Architektury systemów internetowych

Paweł Czarnul
pczarnul@eti.pg.gda.pl
http://fox.eti.pg.gda.pl/~pczarnul

Computer Architecture Department
Technical University of Gdansk, Poland

Plan wykładu

- Technologie, narzędzia i platformy programowania w Internecie
  - Wykład omawia podstawy teoretyczne na bazie najbardziej popularnych architektur systemów wykorzystywanych w Internecie oraz przykładów praktycznych wybranych technologii –
    - przykład wielowarstwowej aplikacji w Internecie bazującej na technologii serwletów z konfiguracją serwera Tomcat, jak również konstrukcją relacyjnej bazy danych
    - Usługi sieciowe (ang. Web Services) I ich integracja, protokół SOAP, rejestr UDDI, język opisu WSDL

Plan wykładu

- Programowanie w Internecie na przykładach w ramach wykładu
  - Serwlety:
    - Projekt książka gości (serwlety zapisujące dane do bazy danych i prezentujące wcześniej zapisane dane użytkownikowi), wykorzystanie serwera bazy danych MySQL do projektu
  - Usługi sieciowe:
    - Analiza i omówienie prostej usługi sieciowej, wdrożenie na serwerze, analiza komunikacji pomiędzy klientem a usługą wraz z omówieniem SOAP, odwzorowanie na HTTP

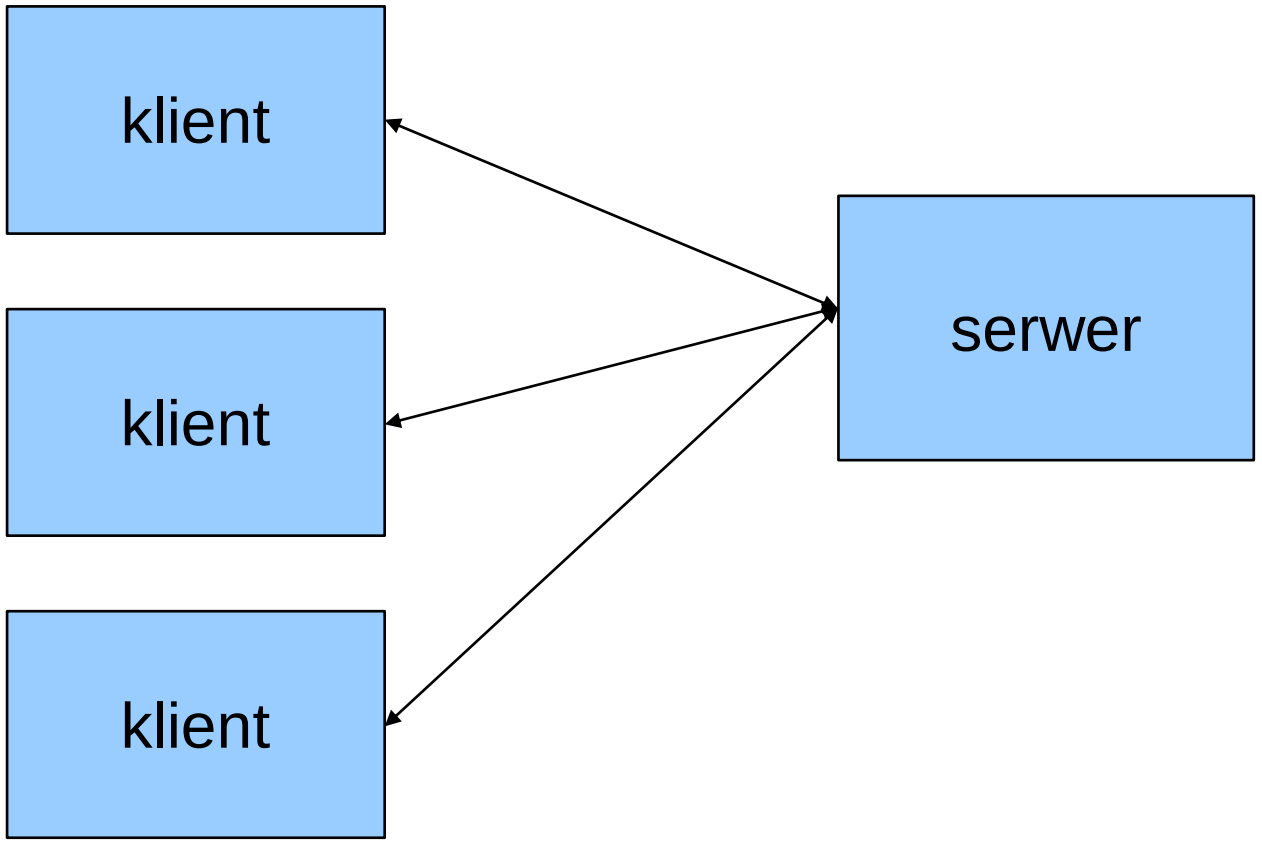
Architektury systemów internetowych/rozproszonych

