UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA APLICADA

INF01151 – SISTEMAS OPERACIONAIS II N
SEMESTRE 2020/2
ATIVIDADE DE PROGRAMAÇÃO GUIADA: WEB SOCKETS

OBJETIVO DA AULA PRÁTICA:

Esta atividade de Programação Guiada tem por objetivo demonstrar o funcionamento de WebSockets. Para isso, uma aplicação simples que realiza a troca de mensagens através do protocolo deverá ser desenvolvida. Nesta aula será utilizada a API Java para desenvolvimento de aplicações WebSocket Java (JSR 356) 1.1.

Tutoriais de apoio: https://jersey.java.net
https://docs.oracle.com/cd/E19776-01/820-4867/6nga7f5ml/index.html
https://docs.oracle.com/javaee/6/tutorial/doc/giepu.html

INSTALAÇÃO E CONFIGURAÇÃO DE DEPENDENCIAS:

Para essa atividade de programação guiada, vamos precisar dos softwares abaixo. Ou, se preferir, pode usar a VM preparada para a atividade (vide abaixo).

- Apache Maven. Para instruções de instalação, acesse https://maven.apache.org/install.html . No Ubuntu Linux, você pode instalar o Apache Maven com o seguinte comando:

```
apt-get install maven
```

Com o comando acima, o Apache Maven e as principais dependências serão instaladas e configuradas automaticamente.

- Java EE Development Kit. Sugere JDK 8 ou 11. Para instruções de instalação, acesse http://jdk.java.net/ . No Ubuntu Linux, você pode instalar o JDK com o comando:

```
apt-get install openjdk-8-jdk-headless
```

ou

```
apt-get install openjdk-11-jdk-headless
```

Dica: Após as instalações, pode ser necessário deslogar e logar novamente no ambiente do sistema operacional, para que as variáveis de ambiente dos recursos instalados sejam exportadas para o usuário logado atualmente.

Dica: Para facilitar o trabalho, disponibilizamos uma imagem de VM com todo o ambiente de desenvolvimento e de testes pré-configurado, pronto para usar. O endereço da VM é: https://acdc.inf.ufrgs.br/ubuntu16.ova . Instruções para importar máquinas virtuais com o VirtualBox podem ser encontradas em https://docs.oracle.com/en/virtualization/virtualbox/6.0/user/ovf.html .

UMA APLICAÇÃO DE CHAT COM WEBSOCKETS:

Você deverá implementar *endpoints* para comunicação através do protocolo WebSocket. O objetivo é observar o envio de mensagens de um cliente (neste caso o navegador) para o servidor implementado em Java. O servidor fornecerá os métodos necessários para a comunicação bem como um caminho para o recebimento das mensagens (`/chat`). Os passos 1 à 5 compreendem a implementação da aplicação que desempenhará o papel de Servidor. A aplicação cliente está definida no diretório `webapp` fornecido na implementação-base.

Passos:

1. **Faça o download da implementação-base:** O arquivo **PG1-WebSockets.zip** está disponível no Moodle da disciplina, descompacte-o. no diretório **/src/main/java** devem estar localizadas as classes que serão exigidas nos próximos passos.

2. Analise a Classe `ChatServer`. Um passo importante para a comunicação através de WebSockets é a definição de uma URL (ou *template URI*) que o *endpoint* definido pela classe irá disponibilizar, além de outras definições, como por exemplo *encoders* usados para enviar mensagens. Realize a anotação da classe para que seja disponibilizado o caminho "/chat". E não esqueça do import de `javax.websocket.server.ServerEndpoint`;

```
@ServerEndpoint("/chat")
public class ChatServer{
        ...
}
```

3. **Persistência de informações das Sessões:** A comunicação entre cliente e servidor é realizada através de sessões. Inclua import `javax.websocket.Session` para realizar o armazenamento de informações sobre as sessões em uma estrutura de dados de sua escolha (Set, List, …), para que as sessões possam ser visíveis pelos métodos a serem implementados.

   Dica: Para que as mensagens enviadas por um cliente sejam recebidas por todos os outros clientes, a estrutura de dados que guardará as sessões na classe `ChatServer` deve ser definida como um atributo estático da classe. Para isso, use a palavra-chave `static` ao declarar a estrutura de dados. Essa estrutura de dados poderá ser declarada como um tipo privado (`private`).

