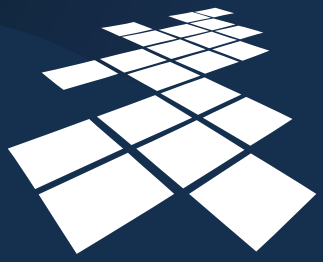
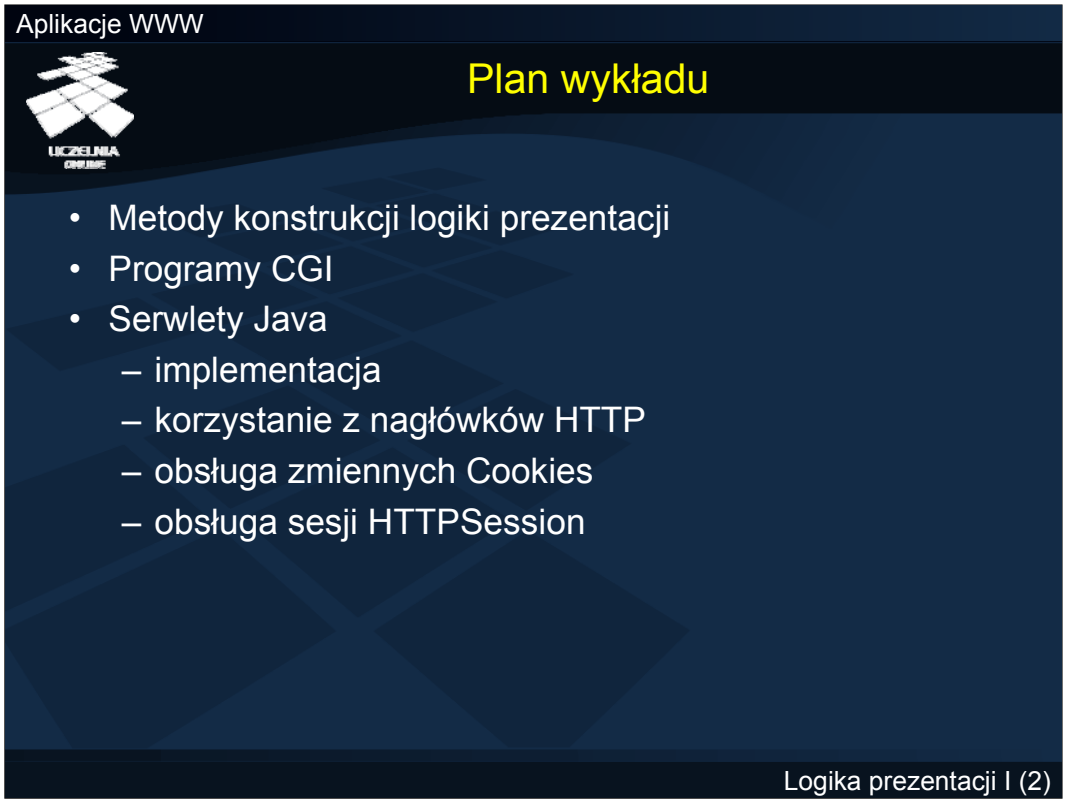
Wykład 5

Logika prezentacji - część I

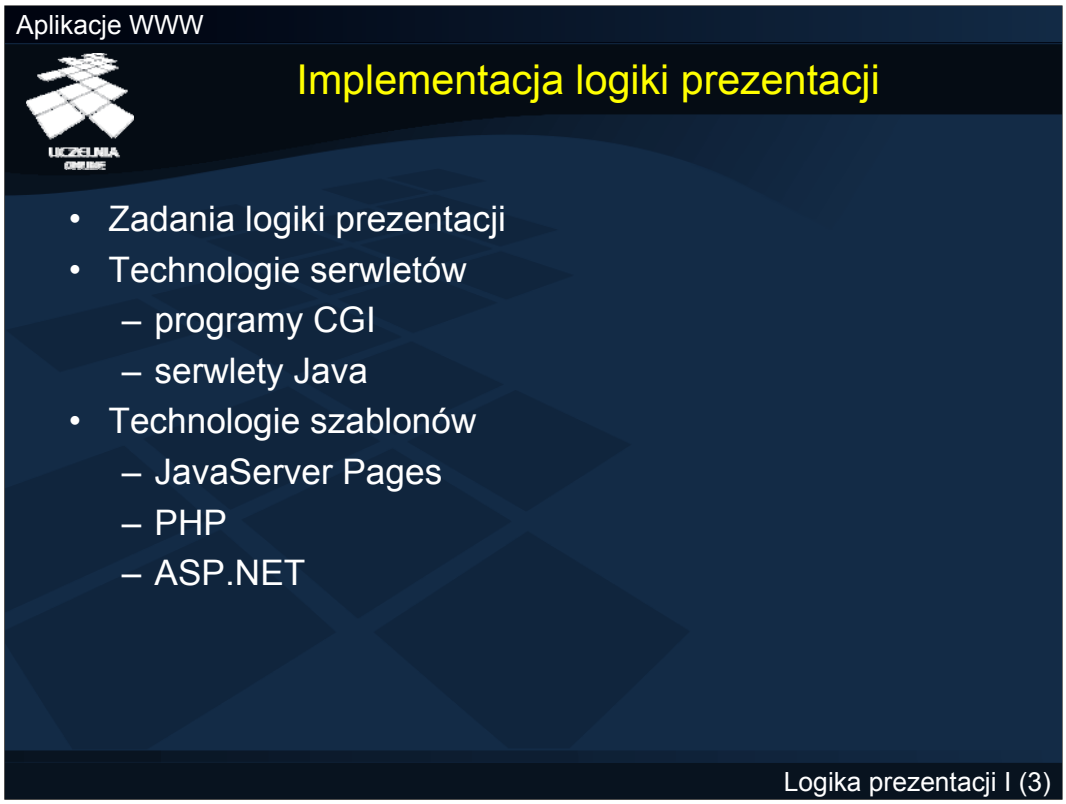
wykład prowadzi: Maciej Zakrzewicz



UCZELNIA
ONLINE



Celem wykładu jest przedstawienie klasyfikacji metod konstrukcji logiki prezentacji oraz omówienie dwóch przykładowych technologii implementacji serwletów: technologii CGI i technologii serwletów Java. Wykład obejmie architekturę i sposoby implementacji programów CGI, architekturę serwletów Java, sposoby implementacji serwletów Java, obsługę nagłówków HTTP, zmiennych Cookies i sesji HTTPSession.



Logika prezentacji stanowi część aplikacji WWW znajdującą się po stronie serwera HTTP odpowiedzialną za generowanie graficznego interfejsu użytkownika, np. w formie dokumentów HTML. Wykonanie kodu logiki prezentacji odbywa się zawsze wyłącznie na żądanie użytkownika końcowego. Z kolei każde żądanie użytkownika końcowego powoduje ponowne wykonanie kodu logiki prezentacji.

W wyniku ewolucji technologii WWW wyodrębnione zostały dwa zasadnicze podejścia do konstrukcji modułów logiki prezentacji:

1. Technologie serwletów - logika prezentacji ma postać aplikacji wykonywalnej, która odpowiada za wygenerowanie kompletnego dokumentu dla użytkownika końcowego. Tego typu aplikacje są zwykle nazywane serwletami (servlets), a najważniejszymi technologiami ich implementacji są: CGI i serwlety Java.

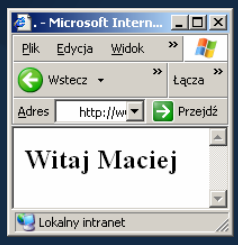
2. Technologie szablonów - logika prezentacji ma postać szablonu dokumentu, w który wplecione są fragmenty kodu wykonywalnego. Obsługa żądania użytkownika końcowego polega na wykonaniu fragmentów kodu, a następnie osadzeniu ich wyników w szablonie. Najważniejszymi technologiami implementacji szablonów są: JavaServer Pages, PHP, ASP.NET.

W dalszej części tego wykładu omówimy wybrane technologie serwletów, wykorzystywane do implementacji logiki prezentacji.

Przykład: serwlety vs. szablony

Serwlet

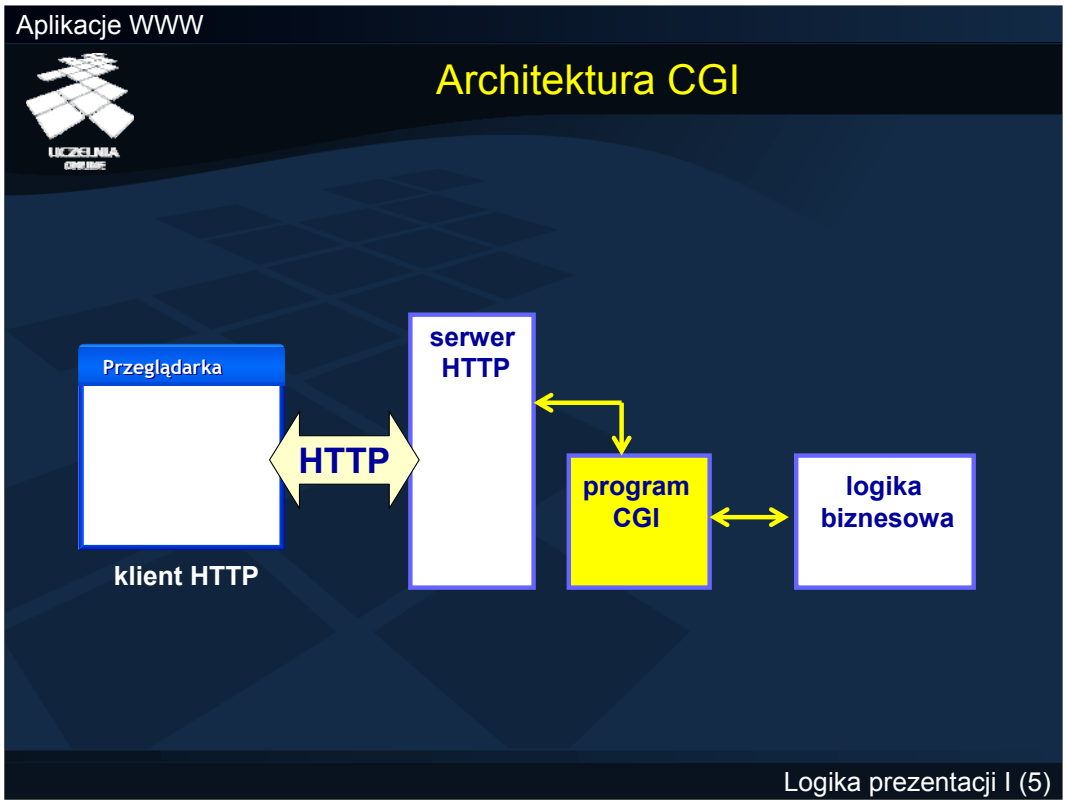
```
import javax.servlet.http.*;
import java.io.*;

public class MyServlet1 extends HttpServlet {
  public void doGet(HttpServletRequet request,
                    HttpServletResponse response)
                    throws IOException {
    String imie = request.getParameter("imie");
    response.setContentType("text/h
    PrintWriter out = response.getWr
    out.println("<HTML><BODY>");
    out.println("<H2>Witaj "+imie+"<
    out.println("</BODY></HTML>");
}}
```