Rozwój

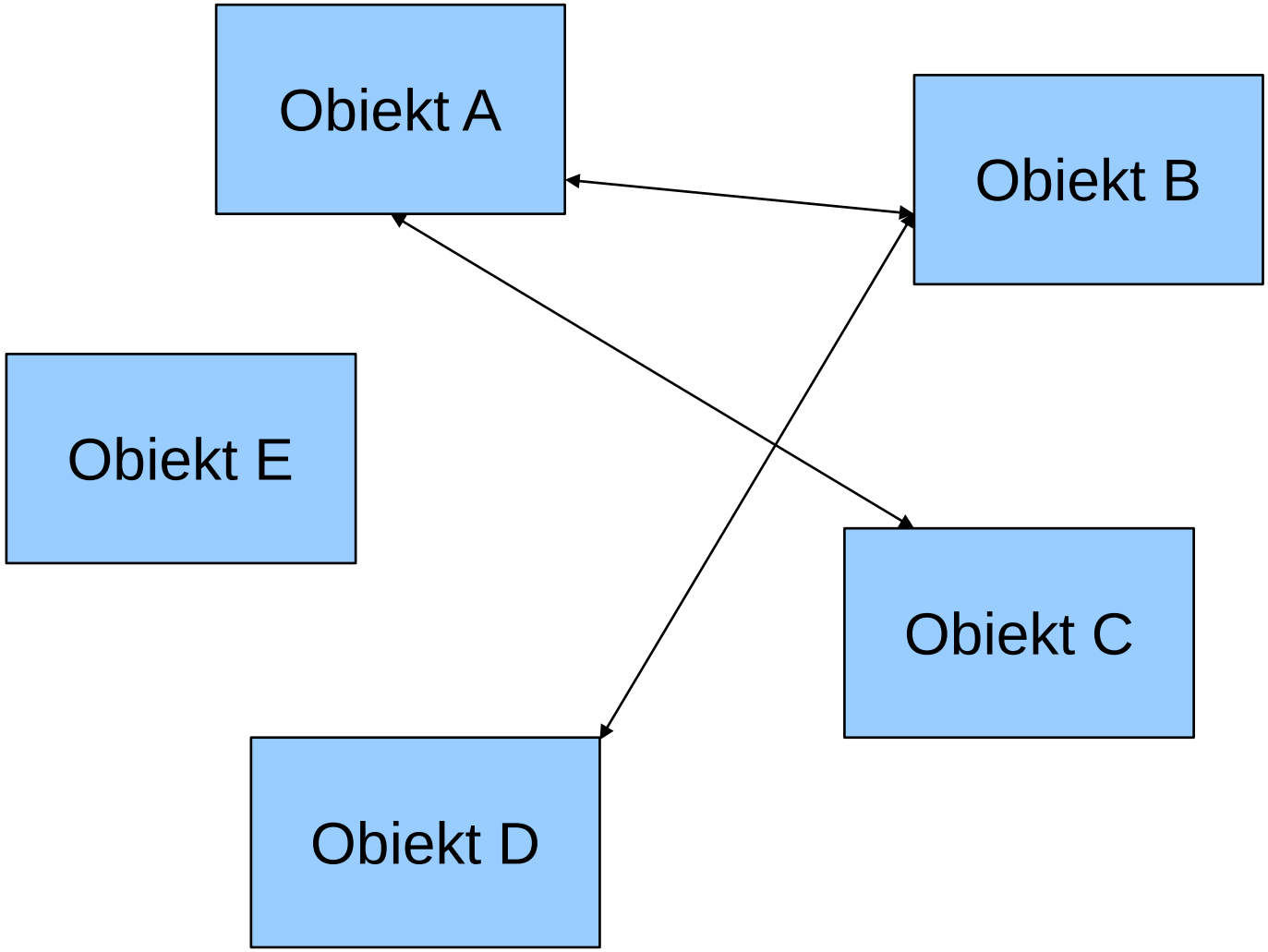
Architektury systemów rozproszonych

1. High Performance Computing/przetwarzanie klastrowe
   - Symulacje na klastrach/superkomputerach: elektromagnetyzm, symulacje zmian pogody, symulacje medyczne
   - PVM, MPI
2. Klient-serwer
   - Model dla aplikacji biznesowych: klient woła funkcje dostępne na serwerze
   - RPC, sockety, Java RMI
3. Rozproszone systemy obiektowe
   - Klient serwer na bazie obiektów
   - Usługi zdefiniowane i wspierane przez środowisko rozproszone/obiekty
   - Np.. CORBA (blokowanie przez firewalle?)
4. Wielowarstwowe aplikacje internetowe
   - Wyróżnione i wydzielone warstwy: użytkownika, prezentacji, logiki biznesowej, systemów baz danych np..przeglądarka internetowa/Tomcat/MySQL
   - Łatwe do wykorzystania technologie jak PHP, serwlety, strony JSP
   - Komponentowy rozwój oprogramowania np. J2EE, Microsoft .NET

