Group

What am I going to cover?

- Source Control configuration options
- Source Control via command line
- Custom Check-in Policies

What is Team Foundation Server?

- "Glue" that connects your team
  - What am I supposed to do?
  - What's my progress?
  - What's broken?
- Helps manage project artifacts
  - Source Control

Why is it good?

- Helps your team communicate
- Centralized place to manage your project
- Data is captured automatically
- Everyone looks at the same data
- Customizable
- Real version/source control

TFS Source Control

- Real, enterprise-quality source control
- Uses SQL Server 2005 as the repository
- Transactional, atomic

Speaker Name

Why use source control?

- Minimize / eliminate lost work
- Reproducible builds & product state

TFS Source Control: Features

- Workspaces
  - Area on local disk where you edit files
- Check in / check out
  - Check out marks the beginning of your edits
  - Check in commits your changes to the repository
    TFS allows shared check out
- Changesets
  - Group of changes that happen when you check in
- Shelving
  - Similar to check in
  - Changes get stored on the server
  - Not visible as part of the main project source tree
- Branching
  - Used to manage multiple versions of a product

Modifications & Check in's

- All these operations are batched
  - Add, Delete
  - Moves, Renames
  - Modifications
  - Branches / Merges
- Batch is "sent" at check in
- Check in is atomic → Changeset

TFS Source Control: Support beyond VS2005

- Team Foundation Server MSSCCI Provider
  - Available for download from microsoft.com
- Source control only
- Supports:
  - Visual Studio .NET 2003
  - Visual C++ 6 SP6
  - Visual Visual Basic 6 SP6
  - Visual FoxPro 9 SP1
  - Microsoft Access 2003 SP2
  - SQL Server Management Studio
  - Sparx Systems Enterprise Architect 6.0
  - Sybase PowerBuilder 10.5

Overview of Source Control Settings

- Demo
- Check out settings
- Check in policy settings
- Check in note settings

What should go into source control?

- Solution files -- *.sln
- Project files -- *.csproj, *.vbproj
- Source Control Project Metadata (*.vspscc)
  - Project bindings
  - Source control configuration
- Application config files (*.config)
- Source files
- Binary dependency references

Speaker Name

What should not go in source control?

- Solution user option files (*.suo)
  - Local user customizations
- Project user option files (*.user)
  - Local user customizations
- WebInfo files (*.webinfo)
- Build outputs
  - /bin/debug
  - /bin/release
  - /obj

Best Practices

- http://www.codeplex.com/TFSGuide
- Lots of great info
- Beta 1

Best Practice: Client-side vs Server-side Structure, Part 1

- Directory structure should be the same
  - Client-side structure should match server
- Simplifies "Get Latest"
- Everyone has the same directory layout
- This doesn't mean that everyone has to has to put their source in the same place on disk
  - The relative paths should match
  - Root path of source tree can be different

Best Practice: Client-side vs Server-side Structure, Part 2

C:\DevProjects	→ Root container folder for all team projects
\MyTeamProject1	→ Container folder for TeamProject1
\MyTeamProject2	→ Container folder for TeamProject2

TFS does more than just check-in and check-out

- Branching and merging
  - Facilitates simultaneous development of multiple versions of an app

Best Practice: Repository Setup For Branching & Merging

- Do not add sources directly to the root of your Team Project source control tree
  - $/*My Team Project*/Main
  - $/*My Team Project*/Branch
  - $/*My Team Project*/MyApp1/Main
  - $/*My Team Project*/MyApp1/Branch
- Branch cannot be located under the branch origin
  - Error
  - Recursive
- Do this even if you don't currently need branching!

Best Practice: Don't Branch

- Don't branch without a good reason
- Life only gets more complex with branching
- Favor Labels over Branches
- You can always branch from a Label later

Signs you might need to branch

- Regular broken builds
- Features in parallel development that need to edit one another
  - Branch so that each feature can develop in isolation
  - Merge changes later
- Ask yourself if the productivity gained by the branch is balanced against the pain (anti-productivity) of merging changes back

Branching Scenarios

- No Branching
  - Everyone works from the same code
- Branch for Release
  - Stabilization in order to prepare for a release
- Branch for Maintenance
  - Maintenance of a previous build/release
  - Service pack development
- Branch for Feature
  - Branch from the Trunk to develop a new feature
  - Merge back into the Trunk when feature is done
- Branch for Team
  - Branches for a team working on a set of features

Branching For Isolation: The Process

Figure 5.3 shows a typical timeline when branching for development isolation.

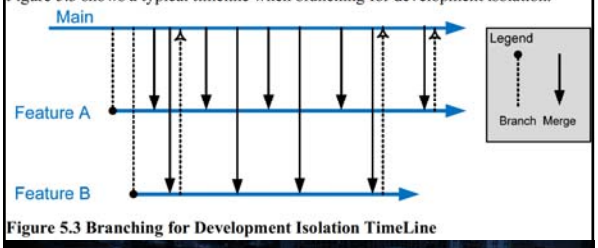


Figure 5.3 Branching for Development Isolation TimeLine

Branching & Merging

- You can specify 3rd-party merge tools
  - Tools → Options → Source Control → Visual Studio Team Foundation → File Extensions… → Add… → Configure Tool

Demo

- Set up the repository
- Add some code
- Create/resolve a check-in conflict
- Create a branch
- Merge changes

Source Control

- Bad news
  - Some features are command-line only
- Good news
  - There's a lot you can do from the command-line
  - Scriptable

TF.exe

- Command-line interface to TFS source control
- 30+ sub commands
  - Kind of like "net" command in Windows
- Why would you want to use the command line version?
  1. It's cool
  2. Some things aren't available through the UI
  3. Good for automated operations (builds, etc)