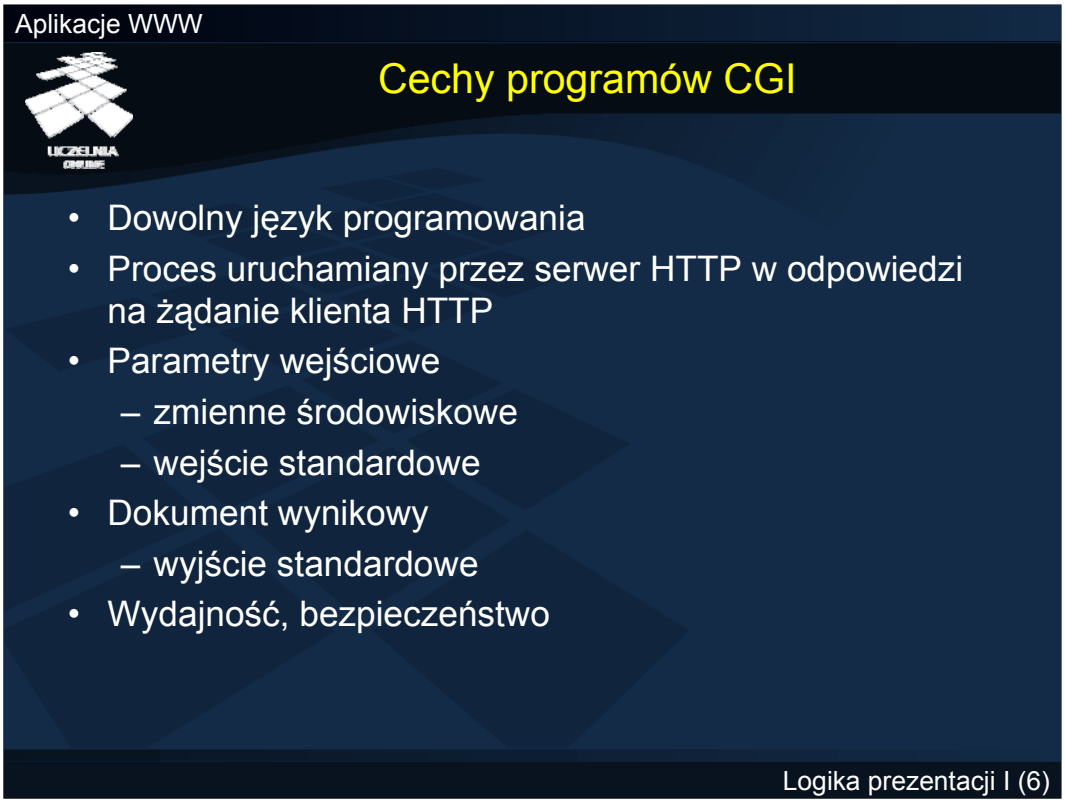
Szablon

```
<% String imie =
     request.getParameter("imie"); %>
<HTML>
  <BODY>
    <H2>Witaj <%=imie%></H2>
  </BODY>
</HML>
```


Powyższy slajd ilustruje oba wymienione podejścia do konstrukcji logiki prezentacji na przykładzie identycznych funkcjonalnie implementacji. Serwlet został zaimplementowany jako tzw. serwlet Java, a szablon jako tzw. JavaServer Page. W większości przypadków kod źródłowy serwletu będzie obszerniejszy od kodu źródłowego alternatywnego szablonu, w związku z czym technologie szablonów cieszą się większą popularnością wśród programistów.



Common Gateway Interface (CGI) był pierwszą specyfikacją opracowaną na potrzeby aplikacji WWW uruchamianych po stronie serwera HTTP. Zgodnie z założeniami CGI, serwer HTTP może na żądanie klienta HTTP uruchomić lokalną aplikację, nazywaną programem CGI, która odpowiada za wygenerowanie treści dokumentu, np. w formacie HTML, przekazywanego w odpowiedzi na żądanie klienta. Program CGI może komunikować się z zewnętrznym kodem logiki biznesowej, jednak nie mieści się to w zakresie omawianej specyfikacji.



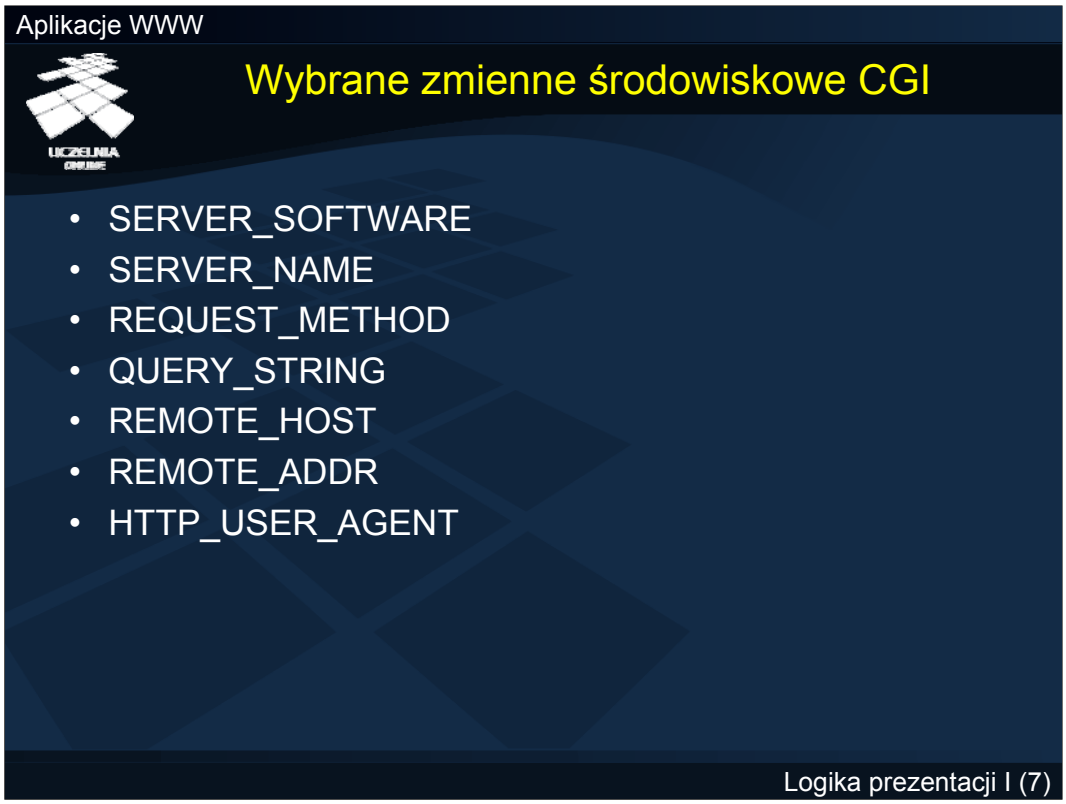
Program CGI może być zaimplementowany w dowolnym języku programowania, który jest wykonywalny w środowisku systemu operacyjnego serwera HTTP (np. C, Pascal, Perl, skrypt UNIX, skrypt BAT).

Na potrzeby obsługi każdego żądania HTTP, serwer HTTP tworzy nowy proces systemu operacyjnego i w jego ramach wykonuje program CGI. Po zakończeniu pracy programu CGI proces ten jest usuwany.

Komunikacja pomiędzy serwerem HTTP a programem CGI ma bardzo uproszczoną formę. Serwer HTTP przekazuje programowi CGI parametry wejściowe za pomocą zestawu predefiniowanych zmiennych środowiskowych, za pomocą mechanizmu systemowego wejścia standardowego (np. stdin), a bardzo rzadko za pomocą parametrów wiersza poleceń. Komunikacja w przeciwnym kierunku odbywa się wyłącznie za pośrednictwem mechanizmu systemowego wyjścia standardowego (np. stdout).

Poważnym problemem programów CGI jest ich niska wydajność spowodowana koniecznością uruchamiania nowego procesu na potrzeby obsługi każdego żądania HTTP. Przykładowo, obsługa 100 równoczesnych żądań HTTP wymaga uruchomienia 100 procesów! Aby rozwiązać ten problem, zaproponowano rozwinięcie technologii CGI o nazwie FastCGI - programy FastCGI mogą być buforowane w pamięci operacyjnej i ponownie wykorzystywane do obsługi kolejnych żądań HTTP.

