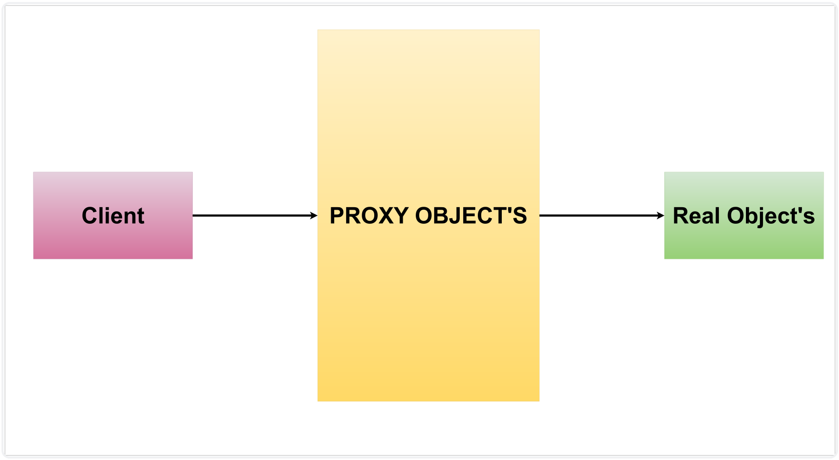
Proxy Design Pattern Explanation | LLD System Design | System Design Interview Question | Java

PROXY Design Pattern

- when client request for **real object** then request cross through mediator called Proxy Object's.



Real Life Usage:

- When User try to access real internet then request goes through PROXY and proxy have list of blocked IP-addresses. **[ACCESS DENIED]**

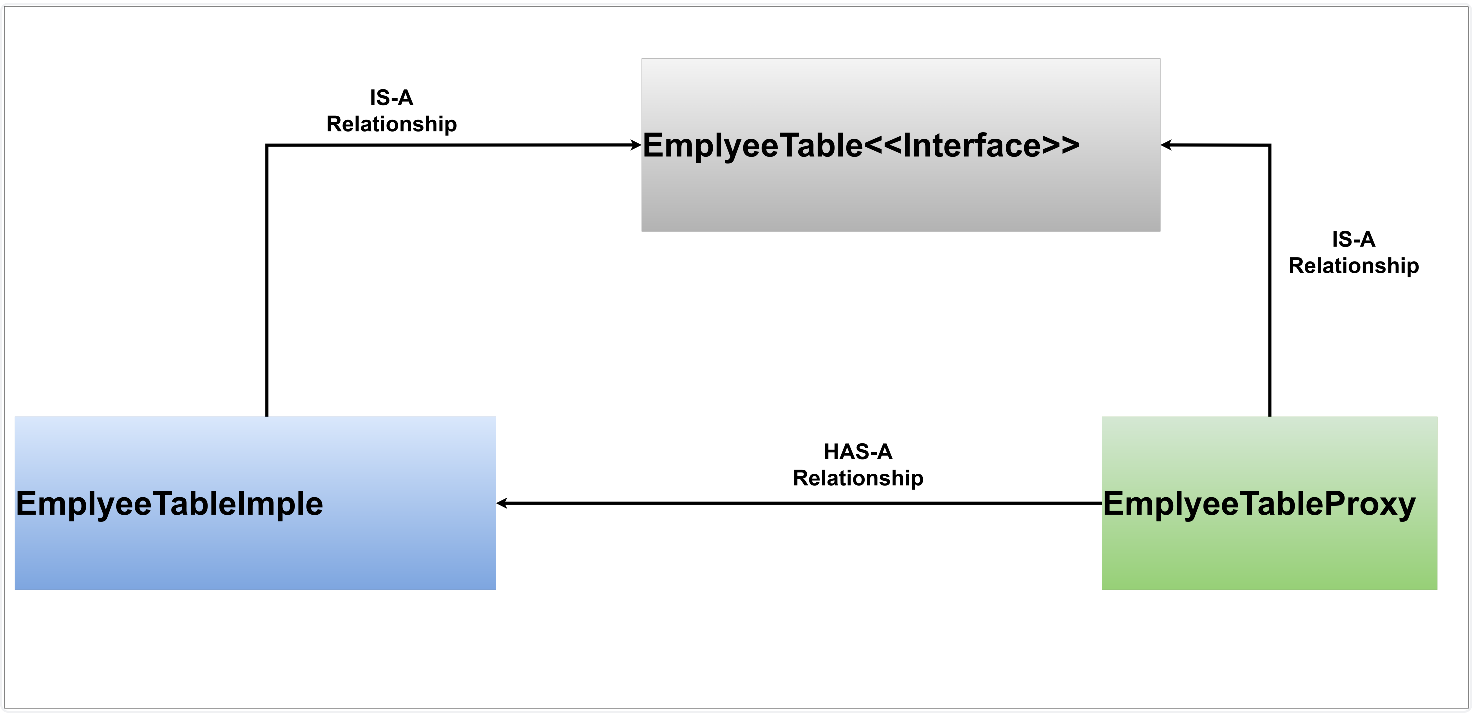
In Caching

- client want's some data then proxy first check in cache memory
- if data is available in cache then it's fine and it will return those data to client.
- if data is not available in cache then it goes for database request.