Klient-serwer

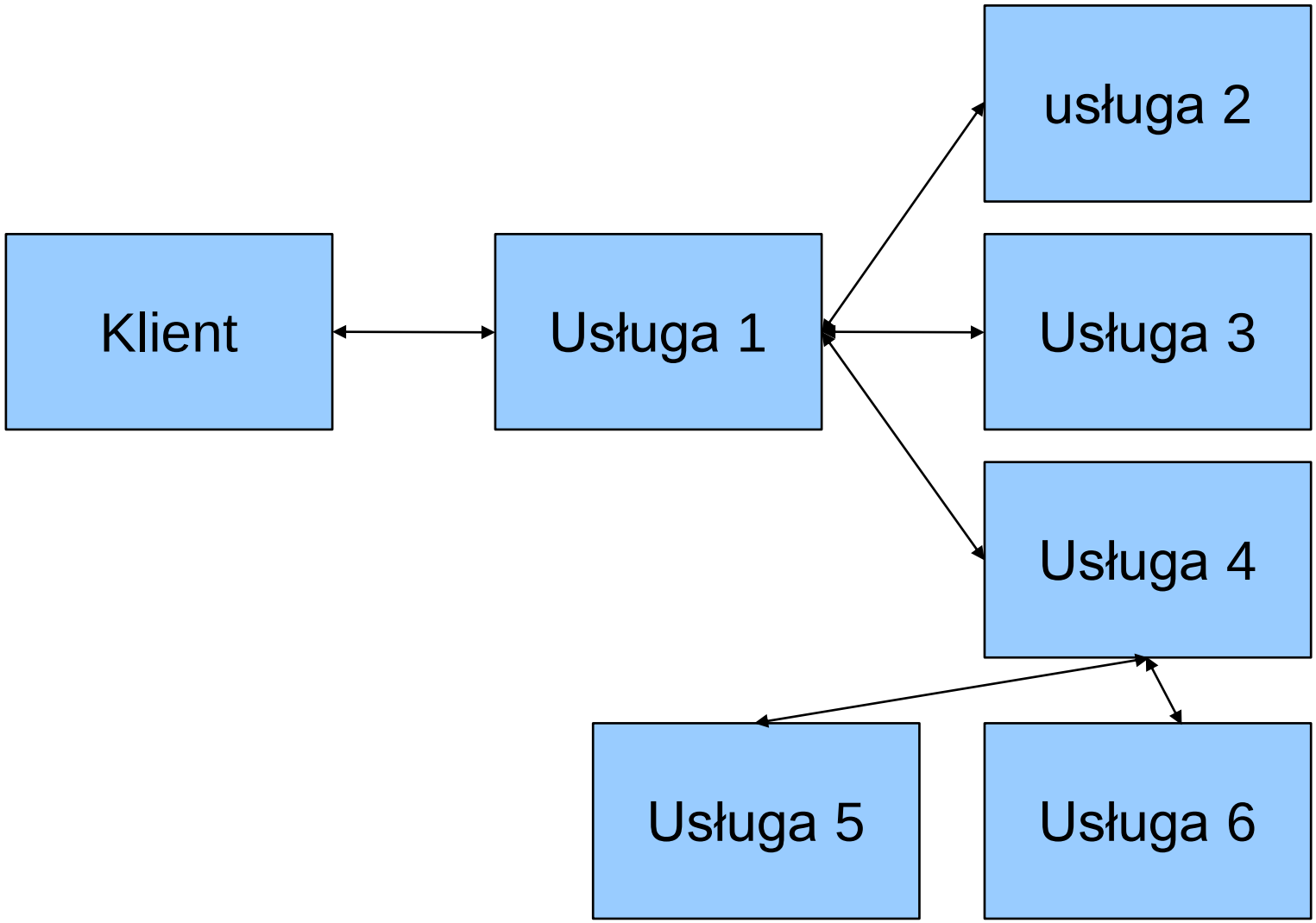


Rozproszone obiekty

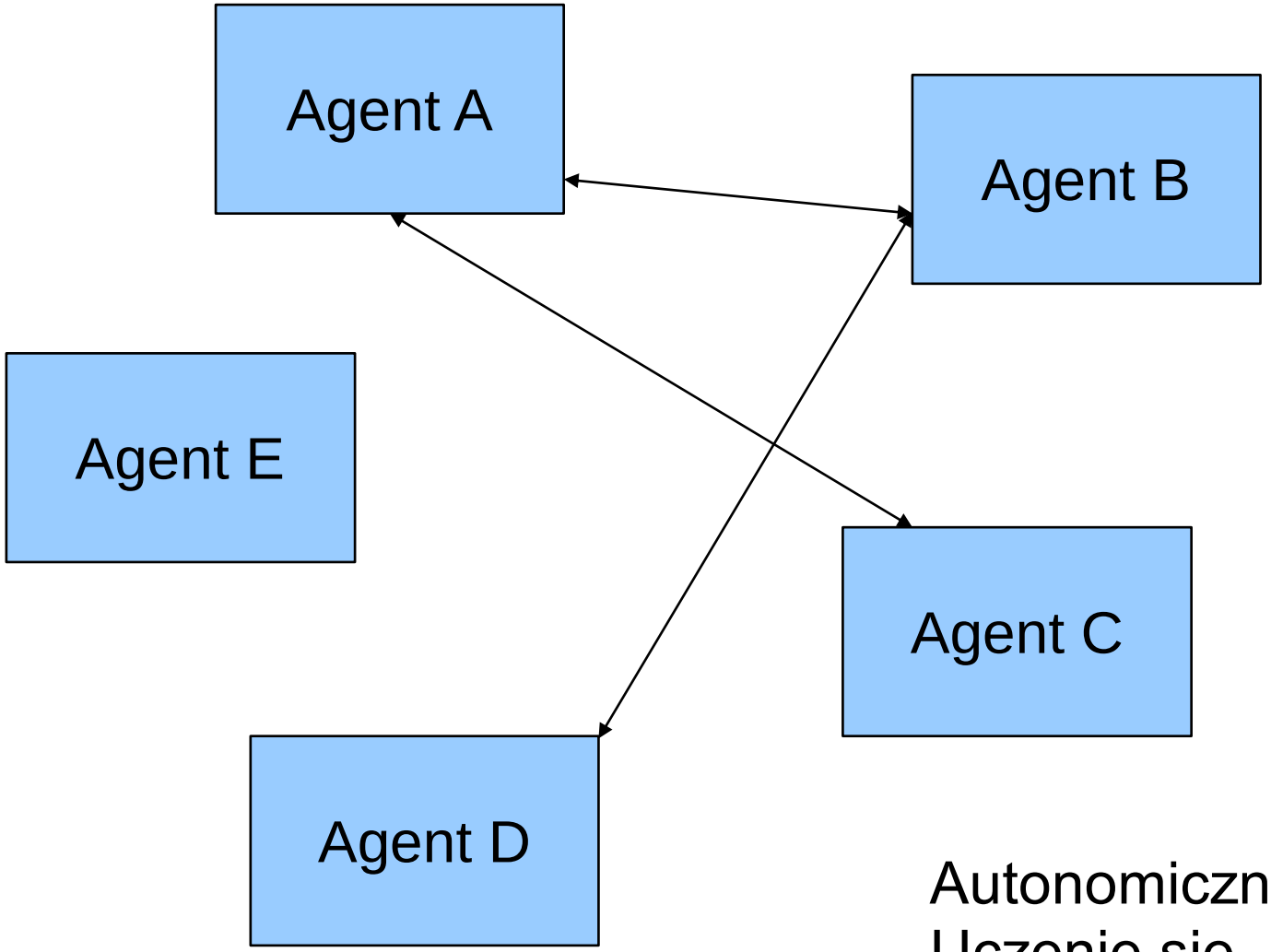


Systemy wielowarstwowe

Systemy zorientowane na usługi

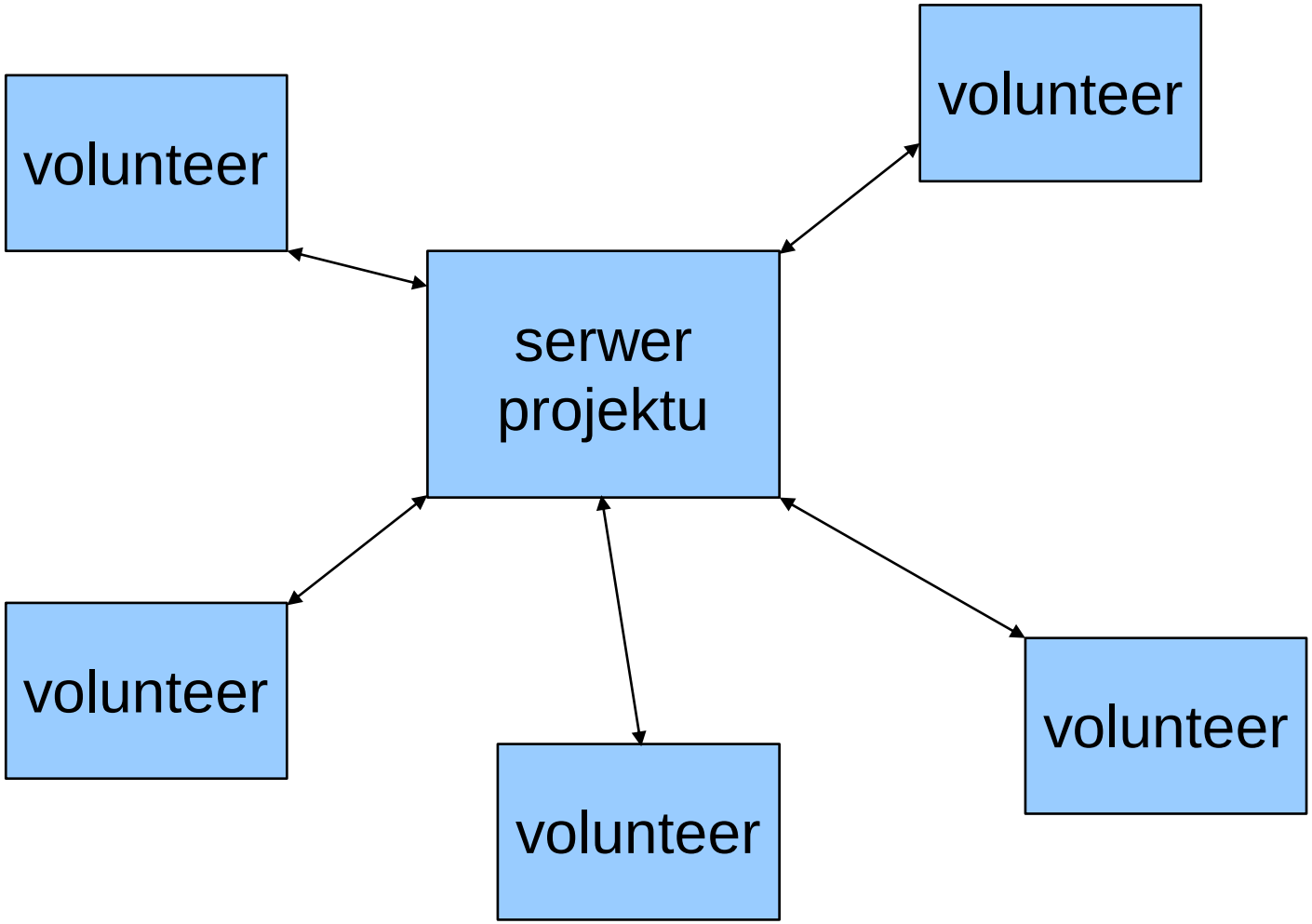


Systemy agentowe

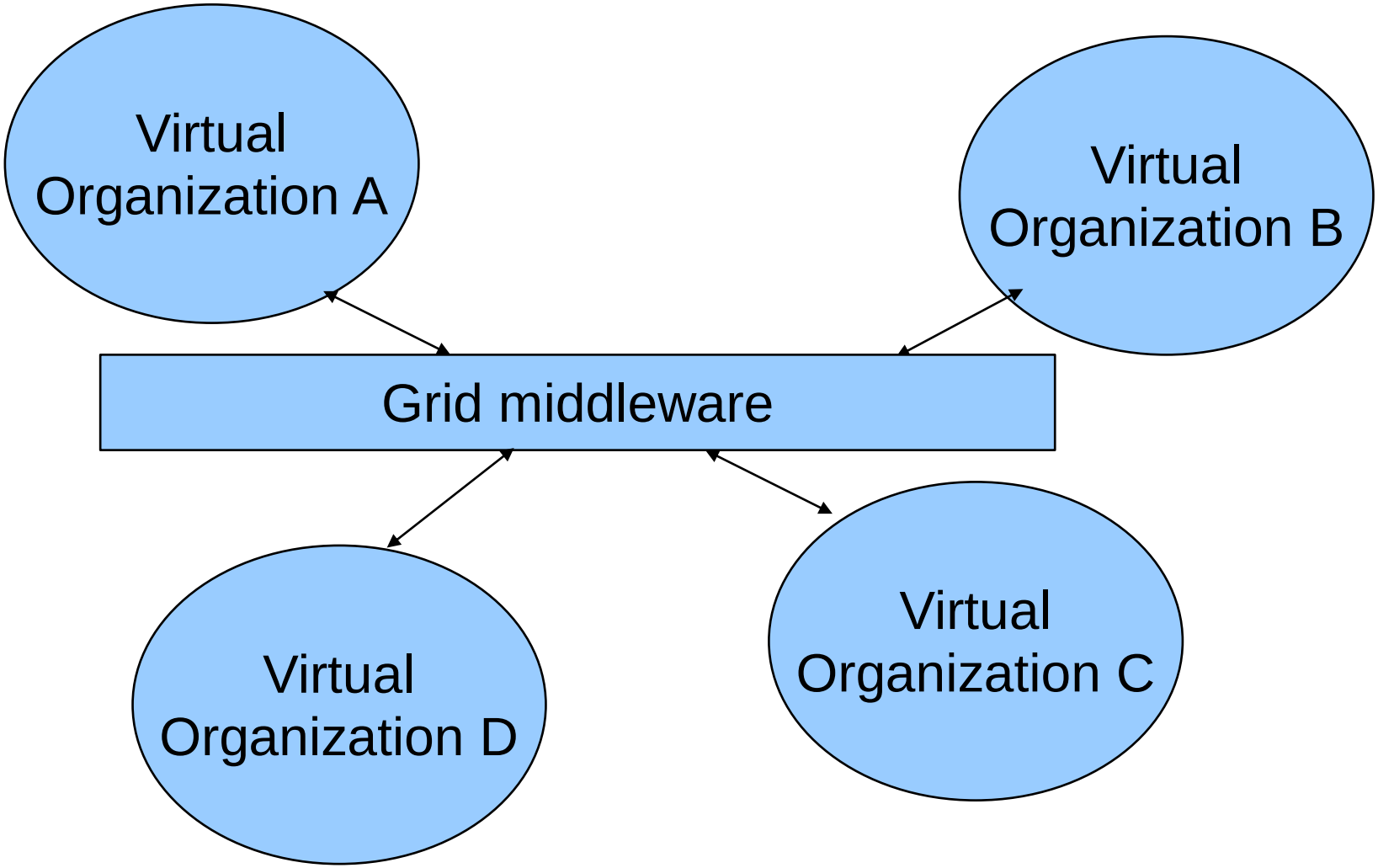


Autonomiczność
Uczenie się
ontologie

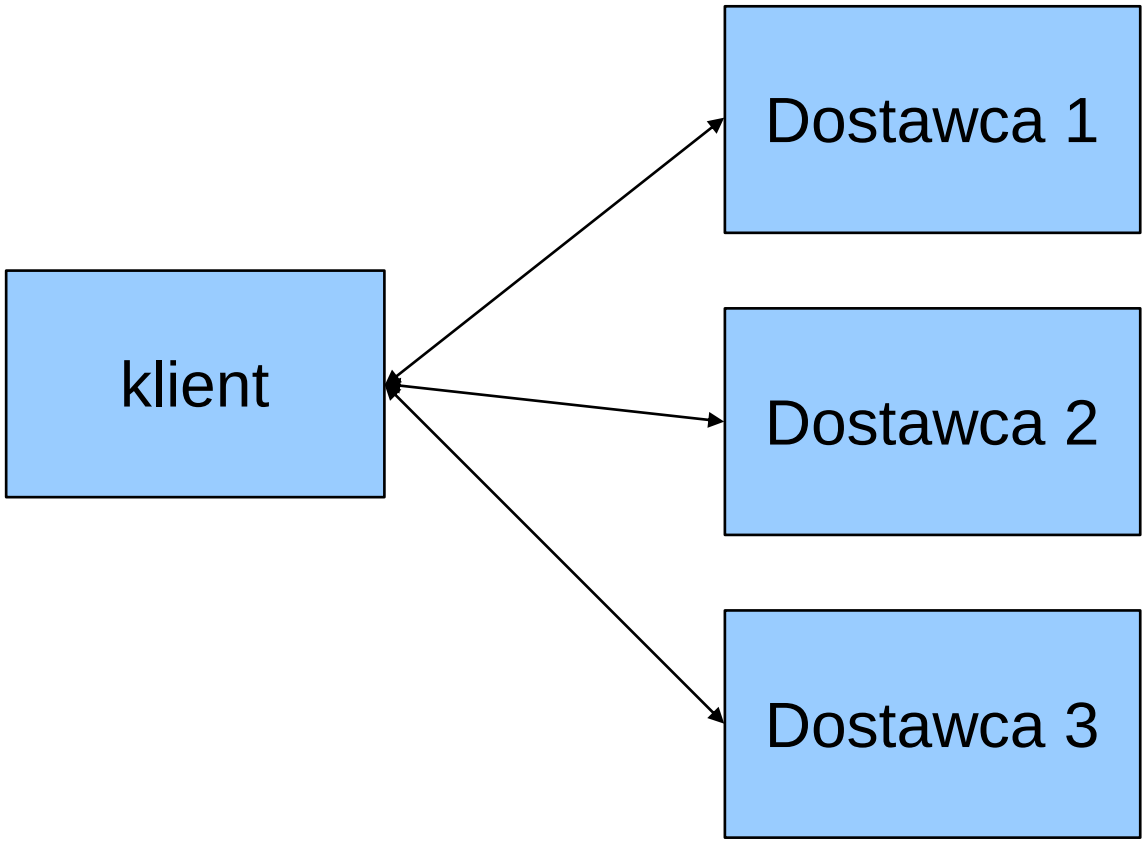
Systemy typu volunteer computing



Systemy typu grid



Systemy typu cloud



Architektury systemów rozproszonych

5. Systemy peer-to-peer
   - Współpraca równorzędnych jednostek w sieci rozproszonej, brak wyróżnionych serwerów, replikacja

6. Systemy agentowe
   - Agenty: autonomiczność
   - Użyteczne gdy można wysłać agenty do wykonania pewnej zdalnej pracy – gdzie mogą się ze sobą komunikować lokalnie, gdy brak łączności ze zdalnymi obiektami a agenty mogą się komunikować lokalnie
   - Agenty a klient-serwer - dyskusja

7. SOA – Service Oriented Computing

- Wołanie zdalnych usług (loosely-coupled) z wykorzystaniem jasno zdefiniowanych standardów
- Szereg standardów bazowych dla bazowej implementacji SOA – Web Services: WSDL, SOAP, XML,
- Web Services nie są blokowane przez firewalle
- UDDI jako rejestr informacyjny
- Szereg standardów związanych z integracją usług - …
- Usługa może wywoływać inne usługi, budowa złożonych scenariuszy składających się z usług – tzw. Workflows (business, scientific)
- Przenośność: działanie na różnym sprzęcie, możliwość implementacji zarówno serwera jak i klienta w różnych językach programowania, szereg technologii do wdrożenia usług sieciowych np.. Apache+AXIS, Microsoft .NET, inne

Architektury systemów rozproszonych

8. Przetwarzanie w przestrzeniach inteligentnych
   - Urządzenia I sensory zaintegrowane w jeden system

9. Systemy gridowe
   - Kontrolowane współdzielenie zasobów – integracja rozproszonych geograficznie zasobów w platformę współpracy
