Sistema de Consulta de Livros

Vamos desenvolver uma aplicação onde o cliente pode acessar informações de um catálogo de livros no servidor. O servidor irá armazenar um conjunto simples de livros e permitirá que o cliente consulte informações específicas, como título, autor e ano de publicação.

Passo a Passo: Exemplo de Sistema de Consulta de Livros

1. Criação da Interface Remota

Crie uma interface remota chamada `CatalogoLivros`, que define os métodos disponíveis para consulta de livros.

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

public interface CatalogoLivros extends Remote {
    public Livro consultarLivro(String titulo) throws
RemoteException;
    public List<Livro> listarLivros() throws RemoteException;

}
```

2. Criação da Classe Livro

Defina uma classe simples chamada `Livro` que representará os livros com informações como título, autor e ano de publicação.

```
import java.io.Serializable;

public class Livro implements Serializable {
    private String titulo;
    private String autor;
    private int anoPublicacao;

    public Livro(String titulo, String autor, int
anoPublicacao) {
```

```
        this.titulo = titulo;
        this.autor = autor;
        this.anoPublicacao = anoPublicacao;
    }

    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public int getAnoPublicacao() {
        return anoPublicacao;
    }

    @Override
    public String toString() {
        return "Livro{" +
                "titulo='" + titulo + '\'' +
                ", autor='" + autor + '\'' +
                ", anoPublicacao=" + anoPublicacao +
                '}';
    }

}
```

3. Implementação da Interface

Agora implemente a interface `CatalogoLivros` no servidor, criando uma lista de livros e os métodos para consultar e listar os livros.

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.List;

public class CatalogoLivrosImpl extends UnicastRemoteObject
implements CatalogoLivros {
    private List<Livro> livros;
```

```
    public CatalogoLivrosImpl() throws RemoteException {
        super();
        livros = new ArrayList<>();
        // Adicionando alguns livros ao catálogo
        livros.add(new Livro("1984", "George Orwell", 1949));
        livros.add(new Livro("A Metamorfose", "Franz Kafka",
1915));
        livros.add(new Livro("O Pequeno Príncipe", "Antoine
de Saint-Exupéry", 1943));
    }

    public Livro consultarLivro(String titulo) throws
RemoteException {
        for (Livro livro : livros) {
            if (livro.getTitulo().equalsIgnoreCase(titulo)) {
                return livro;
            }
        }
        return null; // Retorna nulo se o livro não for
encontrado
    }

    public List<Livro> listarLivros() throws RemoteException
{
        return livros;
    }

}
```

4. Configuração do Servidor

Crie o servidor RMI que registra a implementação `CatalogoLivrosImpl`.

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class Servidor {
    public static void main(String[] args) {
        try {
```

```
            CatalogoLivrosImpl catalogo = new
CatalogoLivrosImpl();
            Registry registry =
LocateRegistry.createRegistry(1099);
            registry.rebind("CatalogoLivros", catalogo);
            System.out.println("Servidor de catálogo de
livros está pronto!");
        } catch (Exception e) {
            System.err.println("Erro no servidor: " +
e.toString());
            e.printStackTrace();
        }
    }

}
```

5. Criação do Cliente

Agora, crie um cliente que irá consultar o catálogo de livros disponíveis no servidor.

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.List;

public class Cliente {
    public static void main(String[] args) {
        try {
            Registry registry =
LocateRegistry.getRegistry("localhost", 1099);
            CatalogoLivros catalogo = (CatalogoLivros)
registry.lookup("CatalogoLivros");

            // Listando todos os livros
            List<Livro> livros = catalogo.listarLivros();
            System.out.println("Lista de livros
disponíveis:");
            for (Livro livro : livros) {
                System.out.println(livro);
            }

            // Consultando um livro específico
```

```
            Livro livroConsultado =
catalogo.consultarLivro("1984");
            if (livroConsultado != null) {
                System.out.println("\nLivro Consultado: " +
livroConsultado);
            } else {
                System.out.println("\nLivro não
encontrado.");
            }
        } catch (Exception e) {
            System.err.println("Erro no cliente: " +
e.toString());
            e.printStackTrace();
        }
    }

}
```

6. Compilação e Execução

Compile todos os arquivos Java:

```
javac *.java
```

Inicie o RMI registry (caso não esteja usando o `LocateRegistry.createRegistry`):

```
rmiregistry
```

Primeiro, execute o servidor:

```
java Servidor
```

Em seguida, execute o cliente numa nova janela de terminal:

```
java Cliente
```

Resultado Esperado

Quando você executar o cliente, ele deverá listar todos os livros disponíveis e consultar o livro "1984". A saída deve ser semelhante ao seguinte:

```
Lista de livros disponíveis:
Livro{titulo='1984', autor='George Orwell', anoPublicacao=1949}
Livro{titulo='A Metamorfose', autor='Franz Kafka', anoPublicacao=1915}
```

```
Livro{titulo='O Pequeno Príncipe', autor='Antoine de Saint-Exupéry',
anoPublicacao=1943}

Livro Consultado: Livro{titulo='1984', autor='George Orwell',
anoPublicacao=1949}
```

Conclusão

Esse exemplo de sistema de consulta de livros ilustra como você pode usar Java RMI para criar uma aplicação distribuída onde um cliente pode acessar informações armazenadas em um servidor remoto. A aplicação pode ser expandida para incluir mais funcionalidades, como adição e remoção de livros, se necessário.