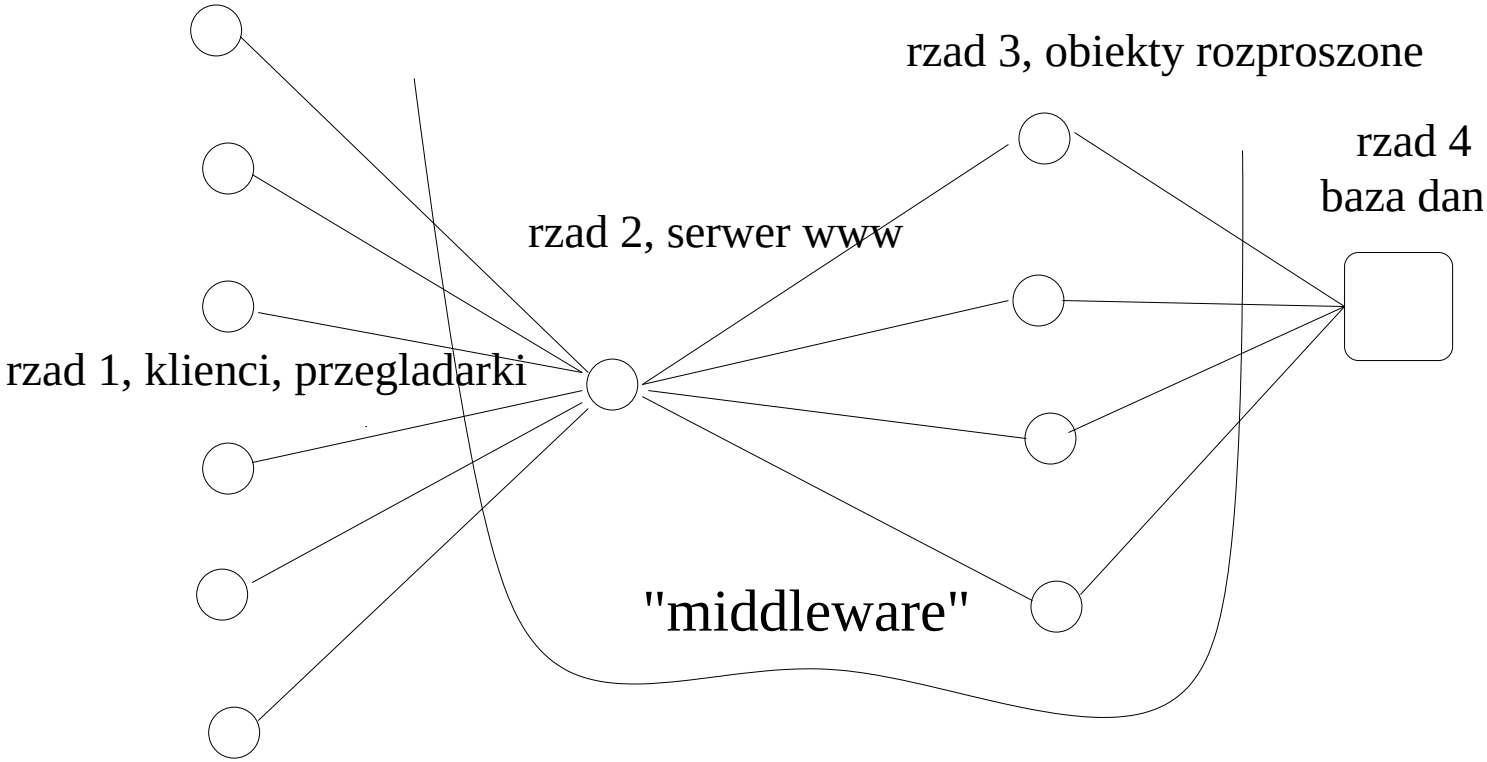
SSI = Sieciowe Systemy Informacyjne

Wykład nr 5

Obiekty rozproszone, WebService
(rząd 3 w arch 4 rzędowej)



