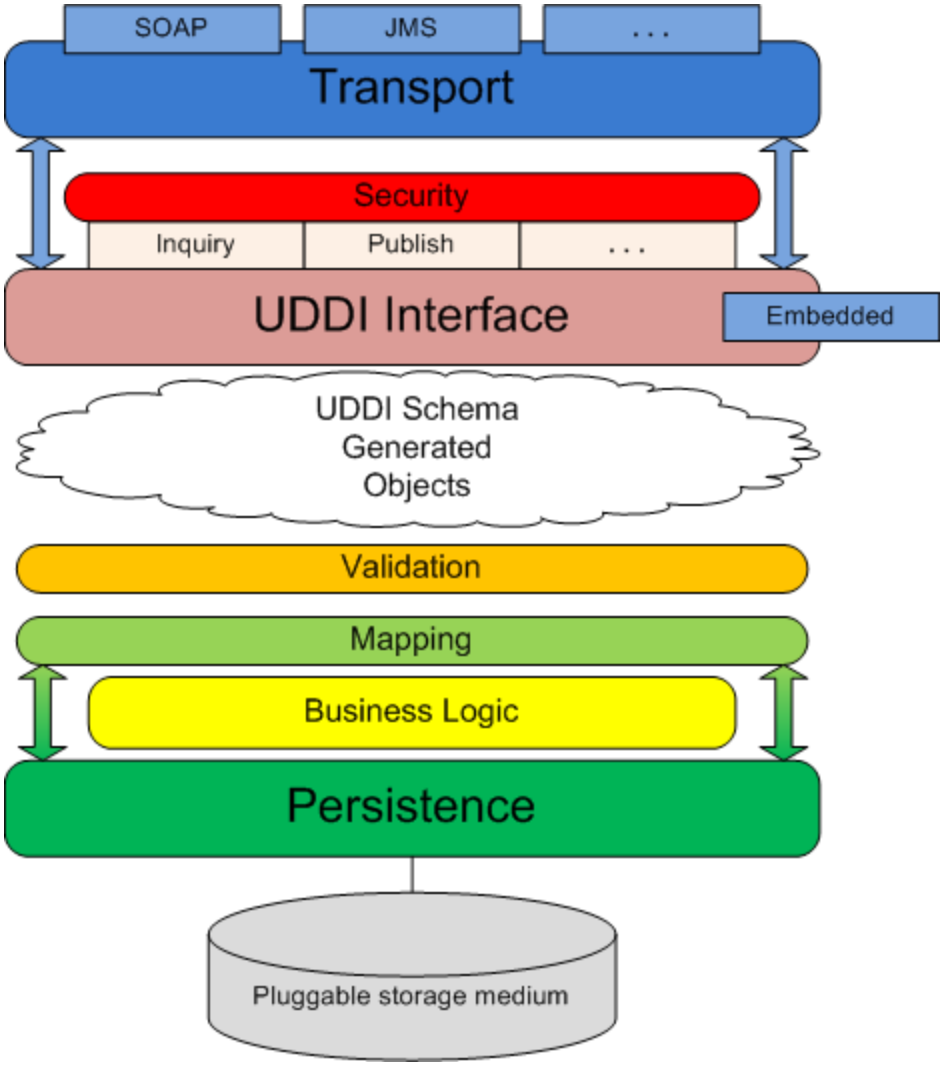
jUDDI Architecture for UDDI v3 Project



The diagram above shows the software layers and components employed in the jUDDI project implementation for UDDI v3. Here is a brief description of each item in the diagram and how they all work together to create the UDDI-compliant registry that is jUDDI:

Transport - the transport layer provides the means to receive and send out requests via a network or messaging service. The UDDI specification details an interface where XML messages are exchanged between client and server but is agnostic as to how those messages are relayed. By default, jUDDI uses Apache CXF to transport messages via SOAP over HTTP, however, the system is designed so other methods of transport can be easily plugged in (for example, JMS).

Security - security is provided by the UDDI specification and is based on policies defined in the specification. jUDDI implements all the mandatory policies and can be extended to support the optional policies. Chief among these policies is access control to the UDDI API exposed by jUDDI. jUDDI fully implements this policy, per the specification, which allows

users to easily plug in their own third-party authentication framework.

UDDI Interface - the UDDI interface defines the methods set forth by the UDDI specification to interact with the registry. Within jUDDI, the interface classes are generated from the UDDI WSDL and they are implemented as POJOs. These classes are annotated with JAX-WS annotations allowing end-users to easily employ any suitable JAX-WS container to expose the interface.

In general, the interface implementation accepts incoming UDDI-based requests and ummarshals these requests to the appropriate schema object. This object is then served to the proceeding layers so the necessary logic can be performed to fulfill the request. After the request is fulfilled, this layer is responsible for marshalling the result and sending the response to the requesting party.

As the interface is implemented as POJOs, it can be accessed via an "embedded" mode. In this scenario, the methods of the implementation classes can be called directly. This allows users to embed jUDDI directly into their application without having to deploy a full-blown jUDDI server.

UDDI Schema Objects - The UDDI specification comes equipped with an XML schema for its many data structures. jUDDI employs XML-binding technology (JAXB) to generate objects from the schema (contained within the WSDL) that are then used as the arguments for the UDDI Interface layer. These objects needn't originate from XML – they can also be instantiated directly to make UDDI calls directly in java code.

Validation – the validation layer reads the schema object input from the UDDI interface layer and, based on rules defined in the specification, makes sure the input is valid for the given UDDI method. Failed validation results in an exception and an immediate return from the method call.

Mapping – the mapping layer is responsible for mapping the UDDI schema objects to the persistence layer model. For all intents and purposes, the mapping layer simply copies data from a schema object to the similar model object. This occurs in both directions, as input objects must be mapped to the model to perform the necessary logic and results obtained from the call must be mapped back to the schema as output to the caller.

Business Logic - the business logic layer is responsible for performing all the business logic associated with the UDDI calls. The logic layer works with objects from the persistence layer and generally consists of querying the model based on user input.
.
Persistence - the persistence layer, as its name implies, is responsible for persisting registry data to a storage medium. To this end, a third-party persistence service that implements the Java Persistence API (Apache OpenJPA, Hibernate) is utilized to manage transactions with the storage medium and also to facilitate the plugging-in of various storage types. By default, jUDDI is packaged with Apache OpenJPA as the persistence provider and Apache Derby as the storage medium. This can easily be configured.