   - Identyfikacja, uwierzytelnianie, zdalne uruchamianie zadań (ukrywanie LSF/PBS), zarządzanie kontami użytkowników, OGSA

10. Przetwarzanie typu volunteer computing
- Duża moc obliczeniowa, problemy wiarygodności I prywatności

11. Przetwarzanie typu cloud
- Udostępnianie:
  - Infrastruktury (IaaS – Infrastructure as a Service)
  - Oprogramowania (SaaS – Software as a Service)
  - Kompletnej platformy (PaaS – Platform as a Service)

Architektury systemów rozproszonych

1. Urządzenia mobilne w Internecie – J2ME
   - Małe wymagania
   - CLDC, CDC
   - MIDP
   - Funcjonalność:
     - Interfejs użytkownika
     - Połączenia sieciowe (HTTP/S, usługi sieciowe)
     - wielowątkowość
     - Multimedia
     - Lokalizacja

Wydajność komunikacji, założenia co do wydajności, zawodności sieci

- Komunikacja – parametry:
  - Przepustowość (sprawdzenie w praktyce)
  - tzw. Startup-time
  - Niezawodność sieci
  - komunikacja strumieniowa, pakietowa etc.
  - Nakładanie obliczeń i komunikacji

Wydajność komunikacji, założenia co do wydajności, zawodności sieci

- Komunikacja
  - Protokoły wykorzystywane w Internecie, w szczególności HTTP(S)
  - Komunikacja żądanie-odpowiedź
  - sesja
  - ciasteczka
  - Protokoły a wydajność komunikacji
  - Zabezpieczenia komunikacji

Podstawy teoretyczne oraz problemy w programowaniu w Internecie

- Łatwość utrzymania kodu
- Potrzeba konfiguracji
- Gruby a cienki klient
- Problemy synchronizacji
- Problemy współbieżności, wielowątkowość, równoległość

Podstawy teoretyczne oraz problemy w programowaniu w Internecie

- Zabezpieczanie systemów opartych o Internet przed włamaniami i niepożądanym działaniem użytkowników
- Przenośność kodu systemów opartych o Internet, zmiany wersji platform
- Bazy danych i dostęp do nich w systemach opartych o Internet (stosowanie rozproszonych baz danych)
- Replikacja danych, replikacja obliczeń
- Serwery nazw, serwery informacyjne