Technologia CGI nie cieszy się też dobrą opinią w zakresie bezpieczeństwa, ponieważ stwarza możliwości uruchamiania dowolnych programów wykonywalnych w systemie operacyjnym serwera HTTP. Z tego powodu wielu administratorów serwerów HTTP wyłącza obsługę CGI.



Dla potrzeb komunikacji serwera HTTP programem CGI zdefiniowano zbiór kilkunastu zmiennych środowiskowych, w których serwer HTTP zapisuje parametry wywołania programu CGI. Odczyt tych parametrów jest zadaniem programisty. Warto nadmienić, że dla potrzeb komunikacji wybrano mechanizm zmiennych środowiskowych, ponieważ jest on obsługiwany przez wszelkie języki programowania.

Najpopularniejsze zmienne środowiskowe predefiniowane na potrzeby programów CGI to:

• SERWER_SOFTWARE - zawiera nazwę i numer wersji oprogramowania serwera HTTP, który przyjął żądanie HTTP,

• SERVER_NAME - zawiera nazwę sieciową, adres IP lub alias DNS komputera, do którego skierowane było żądanie HTTP,

• REQUEST_METHOD - zawiera otrzymaną komendę HTTP, np. GET, POST,

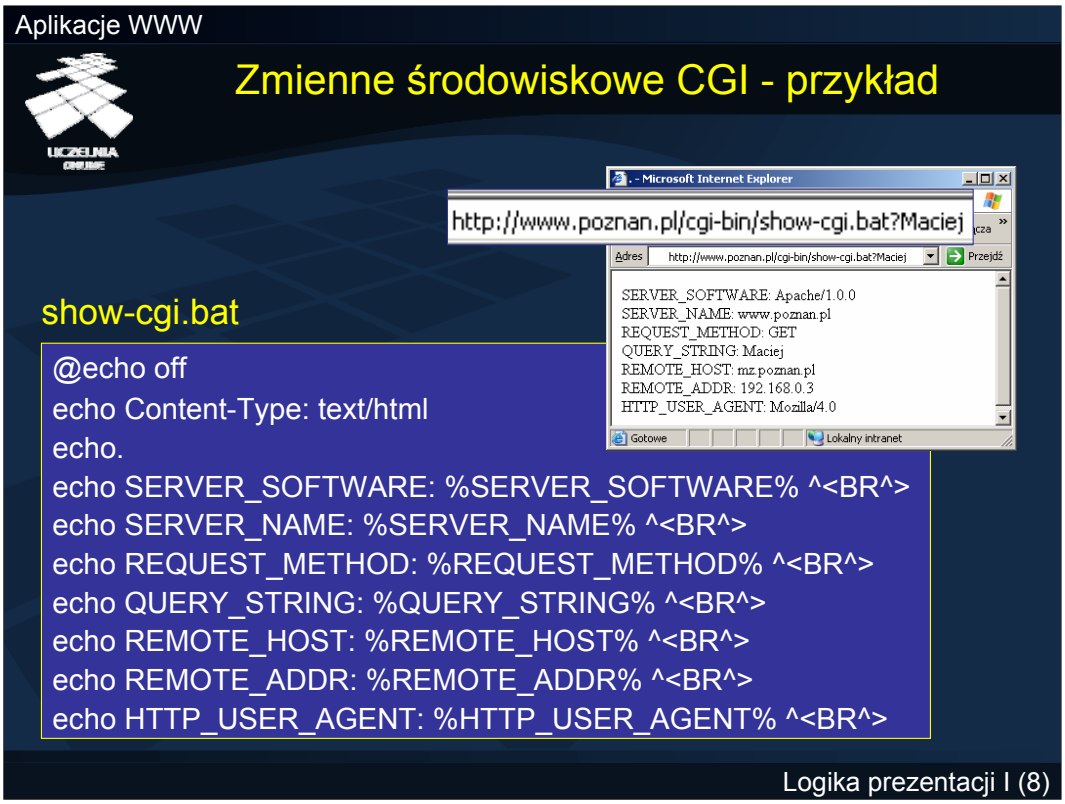
• QUERY_STRING - łańcuch znakowy rozpoczynający się od znaku "?" w adresie URL żądania HTTP; służy do przekazywania parametrów żądania HTTP typu GET

• REMOTE_HOST - zawiera nazwę sieciową komputera, z którego pochodzi żądanie HTTP,

• REMOTE_ADDR - zawiera adres IP komputera, z którego pochodzi żądanie HTTP,

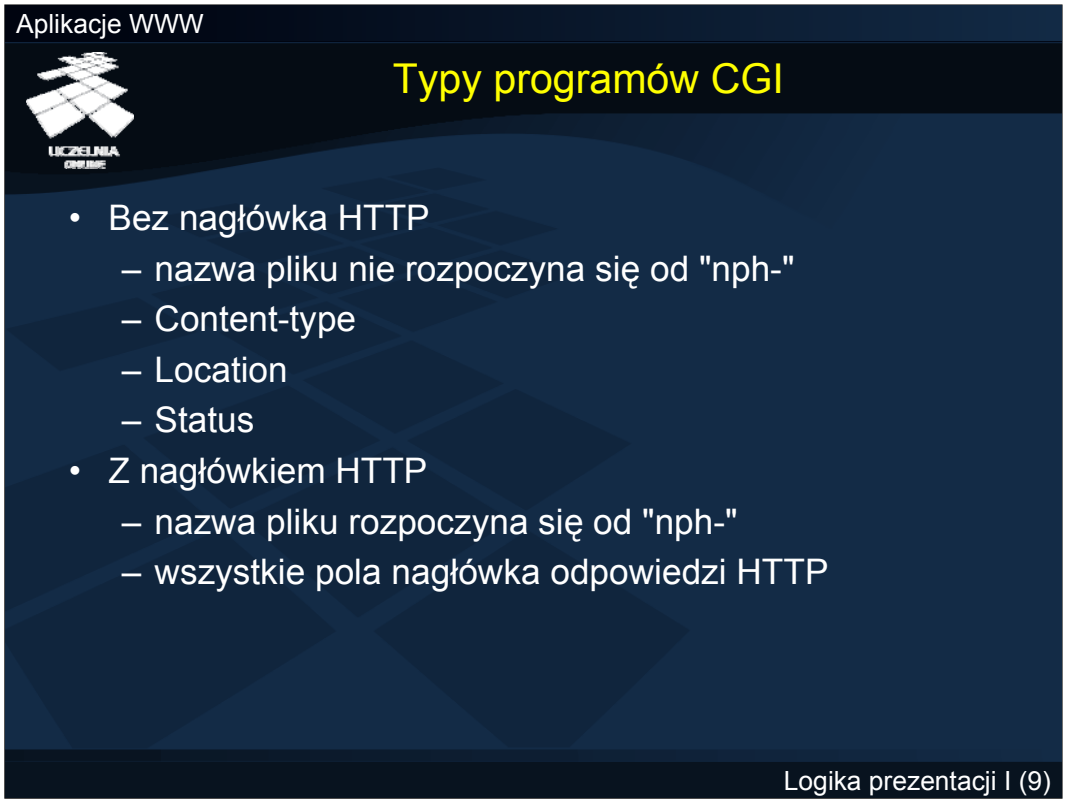
• HTTP_USER_AGENT - zawiera nazwę i numer wersji oprogramowania klienta HTTP, który wysłał żądanie HTTP.

Pozostałe zmienne środowiskowe CGI są opisane w specyfikacji CGI.



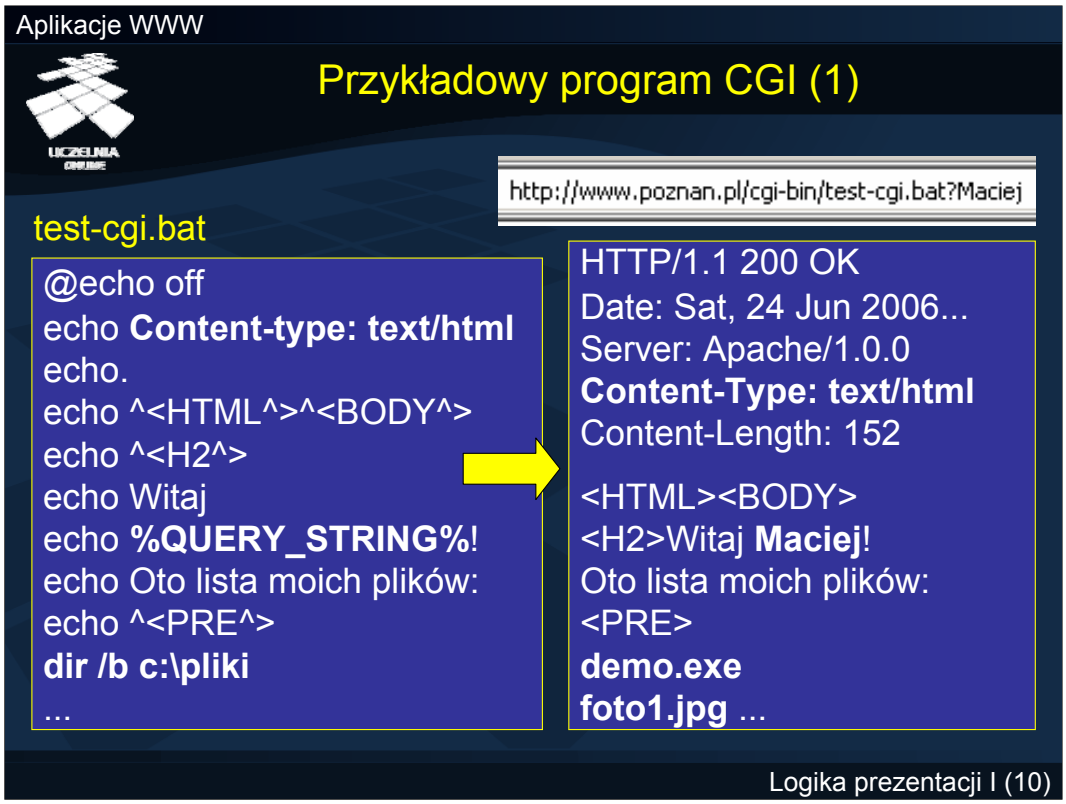
Na slajdzie przedstawiono przykład prostego programu CGI, który w odpowiedzi na żądanie HTTP generuje dokument HTML przedstawiający wartości wybranych zmiennych środowiskowych CGI. Szczegóły implementacji programów CGI zostaną omówione w dalszej części wykładu.

