

## Real-World ASP.NET Workflow Applications



**Paul Mehner**  
 paulm@wintellect.com  
 Copyright © 2006, 2007  
 www.wintellect.com

### Goals

- Identify practical developer knowledge necessary to build real-world state-machine workflow applications hosted in ASP.NET
- Create a centralized reference document & code sample
- Learn tips and tricks often left out of training material

### Build Out Your Runtime Environment

- a) Database layout (who's living with who)
- b) Workflow Runtime Wrapper
- c) Hosting the WF Runtime in ASP.NET
- d) Instance Scheduling (default or manual)
- e) Instance Passivation (i.e. Persistence)
- f) Instance Tracking
- g) Exception handling
- h) Workflow instance suspension & recovery
- i) WF Runtime Configuration
  - a) Connection Strings
  - b) Logging & Tracing

### Build Out Your Runtime Environment (cont)

- j) Debugging
- k) Transactions
- l) Security

### a) Database Layout



**Workflow Meets Application Data**

### Database Configurations—App DB Only

- Pros
  - No extraneous databases or data
  - Improved performance
- Cons
  - Workflow instances consume memory; server memory must be adequate because no passivation is possible.
  - Suspended workflows have no place to persist, so they stay in memory too.
  - Not scalable; workflows cannot be transferred to other machines

### Database Configurations—Separate DB

- Pros
  - It's Microsoft's preferred deployment scenario
  - Separation of application data from the infrastructural data
  - Greater scalability (adding more database machines)
  - Workflow runtime can potentially service multiple applications
- Cons
  - Referential integrity can be easily violated
  - Elevation of database transaction to use the DTC (reducing performance)

### Database Configurations—All in one

- Pros
  - Referential integrity of workflow to application data is easy to maintain
  - Data is more readily available for querying and reporting without setting up cross-database security & views
  - DTC not required for transaction
- Cons
  - Less scalability
  - Implies that WF runtime is servicing only one application
  - Microsoft's tables and sprocs don't follow any apparent naming convention and it may be difficult to tell infrastructural assets from the application data and stored procedures

### Install SQL Persistence & Tracking

- %WinDir%\Microsoft.NET\Framework\v3.0\Windows Workflow Foundation\SQL\EN
  - SqlPersistenceService\_Schema.sql
  - SqlPersistenceService\_Logic.sql
  - Tracking\_Schema.sql
  - Tracking\_Logic.sql
- Run against application database, or create separate persistence and tracking databases

### Create An Associative Table

- Create an associative table to bind the primary key of your business domain with a corresponding workflow identifier

Column Name	Data Type	Allow Nulls
WorkflowInstanceID	uniqueidentifier	<input type="checkbox"/>
StartDate	datetime	<input type="checkbox"/>
DomainTypeID	int	<input type="checkbox"/>
DomainKeyID	int	<input type="checkbox"/>
DomainWorkflowTypeID	int	<input type="checkbox"/>
PercentageCompleted	int	<input type="checkbox"/>
WorkflowStatusText	varchar(1000)	<input checked="" type="checkbox"/>
UpdatedBy	int	<input type="checkbox"/>
LastUpdated	datetime	<input type="checkbox"/>
TStamp	timestamp	<input type="checkbox"/>

### Demo—Database Layout

### WF Runtime Wrapper



## Build A WF Runtime Wrapper

- Benefits of Having a Runtime Wrapper
  - Loose coupling of UI to workflow Foundation
  - Coordinate of WF runtime events with UI
  - Facilitate handled exceptions returning messages to UI
  - Flexible custom configurations
  - Explicit initialization control

## Demo—Code Walkthrough Runtime Wrapper

## WF Runtime Hosting



## Hosting the WF Runtime

- Create a singleton instance of the WF Runtime in the Start\_Application event of global.asax and place it in Application
- Add event handlers for runtime events
- Perform additional logging in event handlers (as desired)

## Demo—Hosting the WF Runtime

## Scheduling



### DefaultWorkflowSchedulerService

```
<add
  type="System.Workflow.Runtime.Hosting.DefaultWorkflowSchedulerService, System.Workflow.Runtime, Version=3.0.00000.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
```

```
instance.Start();
```