Podstawy teoretyczne oraz problemy w programowaniu w Internecie

- Programowanie komponentowe, złożone platformy – J2EE (serwer aplikacji)
- Zasady projektowania i implementacji warstwy prezentacji
- Zasady projektowania i implementacji warstwy biznesowej
- Model MVC (Model-View-Controller)

Podstawy teoretyczne oraz problemy w programowaniu w Internecie

- Równoważenie obciążenia w systemach internetowych
- Wzorce projektowe – rozwiązywanie typowych problemów
- XML w systemach internetowych
- Łatwość rekonfiguracji gdy parametry w deskryptorach (np.. J2EE), kod może zostać użyty ponownie z innymi parametrami

Bezpieczeństwo komunikacji, HTTPS, uwierzytelnianie

- Konfiguracja HTTPS (przykład na bazie Tomcata, szyfrowanie symetryczne, asymetryczne, certyfikaty, klucze prywatne, publiczne)
- Możliwość wykorzystania HTTP-AUTH (w przyszłości na przykładzie aplikacji w PHP)
- Możliwość uwierzytelniania na poziomie aplikacji
- Na platformie J2EE, zarządzanie użytkownikami i uprawnieniami uproszczone (również możliwość kontroli na poziomie kodu w razie potrzeby)
- Uwierzytelnianie, delegowanie uprawnień

Obsługa różnych przeglądarek/standardów

- Należy obsługiwać przynajmniej najbardziej typowe przeglądarki internetowe, zaprezentować wyniki ankiet:
  - Aplikacja po stronie serwera może sprawdzić z jakiej przeglądarki korzysta klient
  - Odpowiedni komunikat gdy nie jest obsługiwana
  - Najbardziej optymalnie obsłużyć wszystkie, włącznie z lynx

Gruby a cienki klient – providerzy aplikacji

- Internet
- Intranet - funkcje
- Providerzy aplikacji
- Gruby/cienki klient

Zabezpieczanie warstwy prezentacji

- Sprawdzanie parametrów stron
- Sprawdzanie parametrów w kontekście sekwencji wywołań (bardzo wiele możliwości)
- Weryfikacja parametrów w kontekście wcześniej złożonego żądania
- Usuwanie niebezpiecznych znaków z pól tekstowych itp. – możliwość np. przedłużenia zapytania SQL lub polecenia w shellu etc.

