**Business Delegate**: Reduces coupling between presentation-tier clients and business services. It hides the underlying implementation details of the business service, such as lookup and access details of the EJB architecture.

The *business delegate pattern* tries to decouple the clients from the business services. To achieve this you need:

* **business delegate** that is the object used by clients to request for services;
* **lookup service** is a bridge used by **business delegate** to search for services, it encapsulates the search algorithm according to the request made by the delegate;
* **business service** is the actual service that is offered to clients, usually an EJB or similar J2EE concepts.

By the way this [page](http://www.oracle.com/technetwork/java/businessdelegate-137562.html) explains everything quite clearly..

**Core J2EE Patterns - Business Delegate**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  |  | | --- | --- | |  |  | |

**Context**

A multi-tiered, distributed system requires remote method invocations to send and receive data across tiers. Clients are exposed to the complexity of dealing with distributed components.

**Problem**

Presentation-tier components interact directly with business services. This direct interaction exposes the underlying implementation details of the business service application program interface (API) to the presentation tier. As a result, the presentation-tier components are vulnerable to changes in the implementation of the business services: When the implementation of the business services change, the exposed implementation code in the presentation tier must change too.

Additionally, there may be a detrimental impact on network performance because presentation-tier components that use the business service API make too many invocations over the network. This happens when presentation-tier components use the underlying API directly, with no client-side caching mechanism or aggregating service.

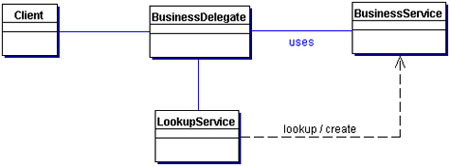
Lastly, exposing the service APIs directly to the client forces the client to deal with the networking issues associated with the distributed nature of Enterprise JavaBeans (EJB) technology.

**Forces**

* Presentation-tier clients need access to business services.
* Different clients, such as devices, Web clients, and thick clients, need access to business service.
* Business services APIs may change as business requirements evolve.
* It is desirable to minimize coupling between presentation-tier clients and the business service, thus hiding the underlying implementation details of the service, such as lookup and access.
* Clients may need to implement caching mechanisms for business service information.
* It is desirable to reduce network traffic between client and business services.

### Structure

Figure 8.1 shows the class diagram representing the Business Delegate pattern. The client requests the BusinessDelegate to provide access to the underlying business service. The BusinessDelegate uses a LookupService to locate the required BusinessService component.

   
**Figure 8.1 BusinessDelegate class diagram**