ArcGIS Pro SDK for .NET: An Overview of the Geodatabase API

Colin Zwicker
Ling Zhang
Nghiep Quang
What will not be deeply discussed

- Add-in model & threading model
  - ArcGIS Pro SDK for .NET: Beginning Pro Customization and Extensibility (Tues 2:30 pm)
  - ArcGIS Pro SDK for .NET: Advanced Pro Customization and Extensibility (Wed 2:30 pm)

- Editing
  - ArcGIS Pro SDK for .NET: Beginning Editing and Editing UI Patterns (Tues 5:30 pm)
  - ArcGIS Pro SDK for .NET: Advanced Editing and Edit Operations (Wed 4 pm)

- Utility Network
  - ArcGIS Pro SDK for .NET: An Overview of the Utility Network Management API (Fri 8:30 am)
Intro to Core.Data

- Geodatabase API namespace Core.Data

- ArcObjects is not exposed to external developers
- Managed .NET API
- Coarse grained objects (layer, CIM model, etc.)

- Add-In extensibility mechanism for external developers
- Looser coupling at the application level than with ArcObjects Class Extensions
Grows on demand to support ArcGIS Pro

- 1.1 – web geodatabase support level (fileGDB, Enterprise)
- 1.2 – versioning, and feature service support
- 1.3 – database access, queryDefinitions
- 1.4 – joins, SQLSyntax, feature service as geodatabase
- 2.0 – Datastore update, Raster, Blob fields, \texttt{datastore.GetPath}
- 2.1 – Annotation, datastore.GetPath, table.GetCount & GetID
  - Versioning: Alter\Create\Delete version, reconcile, conflicts
  - Versioned feature service datastores
Architecture Principles
Architecture Principles

- Managed .NET API that provides access to the Geodatabase and supporting data
- It is an object-oriented API
- Aligns with modern C# practices and existing frameworks
- Adheres to the principles and architecture of the general Pro API
Core.Data API is a DML-only (Data Manipulation Language) API

- Cannot perform schema creation or modification operations:
  - creating tables or adding fields
  - creating domains or adding coded values
- Schema operations are performed using the GP (Geoprocessing) tools
- GP tools can be called from C# using the Geoprocessing API

```csharp
IReadOnlyList<string> args = Geoprocessing.MakeValueArray("C:\Data.gdb", "Places");
IGPResult gpResult = await Geoprocessing.ExecuteToolAsync("CreateFeatureclass_management", args);

if (gpResult.IsSuccess) {
    // ...
}
```
Threading

- Almost all of the methods in Core.Data API should be called on the MTC (Main CIM Thread)
  - API reference documentation on the methods that need to run on the MCT are specified
  - These methods calls should be wrapped inside the QueuedTask.Run call
  - Failure to do so will result in CalledOnWrongThreadException or ConstructedOnWrongThreadException being thrown
- Read “Working with multithreading in ArcGIS Pro” conceptual help to learn more

```csharp
await QueuedTask.Run(() =>
{
    using (Geodatabase geodatabase = new Geodatabase(new DatabaseConnectionFile(new Uri(sdefFilePath))))
    {
        workspaceConnectionString = geodatabase.GetConnectionString();
        // ...
    }
});
```
Core.Data
Datastore

- Container of spatial and non-spatial datasets
- Accessed by:
  - connection properties
  - uri (local file path or url)
  - ArcGIS.Core.Data.Dataset.GetDatastore()
Datastore constructors
Datasets - Table

- **Table**
  - Contains 0 or more Rows
  - Supports Search and Select

- `geodatabase.OpenDataset<Core.Data.Table>("TableName");`
- `Desktop.Mapping.FeatureLayer.GetTable();`
- `Row.GetTable();`
Datasets - Feature class

- Feature class
  - Inherit from tables
  - Tables with shape (point, line, polygon)
  - Contains 0 or more Features
  - Supports spatial queries and Selections

```csharp
geodatabase.OpenDataset<Core.Data.FeatureClass>("FeatureClassName");
If (table is FeatureClass) Desktop.Mapping.FeatureLayer.GetTable() as FeatureClass;
If (row is Feature) Row.GetTable() as FeatureClass;
Open as table:
  geodatabase.OpenDataset<Core.Data.Table>("FeatureClassName");
```
Datasets - Feature dataset

- **Feature Dataset**
  - Collection of related datasets with a common coordinate system
  - Organize feature classes to support topology, network dataset or terrain dataset.
  - Acts as a container

- `geodatabase.OpenDataset<Core.Data.FeatureDataset>("FeatureDatasetName")`
Datasets - Relationships