### ManualWorkflowSchedulerService

```
<add
  type="System.Workflow.Runtime.Hosting.ManualWorkflowSchedulerService, System.Workflow.Runtime, ...
  PublicKeyToken=31bf3856ad364e35" />
```

```
ManualWorkflowSchedulerService manualScheduler =
  runtime.GetService<ManualWorkflowSchedulerService>
  ();
if (manualScheduler == null) { throw new
  WorkflowRuntimeException(""); }
instance.Start();
bool result = manualScheduler.RunWorkflow(instanceID);
```

### Don't Mix Your Schedulers

- Adding a ManualWorkflowSchedulerService will automatically remove the DefaultWorkflowSchedulerService
- Instances of workflow can be created and you can legally call the Start() method on them; however...
- A thread will never be dispatched to service them
- When the ManualWorkflowSchedulerService has been configured you must donate your thread to run the workflow using manualScheduler.RunWorkflow(instanceID)

### Demo—Workflow Instance Scheduling

## Passivation



### Passivation in Web Development

- Without Persistence Store
  - Instance lost when visual studio stops execution
  - Instance lost when exception occurs
- Workflow instance data is part of your application's data
- Web is a stateless environment; applications have state
- Passivation provides a mechanism for keeping workflow and application data synchronized in a stateless environment

**Demo—Passivation In Action**

# Tracking



**Workflow Tracking**

# Exceptions



## Exceptions

- Workflows are compiled into Intermediate Language
- Activities can be thought of as blocks of executable code
- It's bad to catch and swallow exceptions in code
- It's bad to catch and swallow exceptions in workflows
- The runtime catches and swallows exceptions
- You should strive to avoid unhandled exceptions

## Exception Strategies

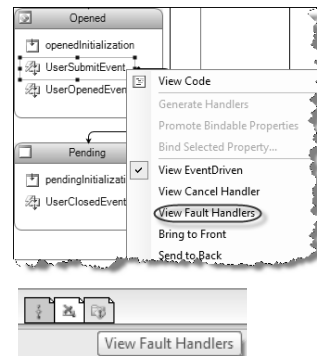
	Do	Explain
1	Diligently and defensively prevent them from ever becoming unhandled	Check parameters before use. Use assertions, unit tests, etc.
2	Turn non-designed conditions into early runtime failures	Throw plenty of exceptions (while keeping rule 1 in mind)
3	Catch specific exceptions	Don't be a lazy slug! You should never catch System.Exception in a workflow!

### You Blew It! – Unhandled Exceptions

- Caught by the Runtime and raised as a WorkflowTerminated event
- Exception exposed as property of EventArgs
- Unhandled Exceptions are very bad—but arguably worse in this style of environment

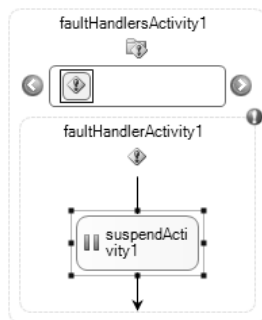
### Exception Handling in Workflows

- On Context Menu of State Machine Workflow
- Lower left corner of Sequential Workflow



### Exception (Fault) Handler

- Individual Fault Handlers are just Sequential Workflows



### Demo-Fault Handlers

### Exceptions In Local Services

- Ignore the workflow defined exception handler
- Must be handled (if appropriate) by the workflow host
- Unfortunately, the workflow host generally does not have any idea about the context of the exception
- Recommendation: consider using custom activities instead of local services to interact with external data

### Use Custom Activities Instead of LocalService Methods

## Suspensions

- Suspensions are used to preserve an in flight workflow when an exception occurs

## Resumptions

## Configuration



## Configuration

- Connection Strings
- Tracing & Logging
- Hosting
- External Services

## Configuration—Tracing & Logging

```
<system.diagnostics>
  <switches>
    <add name="System.Workflow.LogToTraceListeners" value="1" />
    <add name="System.Workflow.Activities.Rules" value="All" />
    <add name="System.Workflow.Runtime" value="Warning" />
    <add name="System.Workflow.Runtime.Tracking" value="Warning" />
    <add name="System.Workflow.Activities" value="Error" />
  </switches>
  <trace autoflush="true" indentsize="4">
    <listeners>
      <add name="TextWriterTraceListener"
        type="System.Diagnostics.TextWriterTraceListener"
        initializeData="TextWriterOutput.log" />
    </listeners>
  </trace>
</system.diagnostics>
```