Wieloprocesorowość, wiele procesów, wątków

- Wieloprocesorowość, klastry, węzły procesory wielordzeniowe
- Wieloprocesowość
- wielowątkowość

MySQL

Konfiguracja bazy danych, tworzenie bazy

Bazy danych

Konfiguracja

- Wygodna instalacja z pakietu w różnych dystrybucjach systemu Linux
- Należy zwrócić uwagę na domyślne ustawienia, w szczególności możliwość domyślnego zablokowania połączeń sieciowych (edytowane w pliku /etc/mysql/my.cnf)
- Nie konfigurować bazy z pominięciem uprawnień
- Możliwość skonfigurowania bezpiecznego połączenia z bazą danych
- Możliwość skonfigurowania replikacji bazy danych, z serwerem master oraz serwerami slave
  - Zrównoleglenie zapytań do bazy danych (niemodyfikujących np. SELECT)
- Szyfrowanie całej bazy danych (cloudscape)

- Bardziej zaawansowane rozwiązania ukrywające równoważenie obciążenia, niezawodność
- Replikacja – wydajność i niezawodność
- Algorytmy ROWA, quorum consensus

Konfiguracja serwera MySQL

- Instalacja i konfiguracja serwera MySQL
  - Instalacja mySQL
  - Uruchom serwer MySQL:
    cd /usr
    /usr/bin/safe_mysqld &
    [root@wolf usr]# Starting mysqld daemon with databases from /var/lib/mysql
  - Lub bez grant tables (jeśli istnieje problem konfiguracyjny, niezalecane ze względów bezpieczeństwa)
  - /usr/bin/safe_mysqld –skip-grant-tables &
    [root@wolf usr]# Starting mysqld daemon with databases from /var/lib/mysql

Konfiguracja serwera MySQL i tworzenie bazy danych

- Instalacja i konfiguracja serwera MySQL
  - Instalacja mySQL

  - Utwórz bazę danych:

```
su
mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.23.49

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
mysql> create database survey;
Query OK, 1 row affected (0.06 sec)
```

Konstrukcja relacyjnej bazy danych dla systemów internetowych

- Standardowa relacyjna baza danych, należy zwrócić uwagę na:

  - Liczbę zapytań kierowanych do bazy danych (minimalizowanie), możliwość pracy pod dużym obciążeniem, równoważenie zapytań

  - Zapewnienie spójności bazy danych (np. serwlet w obsłudze żądania jednego klienta kilka razy modyfikuje bazę danych, możliwość równoczesnej obsługi wielu klientów – możliwość niewłaściwego zarządzania danymi),

  - Transakcje (koncepcja, poziomy izolacji)

  - Sposób zapisu danych w bazie danych, możliwe wykorzystanie formatu XML jako formatu pośredniego, przydatnego w prezentacji danych i konwersji do różnych formatów (jeśli serwis tego potrzebuje)

Konfiguracja serwera MySQL

Instalacja i konfiguracja serwera MySQL

- Ustalenie praw dla użytkownika:

```
mysql>
mysql> grant all on survey.* to pczarnul@localhost identified by
"*********";
Query OK, 0 rows affected (0.00 sec)
mysql>
```

- Utwórz tabele: utwórz skrypt createdatabasescript , który tworzy tabele:

```
create database survey;
use survey;
create table entries (
entry_id char(10) not null,
author_id char(10) not null,
text varchar (200),
unique entry_id (entry_id)
);
```

Tworzenie bazy danych w MySQL

- Instalacja i konfiguracja serwera MySQL
  - Utwórz tabele: utwórz skrypt createdatabasescript , który tworzy tabele:

```
create table authors (
author_id char(10) not null,
firstname varchar (200),
lastname varchar (200),
email varchar(100),
unique author_id (author_id)
);
```

  - Uruchom skrypt createdatabasescript jako użytkownik:

```
mysql -p
<enter password>
source createdatabasescript;
```

  - Zwróć uwagę na nazwę komputera:
    grant all on survey.* to pczarnul@localhost identified by "qwerty";
    Powinno zadziałać!!!

Wykorzystanie protokołu HTTP

- Zalety:
  - Komunikacja typu request-response
  - Serwlet obsługuje metody doGet, doPost
    - Możliwość odczytu parametrów, generowania dynamicznej treści w odpowiedzi
    - Obiekt sesji
    - Cookies
    - Możliwość instalowania filtrów
  - Sposób przekazywania parametrów w metodzie GET, POST (zabezpieczanie serwletów, serwisu przed podmianą parametrów, przykład w praktyce – wcześniejsze żądanie klienta zostaje zapisane do bazy, później weryfikacja – błąd gdy ponownie dane nieprawdziwe)
  - Ukrywanie parametrów, URL stron

Wykorzystanie protokołu HTTP

- Zalety:
  - Nagłówki, ciało metody
  - Parametry, które serwlet może poznać – identyfikacja komputera użytkownika itd.
  - Jak właściwie skonstruować licznik stron?

Tomcat (Serwlety + JSP): Instalacja

- Tomcat – contener dla:
  - Serwletów
  - Stron JSP

1. Instalacja serwera Tomcat w systemie Linux
   - rozpakuj
     ```
     cd /usr/local
     tar xvzf jakarta-tomcat-4.1.12.tar.gz
     ```
   - wyedytuj /etc/profile
     ```
     JAVA_HOME=/usr/local/j2sdk1.4.0
     export JAVA_HOME

     CATALINA_HOME=/usr/local/jakarta-tomcat-4.1.12
     export CATALINA_HOME
     ```
2. Uruchom serwer Tomcat
   ```
   %CATALINA_HOME%\bin\startup          (Windows)
   $CATALINA_HOME/bin/startup.sh        (Unix)
   ```

Tomcat (Serwlety + JSP): uruchomienie/zatrzymanie serwera

- Uruchom serwer Tomcat

```
[root@wolf usr]# $CATALINA_HOME/bin/startup.sh
Using CATALINA_BASE:   /usr/local/jakarta-tomcat-4.1.12
Using CATALINA_HOME:   /usr/local/jakarta-tomcat-4.1.12
Using CATALINA_TMPDIR: /usr/local/jakarta-tomcat-4.1.12/temp
Using JAVA_HOME:       /usr/local/j2sdk1.4.0
[root@wolf usr]#
```

