**The Chappy Service**

The main operations of Chappy are executed by a Web service. The architecture of the service consist of five projects:

* **Chappy\_DAL** – ORM for the database.
* **Chappy\_Model** – DTOs for the service.
* **Chappy\_Model.Extensions** – Helper classes to convert classes in the ORM to DTOs.
* **Chappy\_Service** – Holds the main service interface and its implementation.
* **Chappy\_Host** – Contains Logic that hosts the service on a remote IIS server.

**Chappy\_DAL**

DAL stands for Data Access Layer. Our goal was to create a clean bridge between the Chappy database and the service methods. This was achieved by the Object Relational Mapper (ORM) provided by .NET called the Entity Framework. The entity framework creates classes that represent the databases itself and it classes that represent the tables within the database. Instead of using the ADO.Net library and explicitly opening a connection to the database and executing queries, the classes that the Entity Framework created executes these task for you. However, The Chappy web service will be communicating to other devices remotely via passing JSON objects. The objects that the Entity Framework create has too much code for them to be an ideal object to serialize and send. Also, these classes are auto generated. If one adds serialization code to the entities, it will be erased after an update to the database. This is where the project Chappy\_Model comes in.

**Chappy\_Model**

This project contains classes which are called POCOs – in relation to ORMs and DALs, POCO stands for ‘Plain Old C# Objects’. POCOs are to contain only properties that represent the data – no behavior. For these to be serialized, each class will have an attribute on its declaration header called DataContract. And on each of the class’s property definitions will have an attribute named DataMember. These objects will be used as the arguments that will be required by our service methods and the return value for these methods. The more formal name of these is Data Transfer Objects (DTO). Since most of these DTOs are analogous to the classes that the ORM created, we must map from the ORM entity classes to the DTOs and vice versa. This is where the project Chappy\_Model.Extensions comes in.

**Chappy\_Model.Extensions**

This project contains helper classes that convert entity objects to DTO and vice versa. These helper classes are static classes that contain methods that use a feature of the C# called extension methods. These methods can be called by the entity or DTO objects as if these methods where members of those object allowing a very streamline coding experience. We were then ready for the service implementation itself.

**Chappy\_Service**

The Chappy\_Service is comprised of two main parts – The Interface and the service implementation.

The interface is decorated with the attribute of a ServiceContract, which indicates that the service can be interacted by authorized clients. Each method declaration inside the interface is tagged with the attribute of OperationContract which indicates that the method being declared is required with the ServiceContract. In addition, each method declaration includes a WebInvoke Attribute. The WebInvoke attribute will use the RequestFormat and ResponseFormat of JavaScript Object Notation (JSon) so that our web service can interact with clients no matter the platform.

For every method declared in our interface, we must implement that method and write the method definition. In general each method makes use of Microsoft Entity Framework using Language Integrated Query (LINQ) to retrieve and update data, and calling stored procedures using the Chappy\_DAL library.

Each method that returns an Object, returns a DTO version of the database object. As noted above in the Chappy Model, these Objects are used for the client to communicate to the service and vice versa.

**Chappy\_Host**

Chappy will be hosted on an IIS server. To achieve this we created another project that holds the .svc file which as host definition containing the language and name of the service. And the project contains the web.config which is an xml file with settings that specify how to connect to the database and the rules on how to send and receive messages from the Chappy\_Sevice endpoint.