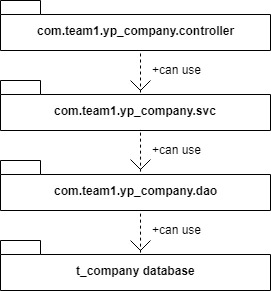
# Module View

## Layered View of Software Companies

### Section 1: The Primary Presentation



### Section 2: The Element Catalog

#### 2.1 com.team1.yp\_company.controller

The controller class inside this folder, namely ‘CompanyController’ in this web service, is built as the HTTP request processor. By annotated with @RestController, this class is equipped with the ability to process the incoming request and generate an HTTP response. In Spring framework, all requests that originated from UI are sent to built-in DispatcherServlet, which is the hub for all HTTP request, and DispatcherServlet would dispatch the request to the controller that is registered for processing a request that sent to a specified endpoint.

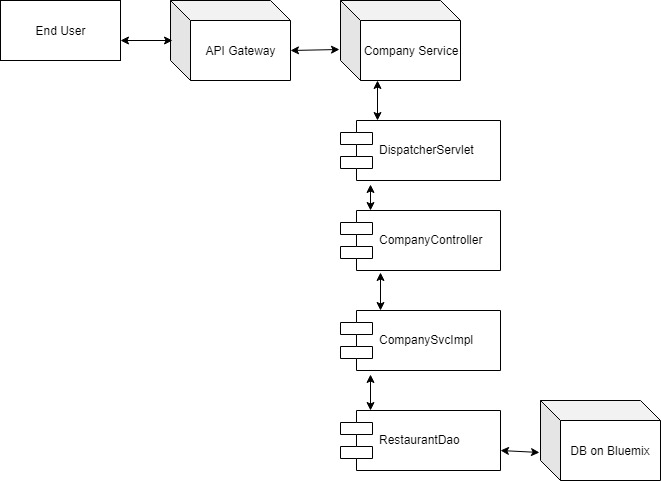
#### 2.2 com.team1.yp\_company.svc

The ‘CompanySvc’ interface, which locates in the fold, list the functions that the Company service is bound to offer. The ‘CompanySvcImpl’ is the implementation of the previous interface. These functions invoke corresponding functions in the ‘CompanyDao’ interface, retrieving data from DB.

#### 2.3 com.team1.yp\_company.dao

The ‘CompanyDao’ interface is implemented as a data-accessing-object or ‘Dao’ in short. Functions defined within this interface interact with the database to retrieve data. In the Company service system, the interaction with DB is implemented with Mybatis persistence framework. It maps functions in this interface to the SQL included in the ‘mapper.xml’ file, which significantly simplified the coding compared to traditional JDBC

## Section 3: Context Diagram



## Section 4: Variability Guide

There is no variability for this diagram.

## Section 5: Rationale

This diagram demonstrates the layered architecture of the web service. The design is derived from both SOA (service-oriented architecture) and MVC pattern. Specifically, class ‘CompanyController’ and ‘CompanySvcImpl’ are both components locate in the controller layer, while the interface ‘CompanyDao’ sit one layer below in the module layer. The whole system would be benefited by the layered design by better modifiability and scalability. In addition, the DI feature that comes with the Spring framework can lead to the desired low-coupling design, which would further improve the modifiability and scalability.