W celu uruchomienia programu CGI użytkownik końcowy formułuje żądanie zawierające adres URL pliku programu CGI. Aby takie wywołania były pomyślnie obsługiwane, serwer HTTP musi potrafić rozróżniać żądania pobrania dokumentu od żądań uruchomienia programu. W tym celu administrator serwera HTTP definiuje, które katalogi dyskowe zawierają programy wykonywalne, a które zawierają dokumenty statyczne.



Istnieją dwa sposoby implementacji programów CGI, różniące się sposobem tworzenia nagłówka odpowiedzi HTTP:

1. Programy CGI nie generujące nagłówka odpowiedzi HTTP - nagłówek odpowiedzi HTTP jest generowany automatycznie przez serwer HTTP. Program CGI może przekazać serwerowi HTTP wybrane informacje do umieszczenia w nagłówku HTTP. Służą do tego celu trzy dyrektywy: "Content-type", która wskazuje format dokumentu wynikowego, "Location", która wskazuje adres URL dla przekierowania żądania, "Status", która zawiera kod zwrotny odpowiedzi HTTP. Programy tego typu muszą być zapisane w plikach, których nazwa nie rozpoczyna się od słowa "nph-".
2. Programy CGI generujące nagłówek odpowiedzi HTTP - kompletny nagłówek odpowiedzi HTTP musi zostać wygenerowany przez program CGI. Programy tego typu muszą być zapisane w plikach, których nazwa rozpoczyna się od słowa "nph-" (Non-Parsed Headers).



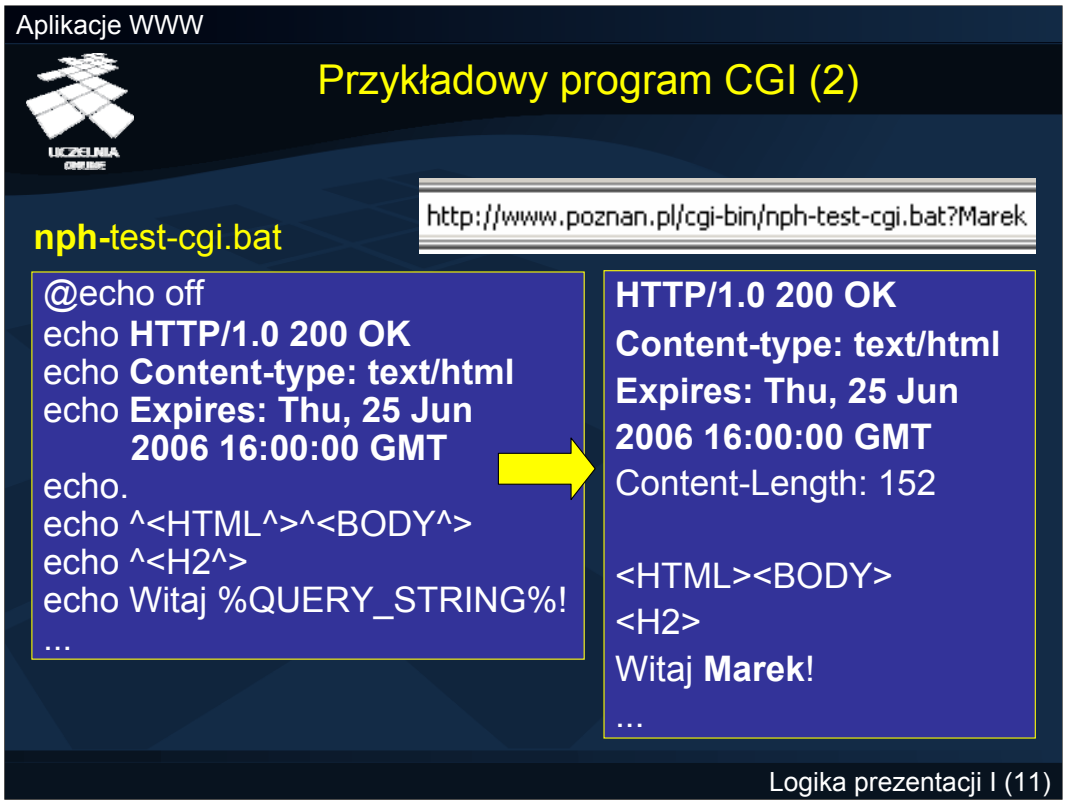
Na slajdzie przedstawiono przykład programu CGI nie generującego nagłówka odpowiedzi HTTP. Program został zaimplementowany w języku skryptów BAT, jest uruchamiany w odpowiedzi na żądanie HTTP zawierające parametr, którym jest imię użytkownika końcowego. Struktura programu jest następująca:

1. Program CGI wysyła do standardowego wyjścia dyrektywę "Content-type", która wskazuje serwerowi HTTP format, w jakim zostanie przekazane ciało odpowiedzi. Należy podkreślić, że w omawianym przypadku "Content-type" jest dyrektywą dla serwera HTTP, a nie polem nagłówka odpowiedzi o tej samej nazwie. Rozróżnienie to nie ma jednak żadnych konsekwencji funkcjonalnych.

2. Program CGI wysyła do standardowego wyjścia pusty wiersz, oznaczający, że zakończono wysyłanie dyrektyw dla serwera HTTP i przystąpiono do generowania treści ciała odpowiedzi HTTP.

3. Program CGI wysyła treść ciała odpowiedzi HTTP. Treść tę stanowi dokument HTML. Ponieważ znaki "<" i ">" pełnią rolę znaków specjalnych w MS DOS, to zostały poprzedzone znakiem sterującym "^". Wewnątrz generowanego dokumentu HTML umieszczono wartość zmiennej środowiskowej QUERY_STRING, która wg specyfikacji CGI zawiera parametry żądania HTTP typu GET. Ponadto, program CGI wykonuje polecenie systemowe DIR, a jego wynik również umieszcza w generowanym dokumencie HTML.

Po prawej stronie slajdu przedstawiono przykładową odpowiedź HTTP powstałą w wyniku pracy programu CGI. Zauważmy, że ta odpowiedź zawiera nagłówek, który został wygenerowany automatycznie przez serwer HTTP, częściowo na podstawie wskazówek (dyrektyw) programu CGI. W przykładowym wywołaniu użytkownik dołączył swoje imię (Maciej) do adresu URL żądania: "http://www.poznan.pl/cgi-bin/test-cgi.bat?Maciej".



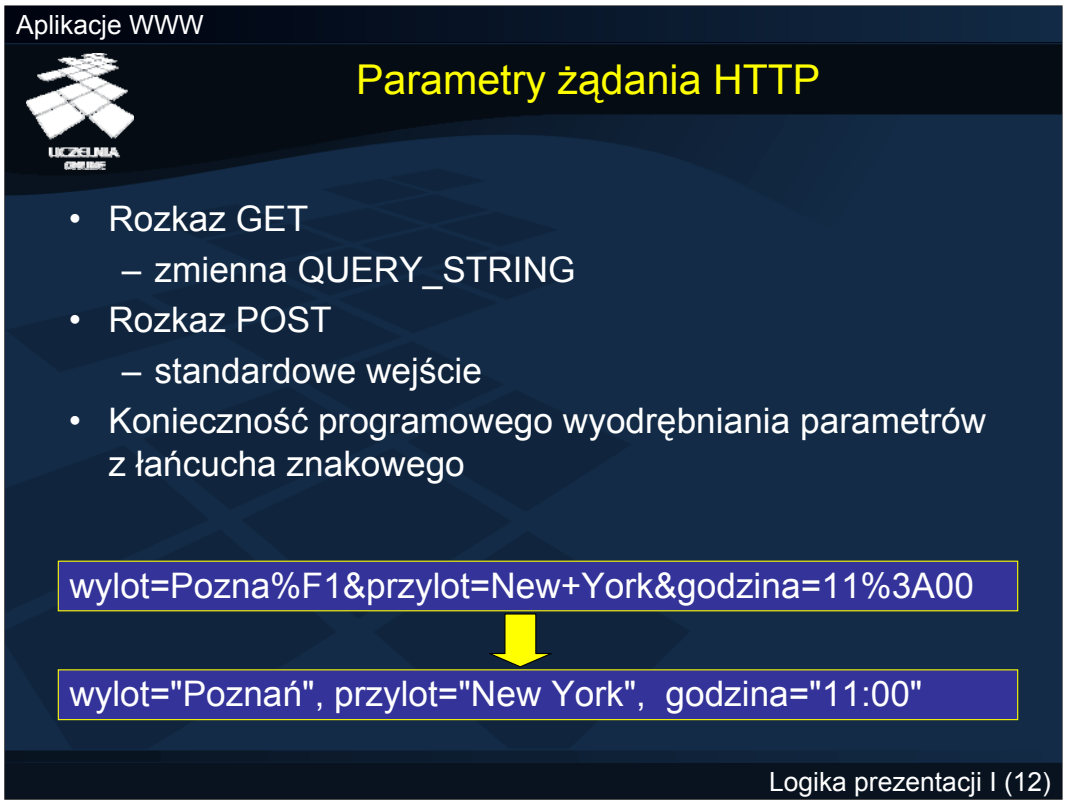
Na tym slajdzie przedstawiono przykład analogicznego programu CGI generującego pełen nagłówek odpowiedzi HTTP. Zauważmy, że nazwa pliku programu rozpoczyna się od słowa "nph-". Struktura przykładowego programu jest następująca:

1. Program CGI wysyła do standardowego wyjścia zbiór pól nagłówka odpowiedzi HTTP. Pola te są bez zmian przekazywane przez serwer HTTP do klienta HTTP.

