Service Locator

Context

Service lookup and creation involves complex interfaces and network operations.

Problem

J2EE clients interact with service components, such as EJB and JMS components, which provide business services and persistence capabilities. To interact with these components, clients must either locate the service component (referred to as a lookup operation) or create a new component. For instance, an EJB client must locate the enterprise bean's home object, which the client then uses either to find an object or to create or remove one or more enterprise beans. Similarly, a JMS client must first locate the JMS Connection Factory to obtain a JMS Connection or a JMS Session.

All J2EE application clients use the JNDI common facility to look up and create EJB and JMS components. The JNDI API enables clients to obtain an initial context object that holds the component name to object bindings. The client begins by obtaining the initial context for a bean's home object. The initial context remains valid while the client session is valid. The client provides the JNDI registered name for the required object to obtain a reference to an administered object. In the context of an EJB application, a typical administered object is an enterprise bean's home object. For JMS applications, the administered object can be a JMS Connection Factory (for a Topic or a Queue) or a JMS Destination (a Topic or a Queue).

So, locating a JNDI-administered service object is common to all clients that need to access that service object. That being the case, it is easy to see that many types of clients repeatedly use the JNDI service, and the JNDI code appears multiple times across these clients. This results in an unnecessary duplication of code in the clients that need to look up services.

Also, creating a JNDI initial context object and performing a lookup on an EJB home object utilizes significant resources. If multiple clients repeatedly require the same bean home object, such duplicate effort can negatively impact application performance.

Let us examine the lookup and creation process for various J2EE components.

1. The lookup and creation of enterprise beans relies upon the following:
   - A correct setup of the JNDI environment so that it connects to the naming and directory service used by the application. Setup entails providing the location of the naming service and the necessary authentication credentials to access that service.
   - The JNDI service can then provide the client with an initial context that acts as a placeholder for the component name-to-object bindings. The client requests this initial context to look up the EJBHome object for the required enterprise bean by providing the JNDI name for that EJBHome object.
   - Find the EJBHome object using the initial context's lookup mechanism.
   - After obtaining the EJBHome object, create, remove, or find the enterprise bean, using the EJBHome object's create, move, and find (for entity beans only).
2. The lookup and creation of JMS components (Topic, Queue, QueueConnection, QueueSession, TopicConnection, TopicSession, and so forth) involves the following steps. Note that in these steps, Topic refers to the publish/subscribe messaging model and Queue refers to the point-to-point messaging model.
   - Set up the JNDI environment to the naming service used by the application. Setup entails providing the location of the naming service and the necessary authentication credentials to access that service.

- Obtain the initial context for the JMS service provider from the JNDI naming service.
- Use the initial context to obtain a Topic or a Queue by supplying the JNDI name for the topic or the queue. Topic and Queue are JMSDestination objects.
- Use the initial context to obtain a TopicConnectionFactory or a QueueConnectionFactory by supplying the JNDI name for the topic or queue connection factory.
- Use the TopicConnectionFactory to obtain a TopicConnection or QueueConnectionFactory to obtain a QueueConnection.
- Use the TopicConnection to obtain a TopicSession or a QueueConnection to obtain a QueueSession.
- Use the TopicSession to obtain a TopicSubscriber or a TopicPublisher for the required Topic. Use the QueueSession to obtain a QueueReceiver or a QueueSender for the required Queue.

The process to look up and create components involves a vendor-supplied context factory implementation. This introduces vendor dependency in the application clients that need to use the JNDI lookup facility to locate the enterprise beans and JMS components, such as topics, queues, and connection factory objects.

Forces

- EJB clients need to use the JNDI API to look up EJBHome objects by using the enterprise bean's registered JNDI name.
- JMS clients need to use JNDI API to look up JMS components by using the JNDI names registered for JMS components, such as connection factories, queues, and topics.
- The context factory to use for the initial JNDI context creation is provided by the service provider vendor and is therefore vendor- dependent. The context factory is also dependent on the type of object being looked up. The context for JMS is different from the context for EJB, with different providers.
- Lookup and creation of service components could be complex and may be used repeatedly in multiple clients in the application.
- Initial context creation and service object lookups, if frequently required, can be resource-intensive and may impact application performance. This is especially true if the clients and the services are located in different tiers.
- EJB clients may need to reestablish connection to a previously accessed enterprise bean instance, having only its Handle object.

