Padrão de Design - Padrão do Localizador de Serviço

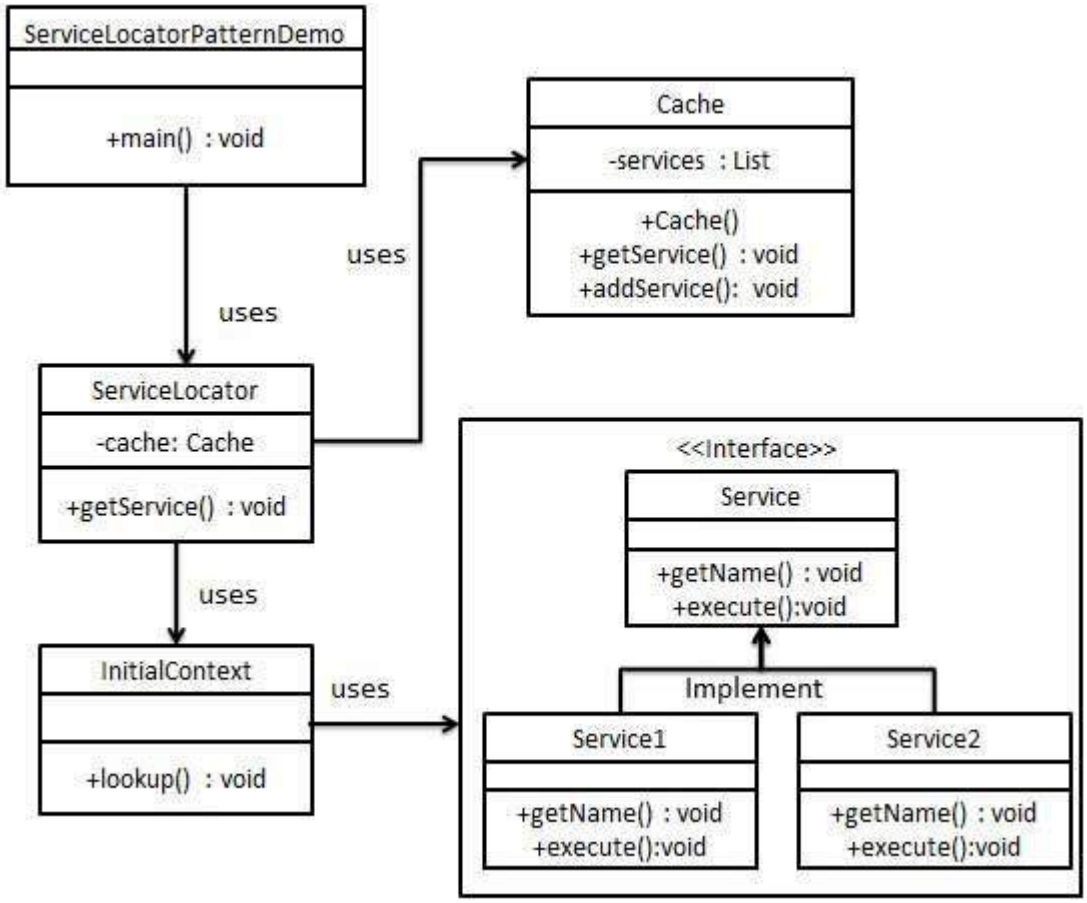
O padrão de design do localizador de serviço é usado quando queremos localizar vários serviços usando a consulta JNDI. Considerando o alto custo de procurar um serviço na JNDI, o padrão Service Locator utiliza a técnica de armazenamento em cache. Pela primeira vez em que um serviço é necessário, o Localizador de Serviços consulta no JNDI e armazena em cache o objeto de serviço. Uma pesquisa adicional ou o mesmo serviço via Service Locator é feita em seu cache, o que melhora o desempenho do aplicativo em grande medida. A seguir, estão as entidades desse tipo de padrão de design.