## Configuration—Hosting

```
<hostingWorkflowRuntime Name="Hosting">
  <CommonParameters>
    <add name="ConnectionString" value="YourConnectionString" />
    <add name="EnableRetries" value="True" />
  </CommonParameters>
  <Services>
    <add type="S.W.R.H.SqlWorkflowPersistenceService, ..."
      unloadOnIdle="true" />
    <add type="S.W.R.H.ManualWorkflowSchedulerService, ..."
      useActiveTimers="true" />
    <add type="S.W.A.ExternalDataExchangeService, ..." />
    <!-- add type="S.W.R.T.SqlTrackingService, ..." /-->
    <!-- add
      type="S.W.R.H.SharedConnectionWorkflowCommitWorkBatchService, ..." /--
  >
</Services>
</hostingWorkflowRuntime>
```

## Configuration—External Services

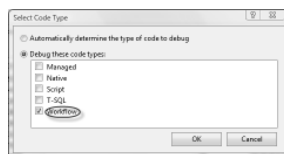
```
<hostingWorkflowRuntime Name="Hosting">
  <CommonParameters>
    <add name="ConnectionString" value="YourConnectionString" />
    <add name="EnableRetries" value="True" />
  </CommonParameters>
  <Services>
    <add type="S.W.R.H.SqlWorkflowPersistenceService, ..."
      unloadOnIdle="true" />
    <add type="S.W.R.H.ManualWorkflowSchedulerService, ..."
      useActiveTimers="true" />
    <add type="YourServiceType, YourServiceStrong Name" />
  </Services>
</hostingWorkflowRuntime>
```

## Debugging



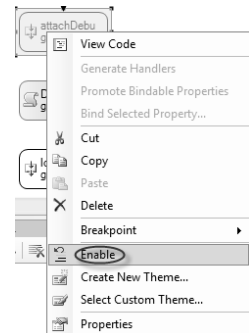
## Attaching Debugger to Worker Process

- Managed Code and Workflow debug settings
- Attaching to a w3wp (or aspnet\_wp) process with "Automatic" or "Managed" after the workflow runtime has been loaded will crash visual studio!
- Not applicable to WebDev



## Enable/Disable Activities

- From the context menu in the workflow designer select Enable/Disable on individual activities



## NUnit & The Workflow Debugger

- Show impact of Workflow vs. Managed setting when attaching a debugger through an NUnit test harness.

## Transactions



## Transactions

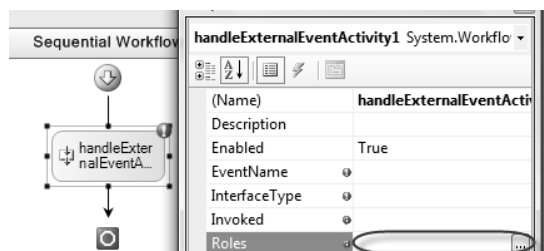
- Workflow and Application Data must be kept in sync.
- Use the TransactionScope activity to enlist in the ambient transaction
- Add SharedConnectionWorkflowCommitWorkBatchService

## Security

**Wintellect**<sup>®</sup>  
Know how.

## Workflow Roles

- Impact of Roles on Workflow
- Resulting Behavior



## Versioning

**Wintellect**<sup>®</sup>  
Know how.

## Versioning Custom Activities

- Group related activities into component libraries
- Version libraries when changes are made & distribute
- Use GAC for side-by-side versioning
- Optionally, use unique assembly names for versions
- Lie about your version number (but at your own risk)

## Workflow Relationships To Rules

- Potentially tight coupling between workflows and rules

### Store Interfaces In Separate Assembly

- Promotes separation of responsibilities
- Reduces assembly versioning dependencies
- Allows implementation to vary without impacting workflows
- Allows parallel team development

### Potential Versioning Strategies

- Version workflows and rules together
- Separate out tight dependencies from loose ones

### Changing Namespaces, etc.

## Local Services



### Local Services

- Abstraction to simplify the underlying messaging infrastructure of WF
- Exposes access to the message queues using an easy-to-use and understand event model
- Abstraction developed to support SharePoint 2007 integration
- Exposes capabilities of the host to the workflow

## Discussion