The Commands

- Add
- Branch / Branches
- Changeset
- Checkin / Checkout
- Configure
- Delete / Undelete
- Dir
- Get
- History
- Label / Labels / Unlabel
- Lock
- Merge / Merges
- Move
- Permission
- Properties
- Rename
- Resolve
- Shelve / Unshelve
- Status
- Undo
- View
- Workfold
- Workspace / Workspaces

Things you (probably) can't do through the GUI

- Find files in TFS by name/wildcard
  - tf dir
- Get particular version of a file(s) by wildcard
  - tf get
- Find checked out / pending change files
  - tf status
- What will be changed by a "get latest"?
  - tf get /preview
- Baseless merges
  - tf merge /baseless
- Eliminating a changeset from a merge
  - tf merge /discard

tf get

- Gets file(s) from server to workspace
- Args
  - /version
  - /all – forces get all files
  - /overwrite – replace read-only files that aren't checked out
  - /force – equivalent of /all + /overwrite
  - /preview – show what would happen but don't do it
  - /recursive
  - /noprompt – no visual dialog boxes
- **Example:** get everything for this workspace for changeset #29
  - tf get * /all /version:c29 /recursive /force

The /version option

- Date
  - /version:D*10/11/2001*
- Changeset #
  - /version:C*1234*
- Label
  - /version:L*labeltext*
- Latest version
  - /version:T
- Workspace Name
  - /version:W*workspacename*

Team Foundation Server Power Tool

- Free download from Microsoft
- tfpt.exe
- Add "C:\Program Files\Microsoft Team Foundation Server Power Tools" to your PATH environment variable

TFPT Commands

- Annotate
- GetCS
- History
- Online
- Query
- Review
- Rollback
- Treeclean
- Treediff
- Unshelve
- UU
- Workitem
- Workspace

Source Control Security

- Team Explorer → Source Control Explorer → *Folder* → Properties…
- Permissions:
  - Read
  - Check Out
  - Check In
  - Label
  - Lock
  - Revise other users' changes
  - Undo other users' changes
  - Administer labels
  - Manipulate security settings
  - Check in other users' changes

Customizing Version Control

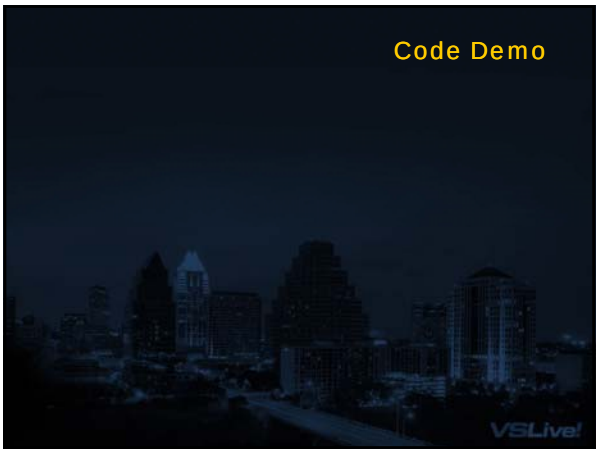
- Create a custom check-in policy
- Extend PolicyBase
  - Microsoft.TeamFoundation.VersionControl.Client.dll
- Mark class as [Serializable]
- PolicyBase.Evaluate() lets you examine
  - What's being checked in
  - Associated work items
  - Check-in comments
  - Other check-in policies

Installing the Check-in Policy

- Compile
- Copy to the server
- Go to HKLM\SOFTWARE\Microsoft\VisualStudio\8.0\TeamFoundation\SourceControl\Checkin Policies
- Add new "string value"
  - Value name must be the same as the DLL name (minus ".dll")
  - Data is the full path to the DLL

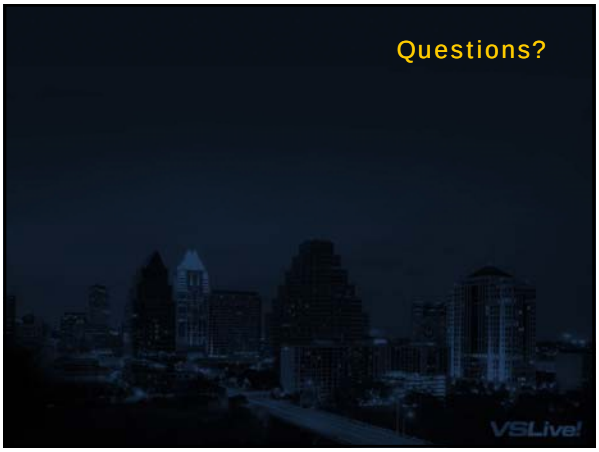
Policy Gotcha!

- Policies are evaluated on the client
- Policy DLL must be installed on every developer's computer
- Server-side policy configs are stored using binary serialization
  - Everyone must have the same version of the policy DLL



Extra Demo: Enforce Coding Standards using Custom SCA Rule

- Create rule
- Register rule by copying it to C:\Program Files\ Microsoft Visual Studio 8\Team Tools\ Static Analysis Tools\FxCop\Rules



About the speaker

- Owner, Benjamin Day Consulting, Inc.
  - Email: benday@benday.com
  - Web: http://www.benday.com
  - Blog: http://blog.benday.com
- Trainer
  - Visual Studio Team System, Team Foundation Server
- Microsoft MVP for C#
- Microsoft VSTS/TFS Customer Advisory Council
- Leader of Beantown.NET INETA User Group

VSLive!

Speaker Name

VSLive! BOSTON