- Zatrzymaj serwer Tomcat

```
%CATALINA_HOME%\bin\shutdown          (Windows)
$CATALINA_HOME/bin/shutdown.sh        (Unix)

[root@wolf usr]# $CATALINA_HOME/bin/shutdown.sh
Using CATALINA_BASE:   /usr/local/jakarta-tomcat-4.1.12
Using CATALINA_HOME:   /usr/local/jakarta-tomcat-4.1.12
Using CATALINA_TMPDIR: /usr/local/jakarta-tomcat-4.1.12/temp
Using JAVA_HOME:       /usr/local/j2sdk1.4.0
```

Tomcat (Serwlety + JSP): Dostęp

1. Sprawdć dostęp serwera poprzez wywołanie:
   http://localhost
2. Gdzie umieścić pliki .class w drzewie katalogów serwera Tomcat?

   skopiuj X.class file do /usr/local/jakarta-tomcat-4.1.12/webapps/examples/WEB-INF/classes

   i uzuskaj dostęp poprzez wywołanie
   http://localhost:8080/examples/servlet/X

Projekt „Książka gości” oparty o serwlety

- Cele projektu
  - Dodaj komentarze i informacje o autorach:
    - Kometarz (tekst), imię, nazwisko oraz email autora
  - Przeglądaj wcześniej dodane komentarze
  - Szukaj czy dany autor już istnieje (identyfikowany przez imię i nazwisko) i nie dodawaj duplikatu informacji o autorze, który już istnieje!!!
- Kod związany z dostępem do bazy danych
  - Otwórz bazę danych
  - Zapisz dane do tabeli
  - Znajdź unikalne id w kolumnie tabeli (może być wykonane przez bazę danych)
  - Pobierz dane z tabeli
- Logika biznesowa
  - Wstaw komentarz
  - Pokaż komentarze

1. Cienki klient – przeglądarka internetowa (dostęp do serwletów)

Kod dostępu do bazy danych

Otwórz bazę (połączenie)

Kod dostępu do bazy danych

```
import java.sql.*;
import java.util.*;
public class Database {

public static Connection OpenDatabase(String sDatabaseUrl,String
sLogin,String sPassword) {

Connection cConnection;
String sSQL;
Statement stmtStatement;

try {
    Class.forName("org.gjt.mm.mysql.Driver");
} catch(java.lang.ClassNotFoundException e) {
    System.err.print("ClassNotFoundException: ");
    System.err.println(e.getMessage());
}
```

Kod dostępu do bazy danych

```
  try {
  cConnection =
DriverManager.getConnection(sDatabaseUrl);//,sLogin,sPassword);
  } catch(SQLException ex) {
  System.err.println("SQLException: " + ex.getMessage());
  cConnection=null;
  }
  return cConnection;
}

public static void CloseDatabase(Connection cConnection) {
  try {
      cConnection.close();
   } catch(SQLException ex) {
      System.err.println("SQLException: " + ex.getMessage());
  }
}
```

Kod dostępu do bazy danych

Zapisz dane do tabeli

Kod dostępu do bazy danych

```
 public static void InsertIntoTable(Connection cConnection,String
sTableName,String[] sFieldNames,String[] sValues) {

// inserts values corresponding to given fields to the given table
StringBuffer sbSQL;
Statement stmtStatement;
        int i;

try {
        stmtStatement = cConnection.createStatement();

sbSQL=new StringBuffer().append("insert into
").append(sTableName).append(" (");

        for(i=0;i<sFieldNames.length;i++) {
        sbSQL.append(sFieldNames[i]);

        if (i<(sFieldNames.length-1)) sbSQL.append(",");
        else sbSQL.append(") values (");
}
```

Kod dostępu do bazy danych

```
    for(i=0;i<sValues.length;i++) {
       sbSQL.append("'").append(sValues[i]).append("'");

       if (i<(sFieldNames.length-1)) sbSQL.append(",");
       else sbSQL.append(")");
    }

    stmtStatement.executeUpdate(sbSQL.toString());
    stmtStatement.close();

} catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }

    }
```

Kod dostępu do bazy danych

Znajdź unikalne id w tabeli
(można wykorzystać opcję auto_increment przy tworzeniu tabeli)

Kod dostępu do bazy danych

```
 public static String GetUniqueId(Connection cConnection,String
sTableName,String sColumnName) {

// returns a new unique id in the given table for the given column
// it is computed as the smallest new value in the column

ResultSet rsResult;
StringBuffer sbSQL;
Statement stmtStatement;
String sId;
int i;

// select the maximum value (if there is at least one entry) and increase it by
one;
// otherwise return 1000

try {
      stmtStatement = cConnection.createStatement();

sbSQL=new StringBuffer().append("select
").append(sColumnName).append(" from ").append(sTableName);
```

Kod dostępu do bazy danych

```
 rsResult=stmtStatement.executeQuery(sbSQL.toString());

 // iterate through the ids
       int nMaxId=0,nTempId;

 while (rsResult.next()) {
if (nMaxId<(nTempId=rsResult.getInt(sColumnName)))
nMaxId=nTempId;
  }
if (nMaxId==0) nMaxId=1000;          // start from 1000 -- the lower values are
reserved to enter new values by the author of this program

        sId=new String(Integer.toString(1+nMaxId));
  stmtStatement.close();
   } catch(SQLException ex) {
  System.err.println("SQLException: " + ex.getMessage());
        sId=null;
   }
   return sId;
```

Kod dostępu do bazy danych

Pobierz dane z tabeli

Kod dostępu do bazy danych

```
// this class represents basic data as returned from a database - a set of columns

class BasicDatabaseData {

    public String[] sColumnValues; // values of the returned columns the programmer is interested in (e.g. an id, a description field, another id, etc.)

    BasicDatabaseData(int nColumnCount) { // initialize with the given number of columns
sColumnValues=new String[nColumnCount];
    }
};
```

Kod dostępu do bazy danych

```
  public static BasicDatabaseData[] GetData(Connection cConnection,String[]
sTableNames,String[] sColumns,String sWhereStatement) {

// this method returns columns from the given table

ResultSet rsResult;
StringBuffer sbSQL;
      Statement stmtStatement;
int i;
BasicDatabaseData[] bddColumnsValues=null; // the return value
Vector vColumnsValues=new Vector();

if (sColumns.length<1) return null; // no output when no input

      try {
   stmtStatement = cConnection.createStatement();

   // create a statement
   sbSQL=new StringBuffer().append("select ");
```

