MARSHALLTOWN Software

MARSHALLTOWN Software SDK Introduction

Contents

Prerequisites & Recommendations 2
Service Architecture 2
SDK Components 3
MtCommon 3
MtCommon.Ax 3
MtCommon.Web 4
MobileScan 4
Example Projects 4
Further Reading 5

The MARSHALLTOWN Software SDK provides a ready-built framework for rapid development of applications which integrate with Microsoft Dynamics AX, and prescribes a robust, flexible, proven architecture for building decoupled services and responsive web applications.

Prerequisites & Recommendations

- Microsoft Visual Studio 2015 or later
- Microsoft .NET Framework 4.5.2 or later
- Microsoft Dynamics AX - AOS and .NET Business Connector or AIF
    - *For development of applications with Dynamics AX integration*
- Microsoft IIS (Internet Information Services) 8.0 or later, with ASP.NET 4.5 or later
    - *For development of web applications*
- Windows Server 2012 or later
    - *For hosting IIS*

Service Architecture

As developers, we struggle to develop and maintain a variety of applications which connect efficient user interfaces to back end data sources. The MARSHALLTOWN Software SDK provides the building blocks to implement patterns and architectures which have been designed to promote the following:

- Rapid application development
- Loose coupling between applications and data sources
- Protocol & platform independence
- Consistency among applications
- RESTful design

In general terms, the architecture prescribed by the SDK is fundamentally based on:

- Strongly-typed **Data Models**
    - Objects which are effectively property bags with no logic
    - Represent the data passed between systems (domains)
    - Optimized for and implemented consistently in each domain
- Common data envelope
    - Data models are passed between domains in a JSON **MtResult** envelope
    - Provides for transmission of results and errors from service calls
- Common pattern for S**ervice Interface/Implementation** layers as applicable to each domain; for example, Dynamics AX 2009 service implementations follow a common pattern involving:
    - Static **Service Interface** classes
        - Provide outward facing interface for a service, mapping/routing calls to actual (concrete) **Service Implementation** classes
        - Translate (serialize/deserialize) data to/from X++ **Data Models**
    - Concrete **Service Implementation** classes
        - Implement the business logic for the **Service Interface**
- Common pattern for **Service Proxy/Contract/Facade** layers enabling service consumption from C# applications, again leveraging base classes and objects:
    - **Service Contract** interfaces
        - Enable loose coupling between C# code and service implementations

- *Optional* **Service Facade** classes wrap **Service Contracts**, providing (if needed):
  - Point of commonality for shared, non-implementation-specific logic
  - Additional layer of decoupling
- **Service Proxy** classes
  - Implement the **Service Contract**
  - Invoke **Service Interfaces** across domains
  - Translate (serialize/deserialize) data to/from C# **Data Models**

Development of each layer is simplified by the SDK.

SDK Components

C# components are provided both as precompiled NuGet packages and as source code. X++ components are provided via XPO files.

MtCommon

The MtCommon library (for C# and X++) provides common, critical SDK components which are useful regardless of the type of application being developed (desktop or web).

- **MtCommon.dll** – C# class library / NuGet package
  - Primary contents:
    - C# data envelope implementation (**MtResult<T>**)
    - Objects critical to error handling mechanisms:
      - **MtException**
      - **MtApiCallException/MtApiCallInfo** (provides debug information about service API calls)
    - Interfaces and classes simplifying access to application configuration settings used by the SDK
- **MtCommon** - X++ object library
  - Primary contents:
    - X++ data envelope implementation (**MT_Result**)
    - JSON serialization/deserialization classes (**MT_JSON**, **MT_TextStream**)
    - Class enabling serialization of complex Data Models (**MT_TypeSpec**)
    - Service Implementation base class & supporting objects (**MT_Service_Base**)

MtCommon.Ax

This C# library provides components which simplify the implementation of the Service Proxy layer for a Dynamics AX service implementation within any C# application (desktop or web).

- **MtCommon.Ax.dll –** C# class library / NuGet package
  - Primary contents:
    - Service Proxy layer base class (**AxBase**), handling:
      - Example: Invocation of AX 2009 Service Interface layer via Business Connector
      - Data envelope validation
      - Data Model translation (serialization/deserialization)
    - Objects critical to error handling mechanisms:

        - **MtAxApiCallInfo** (implementation of **MtApiCallInfo**, provides debug information about Business Connector calls with AX 2009)
    - Interface/class simplifying access to Business Connector settings (**IAxBusinessConnectorSettings/MtAxBusinessConnectorSettings**)

MtCommon.Web

This C# library provides SDK components specifically useful for developing web applications implemented with ASP.NET MVC. When consumed by a new web project as a NuGet package, the package installs several out-of-box templates for standard MVC project files which help to bootstrap the MARSHALLTOWN Software SDK – yielding a quick, ready starting point for development.

- MtCommon.Web.dll – C# class library / NuGet package
    - Primary contents:
        - **Mt-Metro** UI CSS (based on Metro UI CSS) enabling responsive, mobile-friendly UI design
        - Ready-made shared view layout (**_Layout.cshtml**) and related:
            - **MtNavigationContextModel**
                - Base class for strongly typed, shared view model
            - **MtController<TNavigationContextModel>**
                - Controller base class wrapping shared view model
            - Navigation and debug menu-building classes
        - Mechanisms for:
            - Error handling
            - Developer debugging
            - Simple UI user prompts
            - Routing/navigation simplification
            - Dependency injection using **Unity.config** to loosely bind service libraries at run-time
            - Implementing MtResult-aware Web API services compatible with the prescribed service architecture

MobileScan

MobileScan is an iOS application available on the iOS App Store.

- Wraps web applications in a full-screen browser experience for mobile devices
- Enables integration of web applications with MARSHALLTOWN barcode scanner hardware (e.g. MT1890)
    - JavaScript interface to accept and process scans

Example Projects

Three example web projects are provided which demonstrate the service architecture and key SDK features. The **MARSHALLTOWN Software SDK Bootstrapping Guide** provides more detail about each example as well as everything necessary to get started with the projects.

- **FooBar.Web (Simplified)**

  Companion to the **Simplified** Service Architecture diagram, demonstrating an abstract scenario involving a simple AX service and a web application
- **FooBar.Web (Generalized)**

Companion to the **Generalized** Service Architecture diagram, demonstrating an abstract scenario involving a simple AX service and web application

- **MtMobility.Web**

  Simple inventory transfer AX service and web application demonstrating several key concepts

Further Reading

- **MARSHALLTOWN Software SDK Bootstrapping Guide**
  - Guide to getting started with source code and example projects
- **MARSHALLTOWN Software SDK Development Guide**
  - Comprehensive guide and reference for service and application development
- **MARSHALLTOWN Software SDK Style Guide**
  - Reference for UI design using the mt-metro CSS library
- **Service Architecture Drawings**
  - Comprehensive drawings of simplified and generalized service architecture approaches, intended for poster printing
- **Service Architectures for Application Development**
  - Original white paper detailing the guiding principles and evolution of the service architecture prescribed by the SDK