Production Timer Web API

By

## David Scott Blackburn

This document discusses the design of the back-end of the Production Timer - Production Timer Web API. The following is explained:

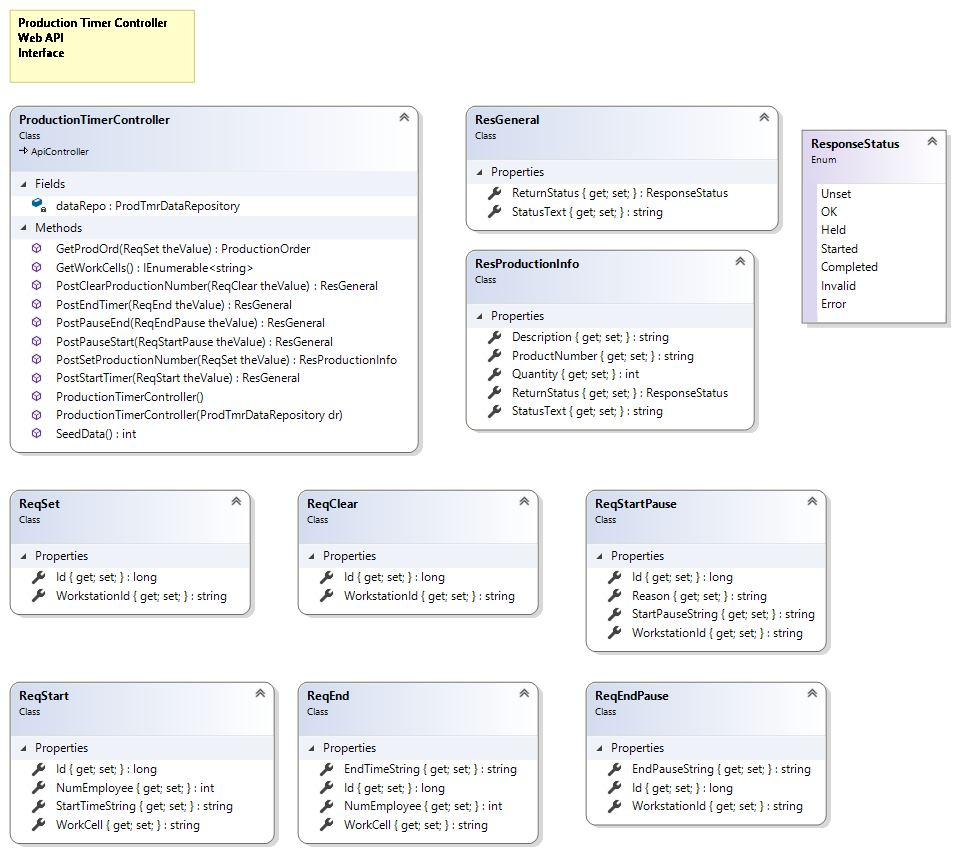
* RESTful Interface
* Interface Class structure
* Business & Entity Framework models
* Database Structure

This program is an emulation of a much larger and sophisticated system. The structures used in this project do not match in anyway the actual structures used in the original system. The data structure used here is the one I would use based on the portion of the system I emulated and to possible project expansion.

# RESTful Interface

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Call Type** | **URI** | **Purpose** | **Request Data** | **Response Data** |
| GET | api/ProductionTimer/WorkCells | Returns a list of Work Cells | None | Array of Strings |
| POST | api/ProductionTimer/Set | Sets Production Order status to SET and returns base Production Order information | ReqSet | ResProductionInfo |
| POST | api/ProductionTimer/Clear | Reset Production Order Status back to READY | ReqClear | ResGeneral |
| POST | api/ProductionTimer/StartTimer | Sets status to START and sets the start time | ReqStart | ResGeneral |
| POST | api/ProductionTimer/EndTimer | Sets status to COMPLETED and set the End Time and Duration | ReqEnd | ResGeneral |
| POST | api/ProductionTimer/PauseStart | Creates a Start Pause record for either a Production Order (Detail) or to Pause Unassigned | ReqStartPause | ResGeneral |
| POST | api/ProductionTimer/PauseEnd | Sets the end time and duration for the previously started pause. | ReqEndPause | ResGeneral |
| POST | api/ProductionTimer/SeedData | Causes the data to be reset | None | None |
| POST | api/ProductionTimer/Get | Returns details for a particular Production Order | ReqSet | ProductionOrder |

This is a simple Web API. In general, an action exists for each POST action. Data is converted, if needed, and a call made to the appropriate database repository method.



# Interface Class structure

The above diagram shows the details regarding the Interface classes. The previous section explained the details regarding the REST interface that is how information is transferred to and from the Web API.

Each of the Production Timer Controller methods (actions) makes a call to the Work Cell or Production Order Repository method.