2. Program CGI wysyła do standardowego wyjścia pusty wiersz, oznaczający, że zakończono wysyłanie pól nagłówka HTTP i przystąpiono do generowania treści ciała odpowiedzi HTTP.

3. Program CGI wysyła treść ciała odpowiedzi HTTP. Treść tę stanowi dokument HTML. Ponieważ znaki "<" i ">" pełnią rolę znaków specjalnych w MS DOS, to zostały poprzedzone znakiem sterującym "^". Wewnątrz generowanego dokumentu HTML umieszczono wartość zmiennej środowiskowej QUERY_STRING, która wg specyfikacji CGI zawiera parametry żądania HTTP typu GET. Ponadto, program CGI wykonuje polecenie systemowe DIR, a jego wynik również umieszcza w generowanym dokumencie HTML.

Po prawej stronie slajdu przedstawiono przykładową odpowiedź HTTP powstałą w wyniku pracy programu CGI. Zauważmy, że ta odpowiedź zawiera pola nagłówka HTTP, które pochodzą bezpośrednio od programu CGI, zostały jednak wzbogacone o pole "Content-length", zawierające rozmiar ciała odpowiedzi HTTP. Rozmiar ten jest automatycznie wyznaczany przez serwer HTTP. W przykładowym wywołaniu użytkownik dołączył swoje imię (Marek) do adresu URL żądania: "http://www.poznan.pl/cgi-bin/nph-test-cgi.bat?Marek".

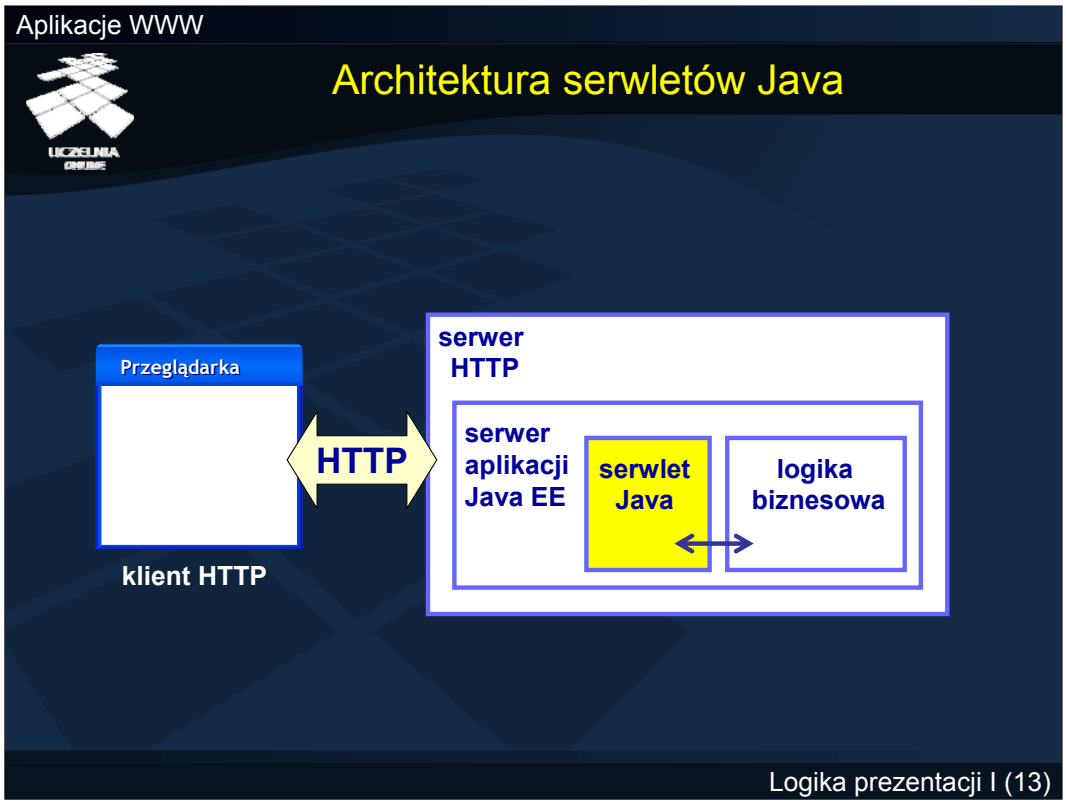


Programy CGI często pobierają parametry wywołania od użytkowników końcowych. Parametry te są najczęściej przekazywane za pomocą jednej z dwóch metod:

1. Metoda HTTP GET, w ramach której klient HTTP dołącza parametry wywołania do adresu URL żądania, oddzielając je znakiem "?", a kolejne parametry separując od siebie znakiem "&". Takie parametry są przekazywane programowi CGI za pośrednictwem predefiniowanej zmiennej środowiskowej o nazwie QUERY_STRING.

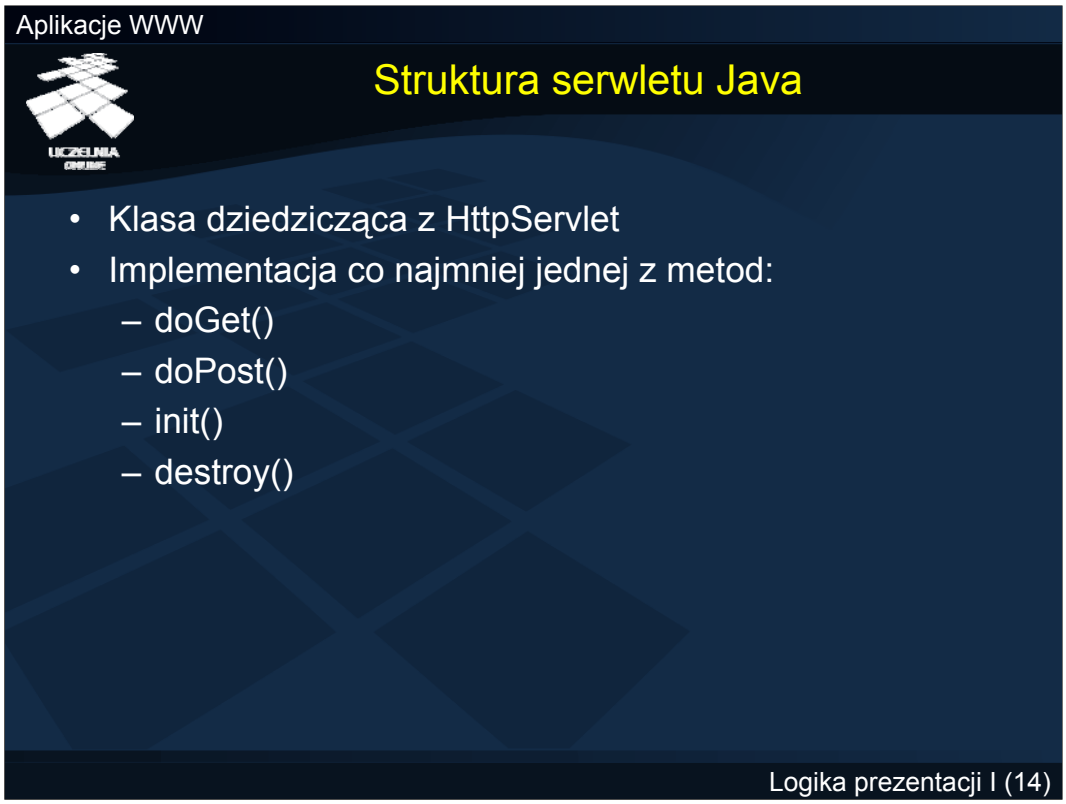
2. Metoda HTTP POST, w ramach której klient HTTP umieszcza parametry wywołania w ciele żądania HTTP. Takie parametry są przekazywane programowi CGI za pośrednictwem mechanizmu systemowego wejścia standardowego.

Niezależnie od wybranej metody przekazywania parametrów wywołania, program CGI otrzymuje pojedynczy łańcuch znakowy zawierający listę wszystkich parametrów wraz z ich wartościami. Zadaniem programisty jest wyodrębnienie z takiego łańcucha poszczególnych nazw i wartości parametrów.



Serwlety Java to technologia implementacyjna wchodząca w skład obszernej dystrybucji Java Platform, Enterprise Edition (Java EE, dawniej J2EE). Serwlet Java jest programem Java umiejscowionym po stronie serwera HTTP, służącym do automatycznego generowania dokumentów (np. HTML) stanowiących odpowiedzi na żądania HTTP. Serwlety Java wymagają dedykowanego środowiska uruchomieniowego nazywanego serwerem aplikacji Java EE, który odpowiada m.in. za zarządzanie pamięcią operacyjną, kontrolę dostępu, komunikację z bazami danych, itp. Przykładami serwerów Java EE są: BEA Weblogic (www.beasys.com), Borland Visibroker (www.borland.com), Caucho Resin (www.caucho.com), JBOSS (www.jboss.org), IBM WebSphere (www.ibm.com), Jakarta Tomcat (jakarta.apache.org), Oracle Application Server (www.oracle.com), Orion (www.orionserver.com), Sun Java Web Server (www.sun.com), W3 Jigsaw (www.w3.org), itd.

Serwlety Java mogą odwoływać się do kodu logiki biznesowej zaimplementowanego w formie obiektów Java Beans, Enterprise Java Beans, itp.