Solution

Use a Service Locator object to abstract all JNDI usage and to hide the complexities of initial context creation, EJB home object lookup, and EJB object re-creation. Multiple clients can reuse the Service Locator object to reduce code complexity, provide a single point of control, and improve performance by providing a caching facility.

This pattern reduces the client complexity that results from the client's dependency on and need to perform lookup and creation processes, which are resource-intensive. To eliminate these problems, this pattern provides a mechanism to abstract all dependencies and network details into the Service Locator.

Structure

Figure 1.1 shows the class diagram representing the relationships for the Service Locator pattern.

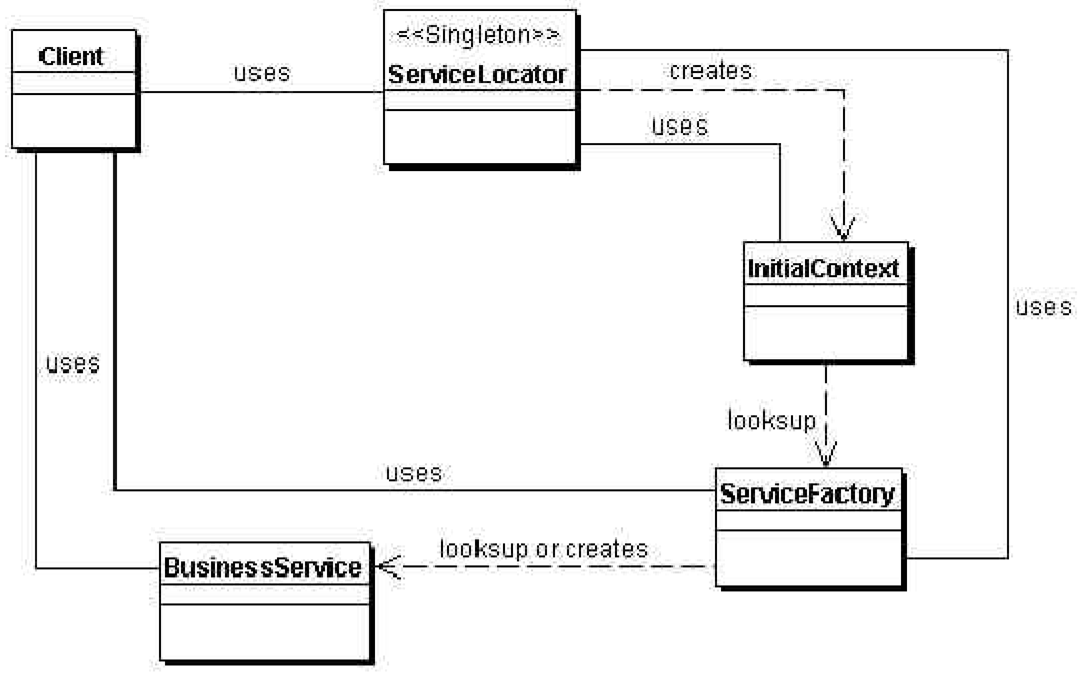

Figure 1.1 Service Locator class diagram.

Participants and Responsibilities

Figure 1.2 contains the sequence diagram that shows the interaction between the various participants of the Service Locator pattern.