- Geodatabase stored relationship between tables\feature classes
- Origin and Destination tables
- Cardinality of relationship between features

- Two types:
  - RelationshipClass
    - May not have a backing table
  - AttributedRelationshipClass
    - Inherits from RelationshipClass
    - Has a backing table
    - May have user defined attributes
Definition

- Concept to represent information about the dataset
- Used to describe schema and unique properties
  - e.g., GetFields, HasGlobalID

- Separated from the dataset to facilitate a lightweight mechanism of discovery
- Opening dataset comparatively expensive
- Definitions can be use to filter datasets without opening them

- table.GetDefinition();
- geodatabase.GetDefinition<TableDefinition>("TableName");
- geodatabase.GetDefinitions<TableDefinition>();
Demo: Datastore & Dataset
Querying Data

• **Selection**
  - List of Object IDs
  - Lightweight way to highlight features on map
  - Ability to combine

• **Search**
  - Return Rows (or subset of values) via RowCursor
  - Table bound
  - Supports Recycling
Querying Data - Filtering queries

- QueryFilter
  - Used to restrict the records retrieved (Whereclause, Subfields)
  - Apply pre or postfix clauses (DISTINCT or ORDER BY)
  - Exception to MTC rule (can be created on a non-MTC thread)
Querying Data - QueryDef

- **QueryDef**
  - Available from the geodatabase datastore (exception, not available with Feature Services)
  - Tables are input parameter
  - Single table query
  - Or two or more joined tables within the same datastore
  - Rows do not implement GetTable()
  - Does not support field aliases
  - ‘Left’ most shape field supported

- `geodatabase.Evaluate(queryDef, false);`
Querying Data - QueryTable

- **QueryTable**
  - **ReadOnly**
  - Virtual table (for map layer or Geoprocessing)
  - Requires a QueryTableDescription
    - Name
    - QueryDef
    - Key fields (Object ID \ Shape)

- `geodatabase.OpenQueryTable(QueryTableDescription);`
Demo:
Working with rows & features
SQL Helper Method

- **SQLSyntax**
  - Eases the generation of SQL across multiple platforms
  - FileGDB, Oracle or SQLServer
  - Obtained from the datastore
  - Fully qualify a table or field
  - Return owner name for a fully qualified table name
  - Find supported sql functions
  - Find keywords
  - Useful when creating fields or table via geoprocessing
    - Invalid characters
Joins

- Used to combine fields from two tables into a single table representation
- Supports table from one or two datastores
- Created using relationship classes
  - Relationship class stored in the geodatabase
  - Or Virtual relationship class
- Result is read-only, but reflect updates to underlying tables
- Do not support Many-Many cardinality
  - VirtualRelationshipClassDescription will raise an exception
Joins & Relates

- `join.GetJoinedTable()`
- Definitions not supported on joined tables where tables are from multiple datastores
- Every attempt will be made to push join to database for performance
  - Within a single datastore

- Virtual relationship class can be used to support a Relate
  - Created with `VirtualRelationshipClassDescription`
- `table.RelateTo(rightTable, virtualRelationshipClassDescription)`
**Database and QueryLayers**

- **Database**: Datastore for a DBMS without geodatabase enabled
- Allows queries of these tables via QueryLayers

- QueryLayers are created via `Database.GetQueryDescription`
  - From a table
  - From a string containing a SQL query

- Modify `QueryDescription`
  - unique ID field, must have not Nulls
  - single geometry column

- Pass to `Database.OpenTable(QueryDescription)`
When to use difference query constructs

- Simple single registered table from geodatabase? -> table.Search or fc.Search
- Onetime query with join within a geodatabase? -> QueryDef
  - Don’t like sql? Join
- Reuse above query multiple times? -> QueryTable
- Use result as Layer or input to GP tool? -> QueryTable
- Join multiple tables in the same datastore? -> QueryDef
  - Join does support joining a table to an existing JoinTable
- Database simple query? -> QueryLayer
- Database join? -> QueryLayer
- Cross datastore join? -> Only possible via Join
Demo: Joins
Versioning Support

- **VersionManager**
  - Available only if `Geodatabase.IsVersioningSupported`
  - Access to HistoricVersions or public owned versions

- **Current capabilities**
  - Create Versions
  - Connect to a specific named version
  - List all versions in a geodatabase, including properties
  - List differences between Tables and Feature Classes from different Versions
Versioning Support

- Version
  - Alter or Delete
  - Reconcile
  - Refresh
Editing

• ArcGIS.Desktop.Editing
  - EditOperation

• ArcGIS Pro SDK for .NET: Beginning Editing and Editing UI Patterns
  - Tuesday 5:30 pm
  - Mojave Learning Center

