WebSocket

Celem ćwiczenia jest zapoznanie z technologią WebSocket do dwukierunkowej komunikacji między serwerem i przeglądarką. Strona serwerowa w ćwiczeniu będzie implementowana z wykorzystaniem Java API for WebSocket.

Ćwiczenia zostały przygotowane dla środowiska NetBeans 13 i zintegrowanego z NetBeans serwera Payara.

1. Uruchom NetBeans i utwórz nowy projekt. Wybierz File→New Project... i z listy kategorii wybierz Java with Maven, a z listy projektów wybierz Web Application. Kliknij przycisk Next >. Jako nazwę projektu podaj **WebSocket**, wyczyść pole z nazwą pakietu i kliknij przycisk Next >. Wybierz wersję Java EE **Jakarta EE 8 Web** i upewnij się, że jako serwer aplikacji wybrany jest serwer Payara (jeśli nie jest dostępny do wyboru, to kliknij Add... i pobierz oraz zainstaluj najnowszą dostępną wersję). Kliknij przycisk Finish.

   Otwórz plik konfiguracyjny projektu Mavena `pom.xml` i odszukaj wersję maven-war-plugin. Jeśli podana jest wcześniejsza niż 3.3.2, to zmień ją na 3.3.2 i zapisz zmiany.

2. Utwórz WebSocket Server Endpoint:

   a) Dodaj do projektu nową klasę VotingWebSocketServer umieszczając ją w pakiecie lab.websocket.
   b) Oznacz klasę adnotacją @ServerEndpoint("/actions"). Zaimportuj użytą adnotację.
   c) Dodaj w klasie poniższe metody cyklu życia WebSocket. Zaimportuj wykorzystane adnotacje oraz interfejs Session z pakietu javax.websocket. Obiekt Session reprezentuje konwersację między dwoma końcówkami WebSocket.

```
@OnOpen
public void onOpen(Session session) {
}

@OnClose
public void onClose(Session session) {
}

@OnError
public void onError(Throwable error) {
}

@OnMessage
public void onMessage(String message, Session session) {
}
```

   d) Oznacz klasę adnotacją CDI @ApplicationScoped.

3. Utwórz klasę do obsługi sesji WebSocket:

   a) Dodaj do projektu nową klasę VotingSessionHandler umieszczając ją w pakiecie lab.websocket.
   b) Oznacz klasę adnotacją CDI @ApplicationScoped.
   c) Zadeklaruj w klasie kolekcję do przechowywania sesji klientów i zmienne do pamiętania liczby głosów oddanych kandydatów w wyborach.

```
private final Set<Session> sessions = new HashSet<>();

private int votesHillary = 0;
private int votesDonald = 0;
```

   d) Zaimportuj wykorzystaną klasę i interfejs zbioru.
   e) Dodaj do klasy metody do dodawania i usuwania sesji klientów.

```
public void addSession(Session session) {
    sessions.add(session);
}

public void removeSession(Session session) {
    sessions.remove(session);
}
```

   f) Dodaj do klasy metody do rejestracji głosów na kandydatów.

```
public void voteForHillary() {
    votesHillary++;
}

public void voteForDonald() {
    votesDonald++;
}
```

   g) Dodaj do klasy prywatne, pomocnicze metody do wysyłania danych wybranego lub wszystkich klientów oraz metodę zwracającą aktualne wyniki głosowania w formacie JSON. Zaimportuj wykorzystywane klasy związane z obsługą formatu JSON w Javie.

```
private JsonObject createVotingResultsMessage() {
    JsonProvider provider = JsonProvider.provider();
    JsonObject message = provider.createObjectBuilder()
            .add("votesHillary", votesHillary)
            .add("votesDonald", votesDonald)
            .build();
    return message;
}

private void sendToAllConnectedSessions(JsonObject message) {
    for (Session session : sessions) {
        sendToSession(session, message);
    }
}

private void sendToSession(Session session, JsonObject message) {
    try {
        session.getBasicRemote().sendText(message.toString());
    } catch (IOException ex) {
        sessions.remove(session);
    }
}
```

h) Na końcu ciała metody do dodawania nowej sesji, dodaj poniższą instrukcję, która wyśle do nowej sesji aktualne wyniki głosowania z wykorzystaniem zdefiniowanych wcześniej metod pomocniczych.

```
sendToSession(session, createVotingResultsMessage());
```

i) Dodaj w klasie metodę, która wyśle do wszystkich aktywnych sesji aktualne wyniki głosowania z wykorzystaniem zdefiniowanych wcześniej metod pomocniczych.

```
    public void pushResults() {
        sendToAllConnectedSessions(createVotingResultsMessage());
    }
```

4. Wróć do edycji klasy VotingWebSocketServer i zaimplementuj w niej obsługę zdarzeń generowanych przez WebSocket:

a) Wstrzyknij w klasie referencję do obiektu klasy obsługującej sesje WebSocket.

```
    @Inject
    private VotingSessionHandler sessionHandler;
```

b) W metodzie oznaczonej @OnOpen dodaj instrukcję dodania nowej sesji.

```
sessionHandler.addSession(session);
```

c) W metodzie oznaczonej @OnMessage dodaj poniższy kod rejestrujący głos, który przyszedł w wiadomości z przeglądarki, a następnie przesyłający aktualne wyniki do wszystkich klientów.

```
        if ("Hillary".equals(message))
            sessionHandler.voteForHillary();
        if ("Donald".equals(message))
            sessionHandler.voteForDonald();

        sessionHandler.pushResults();
```

d) W metodzie oznaczonej @OnClose dodaj poniższy kod usuwający sesję ze zbioru aktywnych sesji.

```
sessionHandler.removeSession(session);
```

e) W metodzie oznaczonej @OnError należałoby zaimplementować obsługę błędów (np. zapis informacji o błędzie do logu). W naszej aplikacji dla uproszczenia pominiemy ten krok.

5. Zaimplementuj stronę HTML do oddawania głosów i prezentacji aktualnych wyników głosowania w przeglądarce:

a) Przejdź do edycji strony index.html.

b) Podmień kod strony na poniższy:

```
<!DOCTYPE html>
<html>
    <head>
        <title>WebSocket OnLine Voting System</title>
        <script>
          var socket = new WebSocket("ws://localhost:8080/WebSocket/actions");
          socket.onmessage = onMessage;

          function onMessage(event) {
            var results = JSON.parse(event.data);
            ...
          }

          function vote(name) {
            socket.send(name);
          }
        </script>
    </head>
    <body>
        <h4>Current voting results</h4>
        Hillary: <span id="hillary">0</span> Donald: <span id="donald">0</span>
        <p>
        <form id="addVoteForm">
          <input type="button" value="Vote for Hillary!" onclick="vote('Hillary')">
          <input type="button" value="Vote for Donald!" onclick="vote('Donald')">
        </form>
    </body>
</html>
```

c) Samodzielnie uzupełnij brakujący kod w funkcji onMessage, tak aby po odebraniu wiadomości z serwera uaktualnił się wynik głosowania na stronie.

6. Uruchom aplikację. Następnie otwórz dodatkowe okna przeglądarek i wpisz w nich ręcznie adres strony klienta. Przetestuj zachowanie się systemu po głosowaniu na kandydatów z różnych okien przeglądarki, zwracając uwagę na propagację aktualnych wyników głosowania do wszystkich aktywnych klientów.