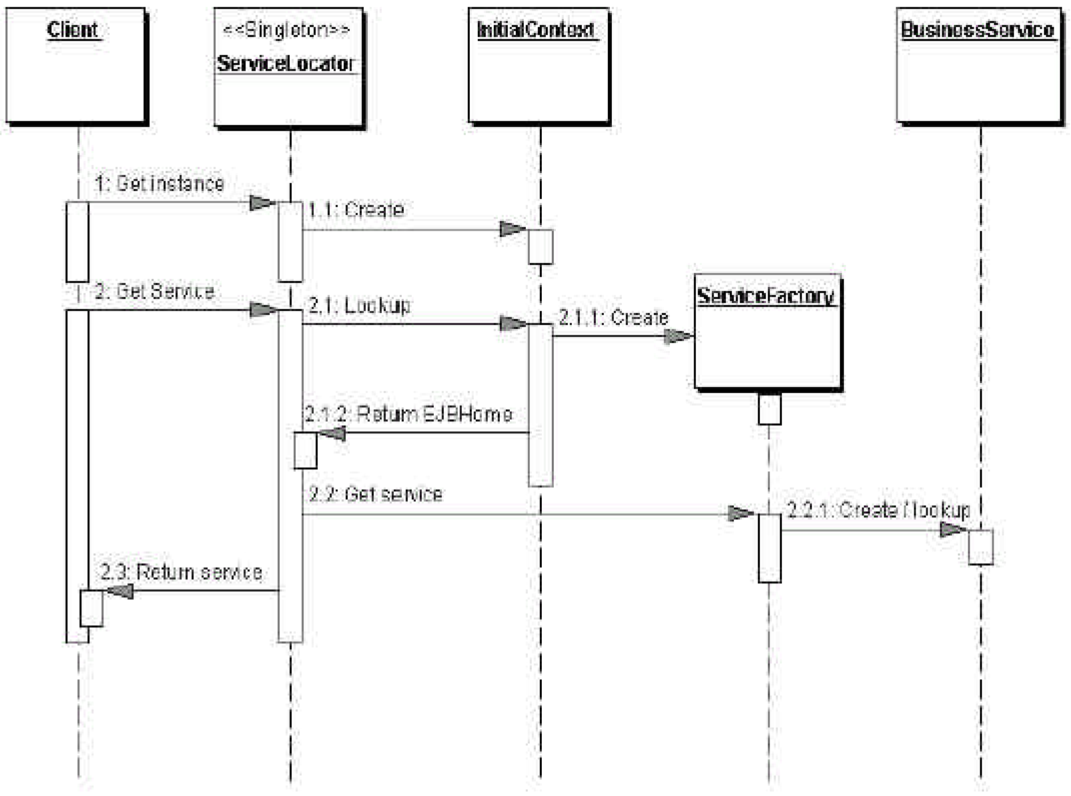

Figure 1.2 Service Locator Sequence diagram.

Client

This is the client of the Service Locator. The client is an object that typically requires access to business objects such as a Business Delegate.

Service Locator

The Service Locator abstracts the API lookup (naming) services, vendor dependencies, lookup complexities, and business object creation, and provides a simple interface to clients. This reduces the client's complexity. In addition, the same client or other clients can reuse the Service Locator.

InitialContext

The InitialContext object is the start point in the lookup and creation process. Service providers provide the context object, which varies depending on the type of business object provided by the Service Locator's lookup and creation service. A Service Locator that provides services for multiple types of business objects (such as enterprise beans, JMS components, and so forth) utilizes multiple types of context objects, each obtained from a different provider (e.g., context provider for an EJB application server may be different from the context provider for JMS service).

ServiceFactory

The ServiceFactory object represents an object that provides life cycle management for the BusinessService objects. The ServiceFactory object for enterprise beans is an EJBHome object.

The ServiceFactory for JMS components can be a JMS ConnectionFactory object, such as a TopicConnectionFactory (for publish/subscribe messaging model) or a QueueConnectionFactory (for point-to-point messaging model).

BusinessService

The BusinessService is a role that is fulfilled by the service the client is seeking to access. The BusinessService object is created or looked up or removed by the ServiceFactory. The BusinessService object in the context of an EJB application is an enterprise bean. The BusinessService object in the context of a JMS application can be a TopicConnection or a QueueConnection. The TopicConnection and QueueConnection can then be used to produce a JMSSession object, such as TopicSession or a QueueSession respectively.