# 

# Business and Entity Framework Classes

## Business Model Classes

**ProdTmrDataRepository** is used to allow for simple access to all of the data repositories available to the system. It is injected into the Production Timer Controller and provides a way to provide alternative data stores for tests or future development.

**IProductionOrder** and **IWorkCell** are interfaces that are implemented by the repository classes. These are defined so that other, alternate version repository classes, such as for a test project can be designed.

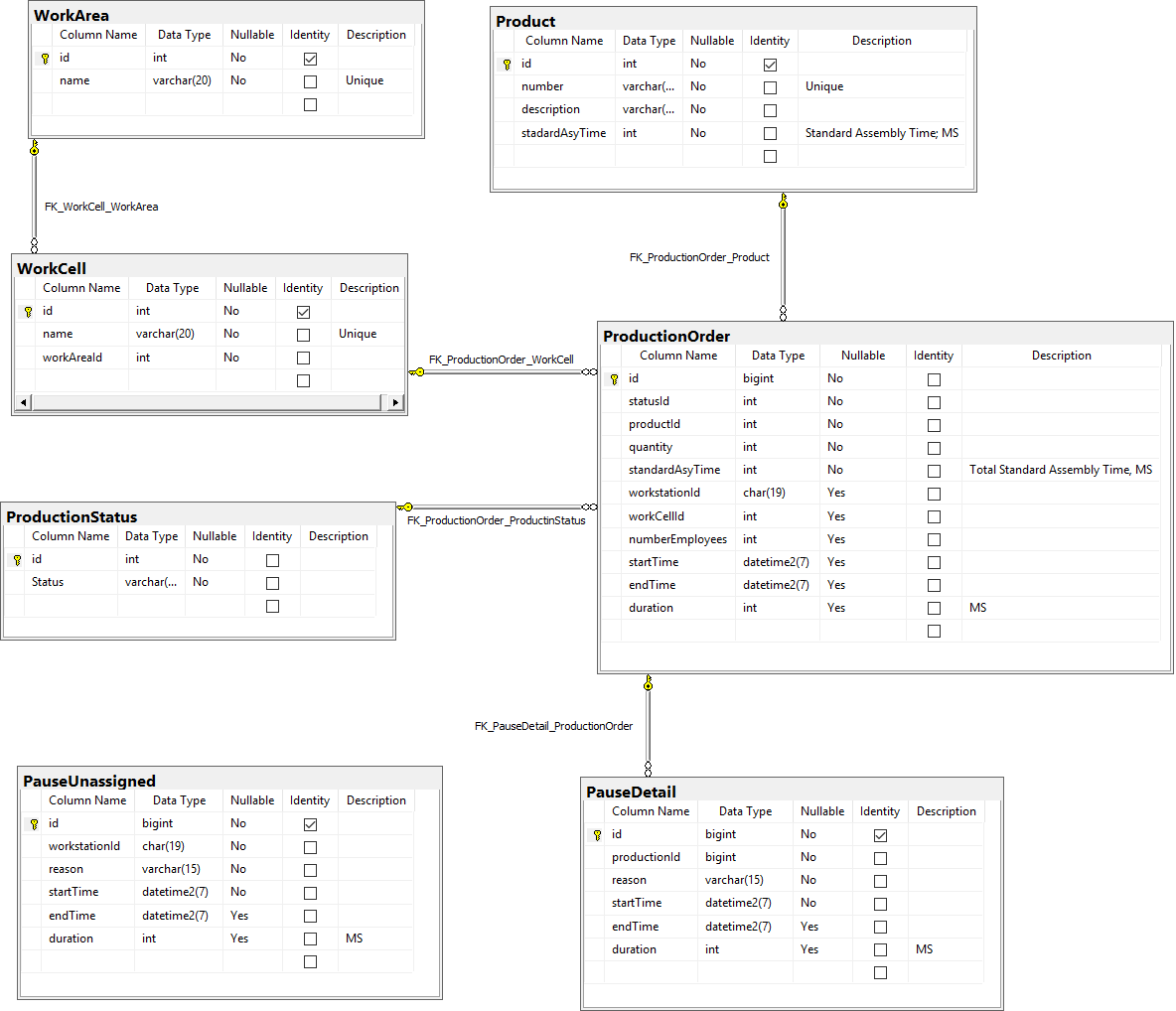
**ProductionOrderRepository** and **WorkCellRepository** are the primary classes that retrieve and process data. These classes makes calls to the Entity Framework class, which actually interfaces with the database.

## Entity Framework Classes

The only class you deal with here is the ProdClassEntities class. This class is set up via the Entity Framework tools and should not be directly manipulated. Stored procedures are the primary method of accessing the Production Order data. Most of the methods of this class require the same parameters that the stored procedure requires and returns the same results that the it returns.

# Database Structure

## Table Structure



## Stored Procedures

The stored procedures are called by the ProdTimerEntities class methods. Entity Framework handles the connection to and communication with SQL Server.

If you have questions or comments please contact me via email.