4. **Implementação dos métodos do ServerEndpoint:** Esses métodos são responsáveis pelas principais funcionalidades de um WebSocket. Em cada método, informações sobre a sessão deverão ser mostradas através de novos registros de log (importe a `java.util.logging.Logger` para isso; não use métodos de outras classes para esse fim). Dos métodos e suas devidas anotações e funcionalidades:

   - Método anotado como @onOpen: Método que manipula o recebimento de conexões dos *peers*. Recebe como parâmetro um objeto `Session` e o armazena para que os outros métodos possam ter acesso. Toda vez que uma nova conexão for estabelecida, crie um novo registro de log indicando o identificador único da sessão atual, utilizando o método **getId()** para isso.

   Lembrando que posteriormente você pode explorar mais métodos disponibilizados pela API para mostrar mais informações sobre as sessões. Os métodos para isso você encontra em:

   https://docs.oracle.com/javaee/7/api/javax/websocket/Session.html.

   - Método anotado como @onClose: Método que é chamado quando uma sessão WebSocket está sendo fechada. Receba como parâmetro um objeto Session e crie um novo registro de log informando que a conexão identificada com o ID será fechada. Após isso, remova a sessão indicada da estrutura de dados que você definiu previamente.

   Como alternativa você também pode receber como parâmetro a razão do fechamento da conexão através do objeto CloseReason, mostrando o código do fechamento ou a string que define a razão. Os métodos para isso você encontra em:

   https://docs.oracle.com/javaee/7/api/javax/websocket/CloseReason.html

   - Método anotado como @onMessage: Método que é chamado para lidar com o recebimento de mensagens WebSocket. A princípio, implemente um método que recebe mensagens em formato de string e replique-as para os outros peers conectados. Essa

funcionalidade pode ser implementada através do acesso a estrutura definida para abrigar as sessões estabelecidas e a invocação do método **getBasicRemote()** para cada uma delas, utilizando o método **sendText()** para encaminhar as mensagens. Ex:

```
session.getBasicRemote().sendText(msg);
```

Fique atento pois será necessário realizar o tratamento, através de `try/catch`, da exceção `IOException`. Para finalizar, não esqueça de importar:

```
import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
```

Não esqueça que você deve importar também as demais classes que você usa no programa, por exemplo, a estrutura de dados de sua escolha (Set, List, ...) para realizar o armazenamento de informações sobre as sessões, implementada no passo 3.

A implementação dos métodos acima é a base para o funcionamento de uma simples aplicação de chat em texto sobre WebSockets. A parte do cliente está definida em webapp/websockets.js, contando com a definição da interface de chat (index.html) e a implementação dos métodos de WebSockets em Javascript.

5. **Observe as definições dos métodos-cliente:** No arquivo websockets.js temos a criação de um canal websocket em ws://localhost:8080/chat e as implementações dos métodos de abertura, envio e fechamento. Basicamente, cada vez que uma guia do navegador da aplicação for aberta, uma conexão WebSocket é estabelecida. A aplicação permite que o cliente crie e envie mensagem para outros clientes conectados.

6. **Disparando a aplicação:** A partir do diretório pai da aplicação, execute o seguinte comando:

```
mvn jetty:run
```

A aplicação estará disponível em `localhost:8080`.

7. **Testando a aplicação:** Após a instanciação da aplicação, observe o log quando um cliente é conectado. Abra algumas guias no navegador e realize o envio de mensagens. A aplicação deve realizar a replicação de mensagens para todos os clientes conectados.

Yay! Se você chegou até aqui, parabéns! Faça o envio do relatório do programa no Moodle, e seu objetivo na atividade de programação guiada estará cumprido.

Mas espere!!! Ainda há uma serie de aspectos interessantes que podem ser explorados. Agora que você finalizou essa aula prática, você pode explorar os passos adicionais definidos no item 4, como por exemplo forçar o fechamento da conexão `websocket` e observar os retornos através de `CloseReason`. Outra alternativa é explorar as configurações do ServerEndpoint para observar dados como o handshake de abertura da conexão. Para isso, utilize como auxílio o tutorial abaixo:

https://docs.oracle.com/javaee/7/tutorial/websocket010.htm

Você pode também utilizar as ferramentas de desenvolvimento disponíveis em seu navegador para visualizar as requisições marcadas como WS durante o uso da aplicação.