• ArcGIS Pro SDK for .NET: Advanced Editing and Edit Operations
  - Wednesday 4:00 pm
  - Santa Rosa
Core.Host

- Using CoreHost in ArcGIS.Core.Hosting namespace
  - Programs using the API can be run “head-less” (i.e., standalone)
  - GP tools must be run directly from Python

- Requirements
  - References (ArcGIS.Core & ArcGIS.CoreHost)
  - 64 bit app with [STAThread]
  - Host.Initialize()
  - Licensing controlled by Pro application

- ArcGIS.Desktop.Editing not possible
  - geodatabase.ApplyEdits
    - Not to be used in Add-in
Best Practices
Garbage Collection

- By default, locks and connections on database objects held by .NET objects are released at non-deterministic times (When garbage collection runs)
  - As a consequence, connections and locks on any dataset referenced by the .NET object will also be held
- Using blocks or calling Dispose on objects after use provides greater control on when connections and file system locks are released

```csharp
FeatureClass featureClassIndeterministic = geodatabase.OpenDataset<FeatureClass>(featureClassName);
using (FeatureClass featureClassDeterministic = geodatabase.OpenDataset<FeatureClass>(featureClassName))
{
    Console.WriteLine(featureClassDeterministic.GetName());
}
```

- Locks acquired by featureClassIndeterministic are released when garbage collector runs
- Locks acquired by featureClassDeterministic are released at the end of the using block
Best Practices

- Do explicitly dispose of objects of the following types:
  - `ArcGIS.Core.Data.Datastore` and its derived classes (e.g., `Geodatabase`)
  - `ArcGIS.Core.Data.Dataset` and its derived classes (e.g., `FeatureDataset`, `Table`, `FeatureClass`, `RelationshipClass`, etc)
  - `ArcGIS.Core.Data.RowCursor`
  - `ArcGIS.Core.Data.RowBuffer`
  - `ArcGIS.Core.Data.Row` and its derived class, `Feature`
  - `ArcGIS.Core.Data.Selection`
  - `ArcGIS.Core.Data.VersionManager`
  - `ArcGIS.Core.Data.Version`
Best Practices (cont’d)

- Two standard ways to explicitly dispose of objects in C#
  - Via the *try/finally* block:

```csharp
public void DoSomething()
{
    Geodatabase geodatabase = null;
    FeatureClass featureClass = null;

    try
    {
        geodatabase = new Geodatabase(connectionProperties);
        featureClass = geodatabase.OpenDataset<FeatureClass>(SOME_FEATURE_CLASS);
        //...
    }
    finally
    {
        if (featureClass != null)
        {
            featureClass.Dispose();
        }

        if (geodatabase != null)
        {
            geodatabase.Dispose();
        }
    }
}
```
Best Practices (cont’d)

- Via the *using* statement:

```csharp
public void DoSomething()
{
    using (Geodatabase geodatabase = new Geodatabase(connectionProperties))
    using (FeatureClass featureClass = geodatabase.OpenDataset<FeatureClass>(SOME_FEATURE_CLASS))
    {
        //...
    }
}
```
Best Practices (cont’d)

- Do remember to explicitly dispose of Row/Feature returned from RowCursor
  - Not explicitly disposing of Row/Feature:

```java
public static long CountRows(Table table) {
    using (RowCursor rowCursor = table.Search()) {
        long count = 0;

        while (rowCursor.MoveNext()) {
            ++count;
        }

        return count;
    }
}
```
Explicitly disposing of Row/Feature:

```csharp
public static long CountRows(Table table)
{
    using (RowCursor rowCursor = table.Search())
    {
        long count = 0;

        while (rowCursor.MoveNext())
        {
            ++count;
            rowCursor.Current.Dispose(); // Need to call Dispose on the Row/Feature as well.
        }

        return count;
    }
}
```
• Canonical form of processing a `RowCursor`

```csharp
public void DoSomething()
{
    using (Geodatabase geodatabase = new Geodatabase(connectionProperties))
    using (Table table = geodatabase.OpenDataset<FeatureClass>(SOME_TABLE))
    {
        using (RowCursor rowCursor = table.Search())
        {
            while (rowCursor.MoveNext())
            {
                using (Row row = rowCursor.Current)
                {
                    //...
                }
            }
        }
    }
}
```
Best Practices (cont’d)

- Don’t create static variables for classes derived from ArcGIS.Core.CoreObjectBase

```csharp
public class SomeClass
{
    private static Table table = Factory.InitializeTable();
    //...
}
```