- **Serviço** - **Serviço** real que processará a solicitação. A referência desse serviço deve ser vista no servidor JNDI.
- **Contexto / Contexto Inicial** - **O contexto** JNDI carrega a referência ao serviço usado para fins de pesquisa.
- **Localizador de serviço** - o **Localizador de** serviço é um ponto de contato único para obter serviços pela pesquisa JNDI que armazena em cache os serviços.
- **Cache** - Cache para armazenar referências de serviços para reutilizá-los
- **Cliente** - Cliente é o objeto que chama os serviços via ServiceLocator.

Implementação

Vamos criar um *ServiceLocator* , *InitialContext* , *Cache* , *Service* como vários objetos que representam nossas entidades. *Serviço1* e *Serviço2* representam serviços concretos.

ServiceLocatorPatternDemo , nossa classe demo, está atuando como um cliente aqui e usará o *ServiceLocator* para demonstrar o Padrão de Design do Localizador de Serviço.



Passo 1

Criar interface de serviço.

Service.java

```
public interface Service {
   public String getName();
   public void execute();
}
```

Passo 2

Crie serviços concretos.

Service1.java

```
public class Service1 implements Service {
   public void execute(){
      System.out.println("Executing Service1");
   }

   @Override
```

```
   public String getName() {
      return "Service1";
   }
}
```

Service2.java

```
public class Service2 implements Service {
   public void execute(){
      System.out.println("Executing Service2");
   }

   @Override
   public String getName() {
      return "Service2";
   }
}
```

etapa 3

Criar InitialContext para pesquisa JNDI

InitialContext.java

```
public class InitialContext {
   public Object lookup(String jndiName){

      if(jndiName.equalsIgnoreCase("SERVICE1")){
         System.out.println("Looking up and creating a new Service1 object");
         return new Service1();
      }
      else if (jndiName.equalsIgnoreCase("SERVICE2")){
         System.out.println("Looking up and creating a new Service2 object");
         return new Service2();
      }
      return null;
   }
}
```

Passo 4

Criar cache

Cache.java

```
import java.util.ArrayList;
import java.util.List;
```

```
public class Cache {

   private List<Service> services;

   public Cache(){
      services = new ArrayList<Service>();
   }

   public Service getService(String serviceName){

      for (Service service : services) {
         if(service.getName().equalsIgnoreCase(serviceName)){
            System.out.println("Returning cached  " + serviceName + " object");
            return service;
         }
      }
      return null;
   }

   public void addService(Service newService){
      boolean exists = false;

      for (Service service : services) {
         if(service.getName().equalsIgnoreCase(newService.getName())){
            exists = true;
         }
      }
      if(!exists){
         services.add(newService);
      }
   }
}
```

Etapa 5

Criar localizador de serviço

ServiceLocator.java

```
public class ServiceLocator {
   private static Cache cache;

   static {
      cache = new Cache();
   }
```

```
   public static Service getService(String jndiName){

      Service service = cache.getService(jndiName);

      if(service != null){
         return service;
      }

      InitialContext context = new InitialContext();
      Service service1 = (Service)context.lookup(jndiName);
      cache.addService(service1);
      return service1;
   }
}
```

Etapa 6

Use o *ServiceLocator* para demonstrar o Service Locator Design Pattern.

ServiceLocatorPatternDemo.java

```
public class ServiceLocatorPatternDemo {
   public static void main(String[] args) {
      Service service = ServiceLocator.getService("Service1");
      service.execute();
      service = ServiceLocator.getService("Service2");
      service.execute();
      service = ServiceLocator.getService("Service1");
      service.execute();
      service = ServiceLocator.getService("Service2");
      service.execute();
   }
}
```

Etapa 7

Verifique a saída.

```
Looking up and creating a new Service1 object
Executing Service1
Looking up and creating a new Service2 object
Executing Service2
Returning cached  Service1 object
Executing Service1
```

```
Returning cached  Service2 object
Executing Service2
```