Implementacja serwletu Java polega na utworzeniu klasy dziedziczącej z klasy bibliotecznej javax.servlet.http.HttpServlet i zaimplementowaniu w niej co najmniej jednej z następujących metod:

• public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

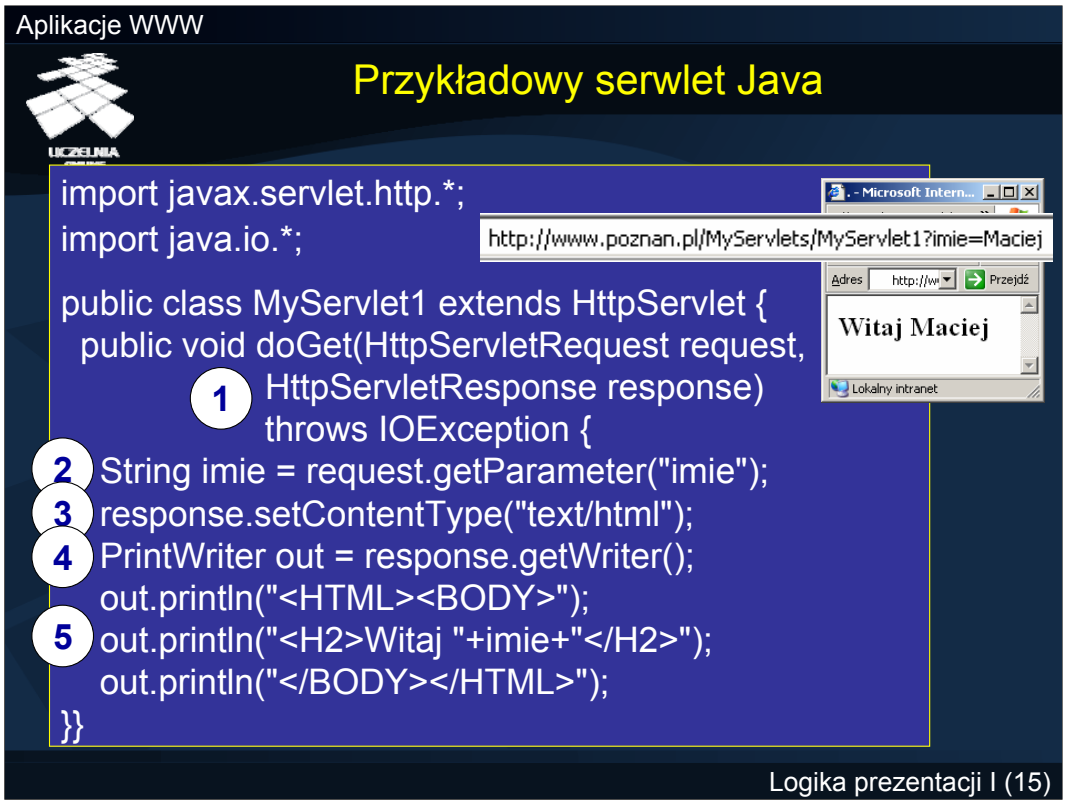
• public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

oraz opcjonalnie:

• public void init(ServletConfig config) throws ServletException

• public void destroy()

Metoda doGet() jest wywoływana w odpowiedzi na żądanie HTTP typu GET. Metoda doPost() jest wywoływana w odpowiedzi na żądanie HTTP typu POST. Metoda init() jest wywoływana w chwili ładowania serwletu do pamięci operacyjnej serwera aplikacji. Metoda destroy() jest wywoływana w chwili usuwania serwletu z pamięci operacyjnej serwera aplikacji. Szczegółowy cykl życia serwletu Java zostanie przedstawiony w dalszej części wykładu.



Na slajdzie zamieszczono kod źródłowy przykładowego serwletu Java. Jego klasa nazywa się MyServlet1 i zawiera tylko jedną metodę: doGet(). Metoda ta będzie wywoływana w odpowiedzi na każde otrzymane żądanie HTTP. Oto najważniejsze składniki kodu źródłowego:

1. Metoda doGet() przyjmuje dwa obiektowe argumenty: request i response. Obiekt request zawiera metadane żądania, natomiast obiekt response służy do przekazywania odpowiedzi. W pewnym uproszczeniu, obiekty te stanowią jednokierunkowe kanały komunikacyjne pomiędzy serwerem aplikacji a serwletem Java. Metoda doGet() propaguje wyjątki typu IOException, które mogą być generowane przez wywołanie response.getWriter().

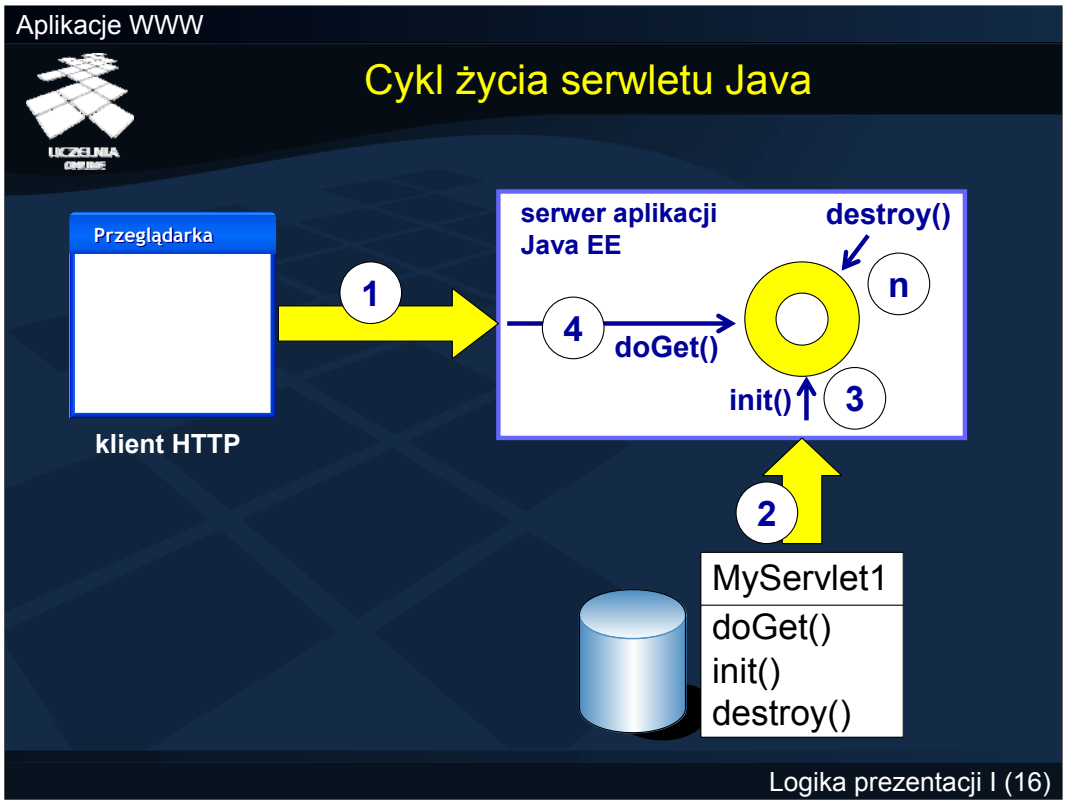
2. Serwlet pobiera wartość parametru wejściowego o nazwie "imie". Parametr ten jest przekazywany przez użytkownika końcowego i zawiera jego imię. Do odczytu wartości parametrów wejściowych służy metoda getParameter() obiektu request.

3. Serwlet przekazuje serwerowi aplikacji informację o formacie generowanego dokumentu wynikowego. Na tej podstawie serwer aplikacji wygeneruje nagłówek odpowiedzi HTTP zawierający pole "Content-type".

4. Serwlet powołuje nowy obiekt klasy PrintWriter, stanowiący wyjściowy strumień alfanumeryczny, poprzez który zostanie przekazana treść ciała odpowiedzi HTTP.

5. Serwlet wysyła treść ciała odpowiedzi HTTP w formie dokumentu HTTP. W treść dokumentu wpleciono wartość odebranego parametru wywołania. Zapis do strumienia wyjściowego odbywa się za pomocą metody println() obiektu out.

W celu uruchomienia serwletu Java, użytkownik końcowy formułuje żądanie zawierające adres URL klasy serwletu. Adres ten zwykle zawiera nazwę klasy oraz zdefiniowaną przez administratora wirtualną ścieżkę dostępową. Parametry wejściowe mogą być przekazywane za pomocą dowolnej z metod protokołu HTTP (GET lub POST).



Cykl życia serwletu Java przebiega według następujących kroków:

1. Klient HTTP przekazuje żądanie HTTP do serwera HTTP. Żądanie jest kierowane do serwera aplikacji Java EE.

2. Serwer aplikacji analizuje adres URL żądania HTTP, pobiera z dysku klasę wskazanego serwletu i tworzy jej obiekt (tzw. obiekt serwletu).