- will cause `ConstructedOnWrongThreadException` because all static variables will be executed on the **GUI thread** when an add-in is first loaded by ArcGIS Pro
Road Ahead
Near to Long Term

- Table Sort
- Data statistics (Min, Max, Sum, …)
- ArchiveClass access
- Data Changes (edits)
- Data Differences (versions \ historical)
- Topology feature interactions
- Dimensions support
Core.Data - Conceptual Help

- https://github.com/Esri/arcgis-pro-sdk/wiki

- Geodatabase
  - ProSnippets: Geodatabase
  - ProConcepts: Geodatabase

- Editing
  - ProSnippets: Editing
  - ProConcepts: Editing
# ArcGIS Pro SDK for .NET Tech Sessions

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Technical Session</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue, Mar 06</td>
<td>1:00 pm - 2:00 pm</td>
<td>An Overview of the Geodatabase API</td>
<td>Mojave Learning Center</td>
</tr>
<tr>
<td></td>
<td>2:30 pm - 3:30 pm</td>
<td>Beginning Pro Customization and Extensibility</td>
<td>Primrose A</td>
</tr>
<tr>
<td></td>
<td>5:30 pm - 6:30 pm</td>
<td>Beginning Editing and Editing UI Patterns</td>
<td>Mojave Learning Center</td>
</tr>
<tr>
<td>Wed, Mar 07</td>
<td>10:30 am - 11:30 am</td>
<td>Mapping and Layout</td>
<td>Pasadena/Sierra/Ventura</td>
</tr>
<tr>
<td></td>
<td>1:00 pm - 2:00 pm</td>
<td>Advanced Pro Customization and Extensibility</td>
<td>Santa Rosa</td>
</tr>
<tr>
<td></td>
<td>2:30 pm - 3:30 pm</td>
<td>Pro Application Architecture Overview &amp; API Patterns</td>
<td>Mesquite G-H</td>
</tr>
<tr>
<td></td>
<td>4:00 pm - 5:00 pm</td>
<td>Advanced Editing and Edit Operations</td>
<td>Santa Rosa</td>
</tr>
<tr>
<td>Thu, Mar 08</td>
<td>9:00 am - 3:30 pm</td>
<td>Getting Started Hands-On Training Workshop</td>
<td>Mojave Learning Center</td>
</tr>
<tr>
<td></td>
<td>5:30 pm - 6:30 pm</td>
<td>Working with Rasters and Imagery</td>
<td>Santa Rosa</td>
</tr>
<tr>
<td>Fri, Mar 09</td>
<td>8:30 am - 9:30 am</td>
<td>An Overview of the Utility Network Management API</td>
<td>Mesquite G-H</td>
</tr>
<tr>
<td></td>
<td>10:00 am - 11:00 am</td>
<td>Beginning Pro Customization and Extensibility</td>
<td>Primrose A</td>
</tr>
<tr>
<td></td>
<td>1:00 pm - 2:00 pm</td>
<td>Advanced Pro Customization and Extensibility</td>
<td>Mesquite G-H</td>
</tr>
</tbody>
</table>
## ArcGIS Pro SDK for .NET Demo Theater Sessions

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Demo Theater Session</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue, Mar 06</td>
<td>1:00 pm - 1:30 pm</td>
<td>Getting Started</td>
<td>Demo Theater 1 - Oasis 1</td>
</tr>
<tr>
<td></td>
<td>4:00 pm - 4:30 pm</td>
<td>Custom States and Conditions</td>
<td>Demo Theater 2 - Oasis 1</td>
</tr>
<tr>
<td>Wed, Mar 07</td>
<td>5:30 pm - 6:00 pm</td>
<td>New UI Controls for the SDK</td>
<td>Demo Theater 2 - Oasis 1</td>
</tr>
<tr>
<td></td>
<td>6:00 pm - 6:30 pm</td>
<td>Raster API and Manipulating Pixel Blocks</td>
<td>Demo Theater 2 - Oasis 1</td>
</tr>
</tbody>
</table>

## ArcGIS Pro Road Ahead Sessions

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Technical Session</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue, Mar 06</td>
<td>4:00 pm – 5:00 pm</td>
<td>ArcGIS Pro: The Road Ahead</td>
<td>Oasis 4</td>
</tr>
<tr>
<td>Thu, Mar 08</td>
<td>4:00 pm – 5:00 pm</td>
<td>ArcGIS Pro: The Road Ahead</td>
<td>Primrose B</td>
</tr>
</tbody>
</table>