- CORBA - obiekty rozproszone, wywoływanie zdalnych metod, prot binarny
- WebService/SOAP - wywoływanie zdalnej procedury lub coś więcej, prot tekstowe używające XML
- WebService/REST - czyste HTTP, dosłowne rozumienie metod http, używa się XML/JSON

CORBA

- na początku jest plik IDL (ang. Interface Definition Lang) ...

```
module HelloApp
{  interface Hello // interfejs ob. CORBY (zbiór metod)
    {  string sayHello();
        long podajLiczbe();
        double sumaLiczb(in double a, in double b);
        typedef long tab[10]; // definicja typu "tab"
        void wektorRazyDwa(in tab tab1, out tab tab2);
    };
};
```

- plik IDL tłumaczy się specjalnym programem na:
  pieniek (po stronie klienta, odwołania do ob. CORBY jak do zwykłego obiektu)
  szkielet (po stronie serwera, umożliwia podanie implementacji ob. CORBY)
- brokery CORBY (j. Java "Java IDL", j. Tcl "combat", j. C++ "mico"),
  referencja do ob. CORBY, string IOR,
  standardowe serwisy CORBY (np. NameService, PropertyService, EventService, TransactionService)
- po stronie serwera jest POA, pozwala na dość złożone rzeczy, np. ???
- zalety CORBY: binarny prot IIOP, niezależność od języka prog, liczne standardowe serwisy, skalowalność (zdolność obsługi dużej liczby klientów/obiektów)
- patrz: http://mhanckow.students.wmi.amu.edu.pl/corba/pzr420_cw_c.htm

WebService/SOAP

- komunikaty SOAP (ang. Simple Object Access Prot)
  xml-owe komunikaty typu żądanie/odpowiedź,
  parametry i wynik zakodowane w xml-u ... (pokazać!)
- plik WSDL (ang. WebService Description Lang)
  opisują operacje WS, też XML, automatycznie generowane na podstawie opisu interfejsu WS w danym języku programowania, np. gSOAP używa plików nagłówkowych .h
- toolkity WS: j. Java "Axis", j. C/C++ "gSOAP", j. Tcl "tclws"
- pieniek klienta tworzony automatycznie na podstawie pliku WSDL

```
# przykład w j. Tcl, toolkit "tclws"
set w1 [::WS::Client::GetAndParseWsdl \
    http://www.webservicex.net/globalweather.asmx?WSDL]
dict get $w1 name
  #% GlobalWeather
::WS::Client::CreateStubs GlobalWeather
GlobalWeather::GetWeather "Poznan" "Poland"
  # + powinno pokazać pogodę na lotnisku w Poznaniu...
```

- styl WS: **rpc**/encoded i **document**/literal
- patrz: http://mhanckow.students.wmi.amu.edu.pl/corba/pzr420_cw_d.htm
- *pytanie*: dlaczego WebService ma w nazwie słowo "Web" ?

WebService/REST

- REST (ang. Representational State Transfer ???)
  to jest najprostsza technologia!!!
  zdalną procedurę uruchamia się wysyłając żądanie http,
  nazwa procedury jest w url-u (lub tam gdzie parametry procedury),
  parametry procedury są umieszczane w danych POST lub w url-u,
  wynik przychodzi w odpowiedzi http,
  parametry i wynik procedury są zakodowane przy pomocy formatów XML lub JSON

- większość j. programowania wspiera prot http od strony klienta
  (np. pakiet http j. Tcl, pakiet java.net w j. Java, pokaz przykład wykl05/http01.tcl)

- jak zaimplementować serwer WS/REST ?
  potrzebujemy serwera www, w którym skrypty mogą generować
  nie tylko odpwiedzi typu "text/html", ale takze "text/xml"
  cenne także wsparcie dla formatów XML i/lub JSON ...

- portale "Facebook", "Google Docs" używają REST API

- OpenStreetMap + Mapnik (http://tile.openstreetmap.org, silnik renderujący mapy)
  także można traktować jako WS/REST ... (pokazać przykład ...)