


# Java Enterprise Edition



# Cześć

Łukasz Chrzanowski

 /lukasz-chrzanowski-dev

 /lukasz4coders

[lukasz@chrzanowski.co](mailto:lukasz@chrzanowski.co)

# Materiały do zajęć

<https://github.com/infoshareacademy/jjddr1-materialy-jee>

# Server-Side Events

Obsługa eventów (SSE)

# SSE: Jak to działa?

- Klient nawiązuje połączenie z serwerem
- Serwer wysyła eventy do klienta
- Klient interpretuje eventy, dokonuje odpowiednich akcji w serwisie
- Serwer zamyka konwersację jeśli kończy wysyłanie eventów

# Servlet Producer

```
resp.setContentType("text/event-stream");
```

```
resp.setCharacterEncoding("UTF-8");
```

```
resp.getWriter().write("data: " + userService.findAllUsers().size() + "\n\n");
```

# JavaScript Consumer

```
var eventSource = new EventSource("/push-new-user");  
  
eventSource.onmessage = function (event) {  
    ...  
    processing event.data  
    ...  
};
```



## Zadanie: EventSource

Utwórz servlet **UserCreationNotifier**, który będzie generował eventy o aktualnej liczbie użytkowników w bazie danych (pamięci).

Utwórz nowy plik **users-list-notification.html**, w którym zaimplementujesz konsumenta eventów i wyświetlisz informację na tym widoku (lub zalogujesz wykorzystując *console.log*)



# Zarządzanie czasem eventów

To programista decyduje jak często eventy będą pushowane do klientów. Do konfiguracji czasu służy parametr **retry** definiowany w odpowiedzi.

Czas podajemy w milisekundach.

```
resp.getWriter().write("retry: 20000\n");
```

# Zadanie: EventSource Timing



Spraw aby eventy były wysyłane do 10 sekund

# Java EE @Remote

Zdalne wywoływanie EJB

# @Remote: idea

Idea interfejsów zdalnych to możliwość wykonywania metod na naszym EJB spoza naszego modułu.



## Zadanie: @Remote

Stwórz **interfejs** zdalny dla Repozytorium Użytkowników:  
**UsersRepositoryDaoRemote**

**Interfejs** powinien zawierać tylko jedną metodę, zwracającą listę imion użytkowników: **getUserNames**

**UsersRepositoryDaoBean** powinien od tej pory implementować dwa **interfejsy: lokalny i zdalny**. Nie zapomnij o zapewnieniu implementacji zdefiniowanej metody.



## Zadanie: Standalone

Utwórz nowy projekt Maven (nie zamykaj dotychczasowego projektu JEE), o nazwie **artifactID:users-engine-standalone**

W celu utworzenia projektu możesz wykorzystać archetyp:  
**org.apache.maven.archetypes:maven-archetype-quickstart**

Zbuduj, uruchom, sprawdź czy aplikacja konsolowa działa poprawnie

## @Remote: library

Aby było możliwe wywołanie zdalne interfejsu należy stworzyć bibliotekę zawierającą zdalny interfejs. Do tego celu tworzymy bibliotekę z interfejsem korzystając z pluginu: maven-ejb-plugin.

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-ejb-plugin</artifactId>
  <version>3.0.1</version>
  <configuration>
    <ejbVersion>3.2</ejbVersion>
    <generateClient>true</generateClient>
    <clientIncludes>
      <clientInclude>
        com/isa/searchengine/dao/UsersRepositoryDaoRemote.class
      </clientInclude>
    </clientIncludes>
  </configuration>
</plugin>
```



## Zadanie: EJB library

Stwórz bibliotekę, która będzie zawierała nowo utworzony interfejs komunikacji zdalnej.

Zbuduj projekt wykorzystując dotychczasowe znane komendy:  
**mvn clean package**

Uruchom wtyczkę **maven-ejb-plugin**  
**mvn ejb:ejb**

Zainstaluj bibliotekę w lokalnym repozytorium:  
**mvn install:install-file -Dfile=users-engine-client.jar  
-DgroupId=com.isa -DartifactId=users-engine-client  
-Dpackaging=jar -Dversion=1.0**



# Zadanie: standalone dependency



Załącz nową bibliotekę users-engine-client jako zależność mavenową w nowym projekcie standalone

```
<dependency>
  <groupId>com.isa</groupId>
  <artifactId>users-engine-client</artifactId>
  <version>1.0</version>
</dependency>
```

# Maven plugin: assembly

Plugin maven-assembly-plugin pozwala na budowanie naszych projektów tak aby artefakt zawierał w sobie wszystkie zależności, moduły, dokumentację, itp..

# Maven plugin: assembly

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <version>3.1.0</version>
  <configuration>
    <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
    <archive>
      <manifest>
        <addClasspath>>true</addClasspath>
        <classpathPrefix>libs/</classpathPrefix>
        <mainClass>
          com.isa.App
        </mainClass>
      </manifest>
    </archive>
  </configuration>
  <executions>
    <execution>
      <id>make-assembly</id>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

# Zadanie: standalone dependency



Zbuduj aplikację w taki sposób aby zawierała wewnątrz wszystkie załączone zależności mavenowe.

Do tego celu służy kolejny plugin:

**Maven-assembly-plugin.**

Zbuduj aplikację, zajrzyj do katalogu target

Sprawdź czy wygenerowany artefakt zawiera zależności:

```
jar -tf <name.jar> | grep „isa"
```

# JNDI

Java Naming and Directory Interface

# JNDI: charakterystyka

JNDI nie znajduje się w specyfikacji JEE ale jest mocno wykorzystywane dlatego kontenery aplikacyjne implementują to API

JNDI służy do wyszukiwania obiektów po ich nazwach

Wyszukiwanie odbywa się poprzez klasę InitialContext i metodę `lookup(String nazwa)`



## Zadanie: obsługa klienta

Przygotuj wymagane zależności w aplikacji standalone. Pamiętaj, że będziemy wykorzystywali implementację dostarczoną przez Wildfly

```
<dependency>
  <groupId>org.wildfly</groupId>
  <artifactId>wildfly-client-all</artifactId>
  <version>17.0.1.Final</version>
</dependency>
```



# Zadanie: konfiguracja klienta

Przygotuj konfigurację, która będzie spełniała wymagania swojej instancji serwera Wildfly

```
Hashtable<String, String> properties = new Hashtable<String, String>();  
properties.put(Context.INITIAL_CONTEXT_FACTORY,  
"org.jboss.naming.remote.client.InitialContextFactory");  
properties.put("jboss.naming.client.ejb.context", "true");  
properties.put(Context.PROVIDER_URL, "http-remoting://localhost:8080");  
properties.put(Context.SECURITY_PRINCIPAL, "<user>");  
properties.put(Context.SECURITY_CREDENTIALS, "<password>");  
Context context = new IldFlyInitialContextFactory().getInitialContext(properties);
```



# Zadanie: wywołanie zdalnej metody

Zaprezentuj w konsoli listę imion użytkowników wykorzystując nowy kontekst.

JNDI NAME znajdziemy w konsolce WildFly: **Runtime -> Standalone Server -> Subsystems -> JNDI View -> java:jboss/exported**

```
UsersRepositoryDaoRemote generator =  
(UsersRepositoryDaoRemote) context.lookup(„<JNDI NAME>”);
```



# Dzięki