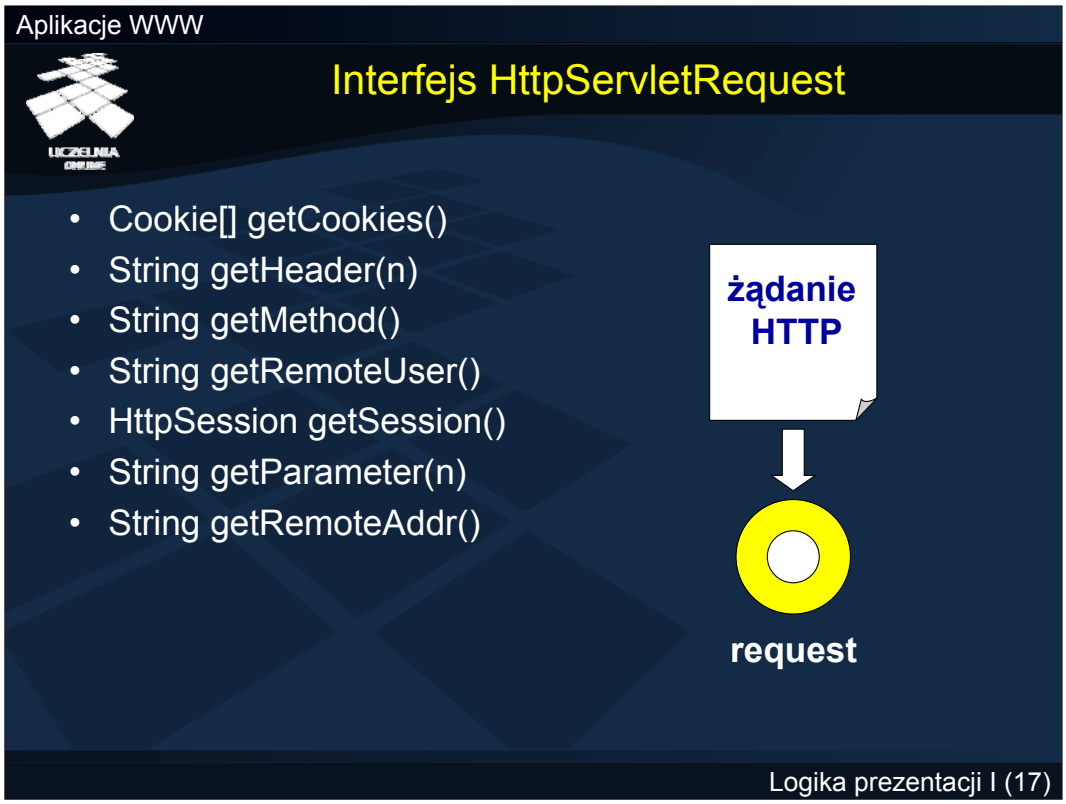
3. Serwer aplikacji wywołuje metodę init() obiektu serwletu.

4. Serwer aplikacji wywołuje metodę doGet() obiektu serwletu. Dokument będący wynikiem działania metody doGet() jest przekazywany klientowi HTTP. Obsługa żądania została zakończona.

Po zakończeniu obsługi żądania HTTP obiekt serwletu pozostaje w pamięci operacyjnej. W związku z tym, podczas obsługi kolejnych żądań nie jest konieczne pobieranie klasy serwletu z dysku ani tworzenie jej nowego obiektu. Kolejne żądania nie powodują też wykonania metody init(), a jedynie doGet().

Obiekt serwletu zwykle pozostaje w pamięci operacyjnej aż do zatrzymania serwera aplikacji. W chwili usuwania obiektu serwletu z pamięci operacyjnej wykonywana jest jego metoda destroy().

Metody init() i destroy() służą zwykle do realizacji zadań inicjacyjnych i końcowych, np. nawiązanie/rozłączenie połączenia z bazą danych, odczyt plików konfiguracyjnych, otwarcie/zamknięcie pliku dziennika.



Metoda doGet() serwletu otrzymuje dwa obiektowe argumenty: request i response. Obiekt request reprezentuje żądanie HTTP, a obiekt response - odpowiedź HTTP. W pierwszej kolejności omówimy strukturę obiektu request.

Obiekt request implementuje metody zgodnie z interfejsem HttpServletRequest. Najważniejsze z nich zostały przedstawione na slajdzie. Ich znaczenie jest następujące:

• Cookie[] getCookies() - udostępnia tablicę zmiennych Cookies przekazanych przez klienta HTTP,

• String getHeader(n) - udostępnia wartość wskazanego pola nagłówka żądania HTTP,

• String getMethod() - wskazuje komendę HTTP użytą w żądaniu, np. GET, POST,

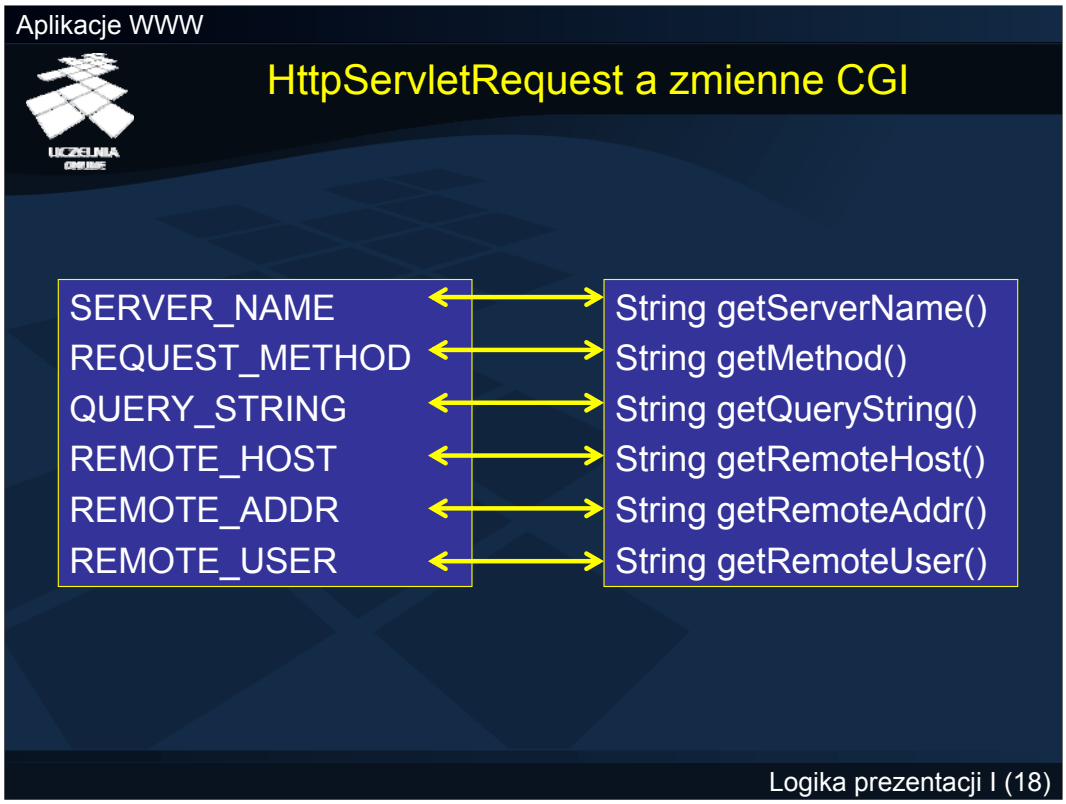
• String getRemoteUser() - zwraca nazwę, pod jaką użytkownik końcowy został uwierzytelniony,

• HttpSession getSession() - odczytuje lub tworzy obiekt aktualnej sesji (sesjami serwletów zajmiemy się w dalszej części wykładu),

• String getParameter(n) - odczytuje wartość wskazanego parametru żądania HTTP,

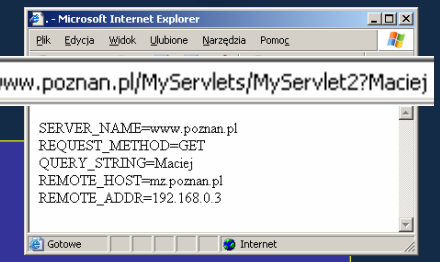
• String getRemoteAddr() - zawiera adres IP komputera, z którego pochodzi żądanie HTTP.

Opis pozostałych metod interfejsu HttpServletRequest znajduje się w dokumentacji Java EE SDK.



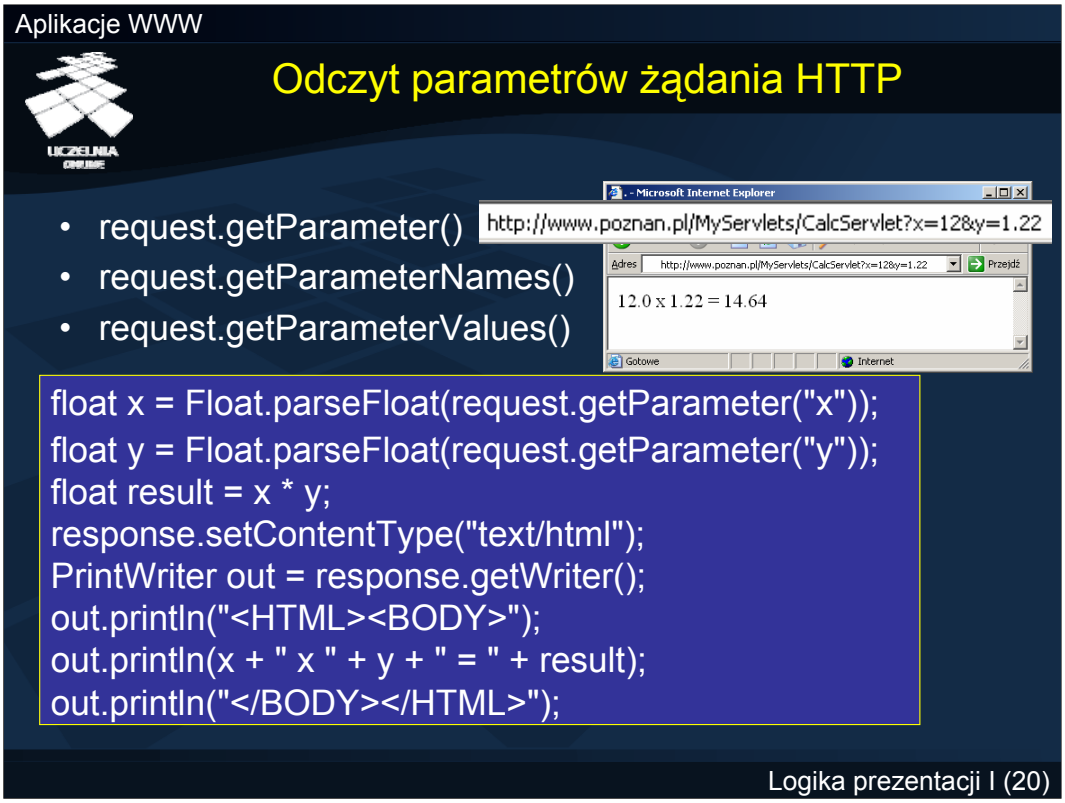
Część interfejsu HttpServletRequest zachowuje kompatybilność z semantyką zmiennych CGI, dzięki czemu funkcjonalność serwletów Java może być nadzbiorem funkcjonalności programów CGI. Na slajdzie przedstawiono wybrane zmienne CGI i odpowiadające im metody interfejsu HttpServletRequest.