Kod dostępu do bazy danych

```
for(i=0;i<sColumns.length;i++) {
sbSQL.append(sColumns[i]);
if (i<(sColumns.length-1)) sbSQL.append(",");
}
sbSQL.append(" from ");
for(i=0;i<sTableNames.length;i++) {
sbSQL.append(sTableNames[i]);
if (i<(sTableNames.length-1)) sbSQL.append(",");
}

sbSQL.append(" ").append(sWhereStatement);
rsResult=stmtStatement.executeQuery(sbSQL.toString());

// iterate through the data
while (rsResult.next()) {
BasicDatabaseData bddTemp=new BasicDatabaseData(sColumns.length);
```

Kod dostępu do bazy danych

```
        for(i=0;i<sColumns.length;i++)
        bddTemp.sColumnValues[i]=new
              String(rsResult.getString(sColumns[i]));

vColumnsValues.add(bddTemp);
}
if (vColumnsValues.size()>0) { // if there are returned entries
bddColumnsValues=new
        BasicDatabaseData[vColumnsValues.size()];
for(i=0;i<bddColumnsValues.length;i++)

bddColumnsValues[i]=(BasicDatabaseData)vColumnsValues.get(i);        // we
only copy references here, no need for memory allocation
}
stmtStatement.close();
   } catch(SQLException ex) {
System.err.println("SQLException: " + ex.getMessage());
   }
   return bddColumnsValues;
```

Kod serwletu

Zapisz komentarz

Kod serwletu – wstaw komentarz

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.sql.*;

public final class InsertComment extends HttpServlet {
    Connection cConnection;     // the connection to the database

    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException, ServletException {
doGet(request,response);
    }
```

Kod serwletu – wstaw komentarz

```
public void doGet(HttpServletRequest request, HttpServletResponse res)
    throws IOException, ServletException
    {
String sFirstName=null;
String sLastName=null;
String sEmail=null;
String sText=null;

        Enumeration e = request.getParameterNames();
        PrintWriter out = res.getWriter ();

out.println("<html>");
out.println("<head>");
out.println("<title>Sample Application Servlet Page</title>");
out.println("</head>");
out.println("<body bgcolor=white>");
```

Kod serwletu – wstaw komentarz

```
    while (e.hasMoreElements()) {
        String name = (String)e.nextElement();
        String value = request.getParameter(name);
        out.println(name + " = " + value);

if (name.equals("firstname")) {
   sFirstName=value;
}
if (name.equals("lastname")) {
   sLastName=value;
}
if (name.equals("email")) {
   sEmail=value;
}
if (name.equals("text")) {
   sText=value;
}
}
```

Kod serwletu – wstaw komentarz

```
if ((sFirstName!=null) && (sLastName!=null) && (sEmail!=null) &&
(sText!=null)) {
    // insert into the database
    if
(null!=(cConnection=Database.OpenDatabase("jdbc:mysql://localhost/"+"surv
ey"+"?user="+"pczarnul"+"&password="+"qwerty","pczarnul@localhost","qwe
rty"))) {

        // check if the author is in the database
        String[] sTableNames={"authors"};
        String[] sColumns={"author_id","firstname","lastname"};
        String sWhereStatement="where firstname="+sFirstName+" and
lastname="+sLastName;

        BasicDatabaseData[] bddDatabaseData=Database.GetData
(cConnection,sTableNames,sColumns,sWhereStatement);
```

Kod serwletu – wstaw komentarz

```
        String sAuthorId;

        if (bddDatabaseData!=null) { // if at least one entry
            // get the author_id from the first entry
            sAuthorId=bddDatabaseData[0].sColumnValues[0]; //
        } else {
            sAuthorId=new String(Database.GetUniqueId
(cConnection,"authors","author_id"));

            String[] sAuthorsFieldNames={"author_id","firstname",
"lastname","email"};
            String[] sAuthorsValues=new String[4];
            sAuthorsValues[0]=sAuthorId;
            sAuthorsValues[1]=sFirstName;
            sAuthorsValues[2]=sLastName;
            sAuthorsValues[3]=sEmail;

Database.InsertIntoTable(cConnection,"authors",sAuthorsFieldNames,sAuth
orsValues);
```

Kod serwletu – wstaw komentarz

```
        }

        // insert into entries
        String[] sEntriesFieldNames={"entry_id","author_id","text"};
        String[] sEntriesValues=new String[3];
        sEntriesValues[0]=newString(Database.GetUniqueId(cConnection,
"entries","entry_id"));
        sEntriesValues[1]=sAuthorId;
        sEntriesValues[2]=sText;

Database.InsertIntoTable(cConnection,"entries",sEntriesFieldNames,sEntries
Values);
    } else  out.println("Cannot connect to
jdbc:mysql://localhost/"+"survey"+"?user="+
"pczarnul"+"&password="+"qwerty");

}
```

✂ Podstawy teoretyczne oraz problemy w programowaniu w Internecie

- Konfiguracja HTTPS (przykład na bazie Tomcata, szyfrowanie symetryczne, asymetryczne, certyfikaty, klucze prywatne, publiczne)
- Możliwość wykorzystania HTTP-AUTH (w przyszłości na przykładzie aplikacji w PHP)
- Możliwość uwierzytelniania na poziomie aplikacji
- Na platformie J2EE, zarządzanie użytkownikami i uprawnieniami uproszczone (również możliwość kontroli na poziomie kodu w razie potrzeby)

Wydajność komunikacji, założenia co do wydajności, zawodności sieci

- Podstawy teoretyczne oraz problemy w programowaniu w Internecie

  - Komunikacja – parametry:

    - Przepustowość (sprawdzenie w praktyce)
    - tzw. Startup-time
    - Niezawodność sieci
    - komunikacja strumieniowa, pakietowa etc.

Obsługa różnych przeglądarek internetowych

- Podstawy teoretyczne oraz problemy w programowaniu w Internecie

  - Należy obsługiwać przynajmniej najbardziej typowe przeglądarki internetowe, zaprezentować wyniki ankiet:

    - Aplikacja po stronie serwera może sprawdzić z jakiej przeglądarki korzysta klient

    - Odpowiedni komunikat gdy nie jest obsługiwana

    - Najbardziej optymalnie obsłużyć wszystkie, włącznie z lynx

Złożoność platformy a wydajność i łatwość programowania

- Podstawy teoretyczne oraz problemy w programowaniu w Internecie

  - PHP, serwlety, JSP jako całościowe technologie – łatwość programowania prostych serwisów, trudność w zarządzaniu dużym kodem
    - Uwierzytelnianie, transakcje etc. często na barkach aplikacji

  - Java EE
    - Programowanie komponentowe

Zabezpieczanie warstwy prezentacji

- Podstawy teoretyczne oraz problemy w programowaniu w Internecie

  - Sprawdzanie parametrów stron
  - Sprawdzanie parametrów w kontekście sekwencji wywołań (bardzo wiele możliwości)
  - Weryfikacja parametrów w kontekście wcześniej złożonego żądania
  - Usuwanie niebezpiecznych znaków z pól tekstowych itp. – możliwość np. przedłużenia zapytania SQL lub polecenia w shellu etc.