Pre-processing and Post-Processing

- using proxy we can do some pre-processing task before actual calling to real object.
- Using proxy we can do some post- processing task after calling real objects

SPRING BOOT BEAN CREATION BEAN PROXY



code:

```java
package System_Design.DesignPatterns.Proxy_Design_Pattern;

public class EmployeeDo {
}
```

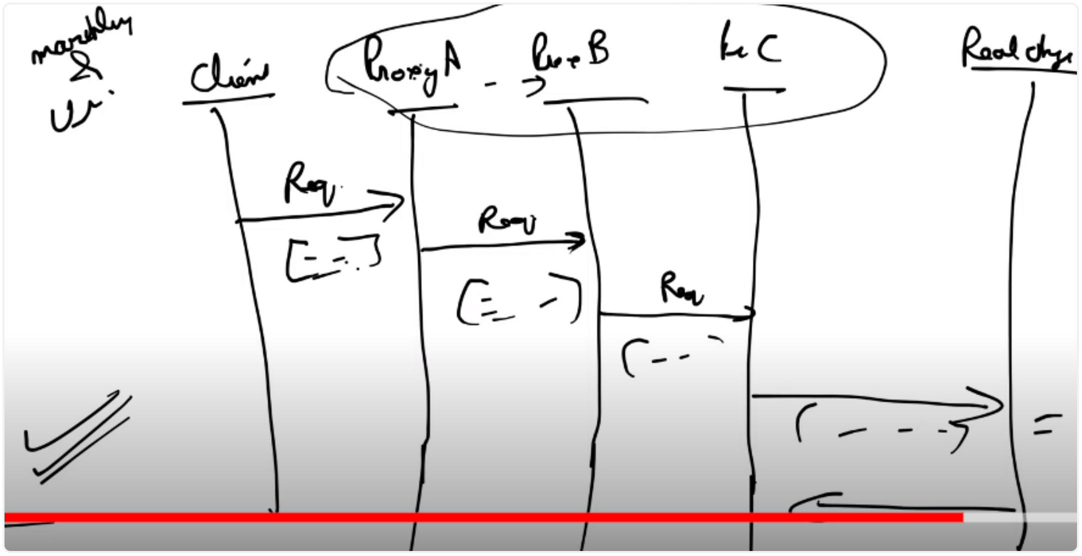
```java
package System_Design.DesignPatterns.Proxy_Design_Pattern;

public interface EmployeeDao {
    public void create(String client, EmployeeDo obj) throws Exception;
    public void delete(String client, int employeeId) throws Exception;
    public EmployeeDo get(String client, int employeeId) throws Exception;
}
```

```java
package System_Design.DesignPatterns.Proxy_Design_Pattern;

public class EmployeeDaoImple implements EmployeeDao{
    @Override
    public void create(String client, EmployeeDo obj) throws Exception {
        //creates new ROW
        System.out.println("Created new Row in Employee Table.");
    }

    @Override
    public void delete(String client, int employeeId) throws Exception {
        //delete a row
        System.out.println("deleted Row with employeeId: "+employeeId);
    }

    @Override
    public EmployeeDo get(String client, int employeeId) throws Exception {
        //fetch row

        System.out.println("Fetching Data From Db");
        return new EmployeeDo();
    }
}
```

```
package System_Design.DesignPatterns.Proxy_Design_Pattern;

public class EmployeeDaoProxy implements EmployeeDao{

    EmployeeDao employeeDaoObj;
    EmployeeDaoProxy(){
        employeeDaoObj=new EmployeeDaoImple();
    }
    @Override
    public void create(String client, EmployeeDo obj) throws Exception {
        if(client.equals("ADMIN")){
            employeeDaoObj.create(client,obj);
            return;
        }
        throw new Exception("Access Denied");
    }

    @Override
    public void delete(String client, int employeeId) throws Exception {

        if(client.equals("ADMIN")){
            employeeDaoObj.delete(client,employeeId);
            return;
        }
        throw new Exception("Access Denied");
    }

    @Override
    public EmployeeDo get(String client, int employeeId) throws Exception {
        if(client.equals("ADMIN") || client.equals("USER")){
           return employeeDaoObj.get(client,employeeId);

        }
        throw new Exception("Access Denied");
    }
}
```

```
package System_Design.DesignPatterns.Proxy_Design_Pattern;

public class ProxyDesignPattern {
    public static void main(String[] args){
        try{
           EmployeeDao empObjProxy = new EmployeeDaoProxy();
           empObjProxy.create("ADMIN",new EmployeeDo());
           System.out.println("Operation Successful");
        }catch (Exception e){
            System.out.println(e.getMessage());
        }
    }
}
output
Created new Row in Employee Table.
Operation Successful

Process finished with exit code 0
```

Marshaling and Un-Marshaling [Multiple Proxy]



An object that is marshalled records the state of the original object and it contains the codebase (codebase here refers to a list of URLs where the object code can be loaded from, and not source code). Hence, in order to convert the object state and codebase(s), unmarshalling must be done.

In RPC, Marshalling and Unmarshalling can be performed on both, the client-side and server-side. Marshalling is the packing of arguments(or parameters) into a message packet whereas Unmarshalling is the unpacking of arguments(or parameters) received from the call packet.