HttpServletRequest - przykład




```
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<HTML><BODY>");
out.println("SERVER_NAME="+request.getServerName()+"<BR>");
out.println("REQUEST_METHOD="+request.getMethod()+"<BR>");
out.println("QUERY_STRING="+request.getQueryString()+"<BR>");
out.println("REMOTE_HOST="+request.getRemoteHost()+"<BR>");
out.println("REMOTE_ADDR="+request.getRemoteAddr());
out.println("</BODY></HTML>");
```


Na slajdzie przestawiono fragment kodu przykładowego serwletu Java wyświetlającego metadane żądania HTTP w sposób analogiczny do programu CGI przedstawionego we wcześniejszej części wykładu.

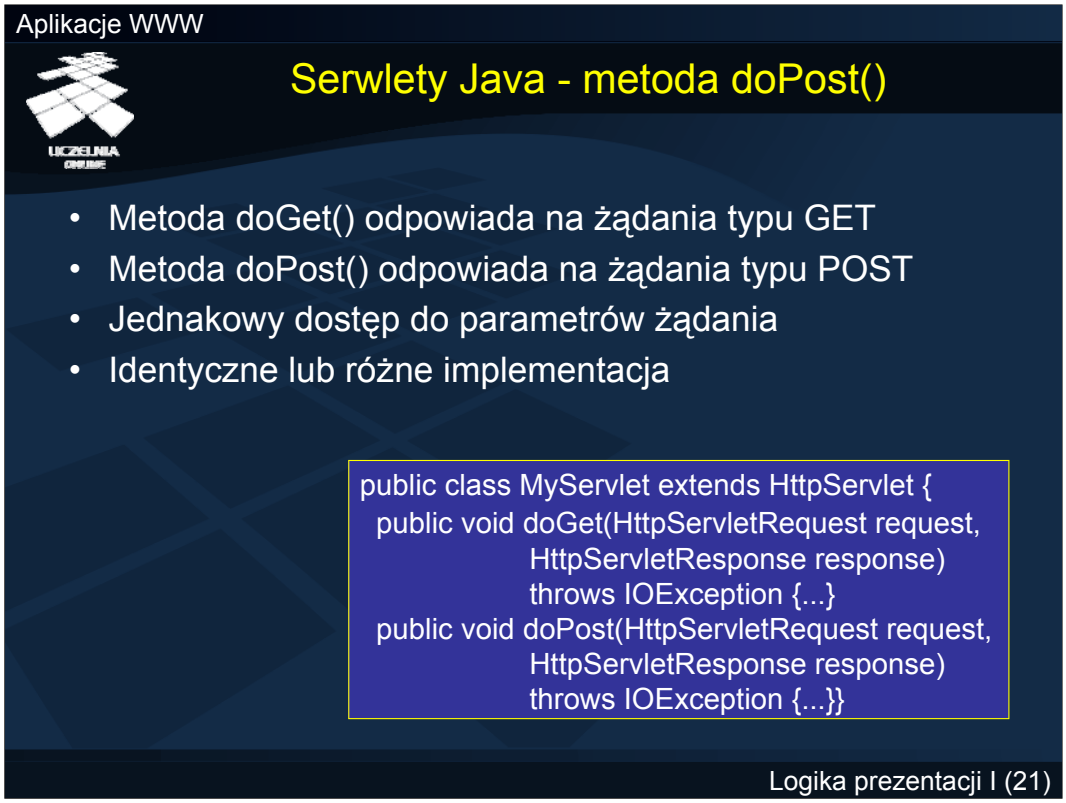


Programista serwletów Java może w nieskomplikowany sposób sięgać do parametrów żądania HTTP, które spowodowało uruchomienie serwletu. Do tego celu służą m.in. następujące metody interfejsu HttpServletRequest:

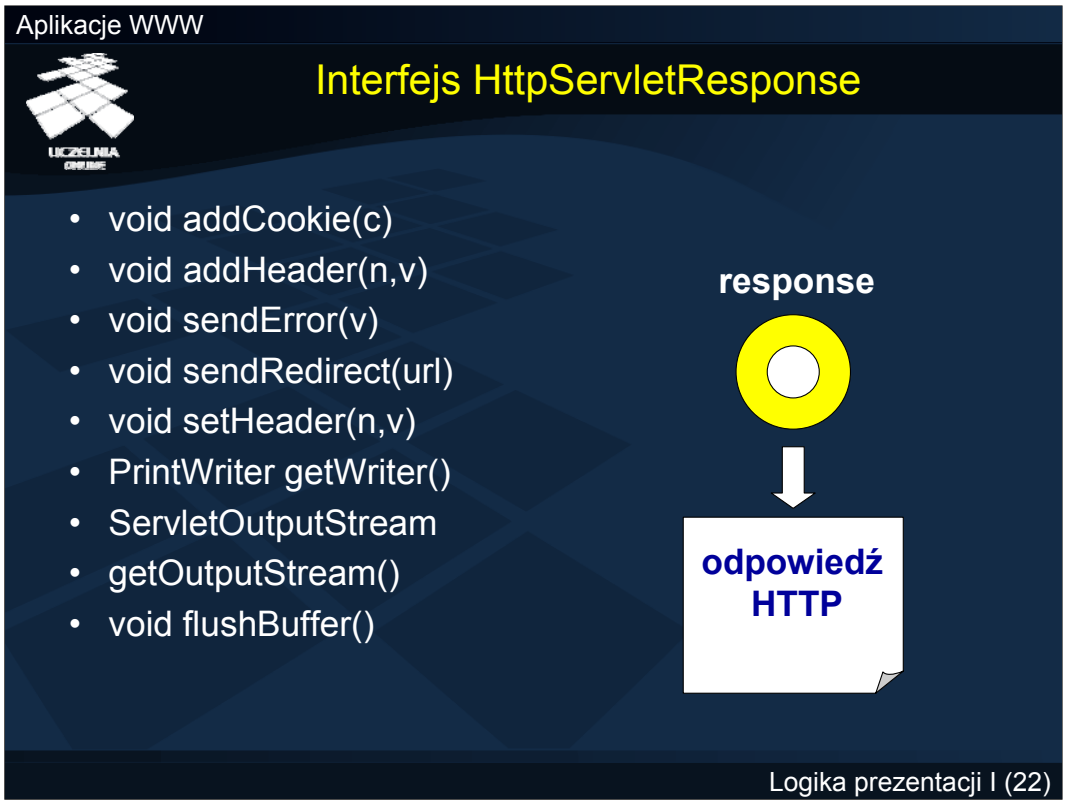
- String getParameter(n) - zwraca łańcuch znakowy zawierający wartość parametru o podanej nazwie,
- Enumeration getParameterNames() - zwraca zbiór nazw otrzymanych parametrów,
- Enumeration getParameterValues() - zwraca zbiór wartości otrzymanych parametrów.

Powyższe metody udostępniają wartości parametrów przekazanych zarówno za pomocą metody GET, jak i POST.

Na slajdzie przedstawiono fragment przykładowego serwletu Java, który pobiera dwa parametry, konwertuje ich znakowe wartości do typu zmiennoprzecinkowego, a następnie wyświetla ich iloraz. Należy podkreślić, że wartości parametrów żądania HTTP są zawsze łańcuchami znakowymi. Ponadto, nazwy parametrów są wrażliwe na wielkość znaków.



W zależności od typu żądania HTTP, obiekt serwletu Java wykonuje jedną z dwóch metod: doGet(), jeżeli otrzymano żądane HTTP typu GET, lub doPost(), jeżeli otrzymano żądanie HTTP typu POST. Obie metody mają identyczne sygnatury. Programista może zaimplementować jedną z nich, obie w jednakowy sposób lub obie w różny sposób. Niezależnie jednak od rodzaju implementacji, dostęp do parametrów żądania odbywa się w taki sam sposób - np. za pomocą metody request.getParameter().



Drugim argumentem metody doGet() lub doPost() jest obiekt response, reprezentujący odpowiedź HTTP. Obiekt reponse implementuje metody zgodne z interfejsem HttpServletResponse. Najważniejsze z nich zostały przedstawione na slajdzie. Ich znaczenie jest następujące:

• void addCookie(c) - umieszcza w nagłówku odpowiedzi HTTP zmienną Cookie,

• void addHeader(n,v) - dołącza pole nagłówka do odpowiedzi HTTP; jeżeli takie pole zostało już dołączone wcześniej, to staje się wielowartościowym,

• void sendError(v) - określa kod zwrotny odpowiedzi HTTP,

• void sendRedirect(url) - wysyła kod zwrotny 302 (Moved Temporarily), powodujący przekierowanie klienta HTTP pod nowy adres URL,

• void setHeader(n,v) - nadaje wartość wskazanemu polu nagłówka odpowiedzi HTTP, nadpisując ew. wartość dotychczasową,

• PrintWriter getWriter() - zwraca obiekt wyjściowego strumienia alfanumerycznego, poprzez który zostanie przekazana treść ciała odpowiedzi HTTP; może służyć np. do wysłania dokumentu HTML

• ServletOutputStream getOutputStream() - zwraca obiekt wyjściowego strumienia binarnego, poprzez który zostanie przekazana treść ciała odpowiedzi HTTP; może służyć np. do wysłania obrazu GIF

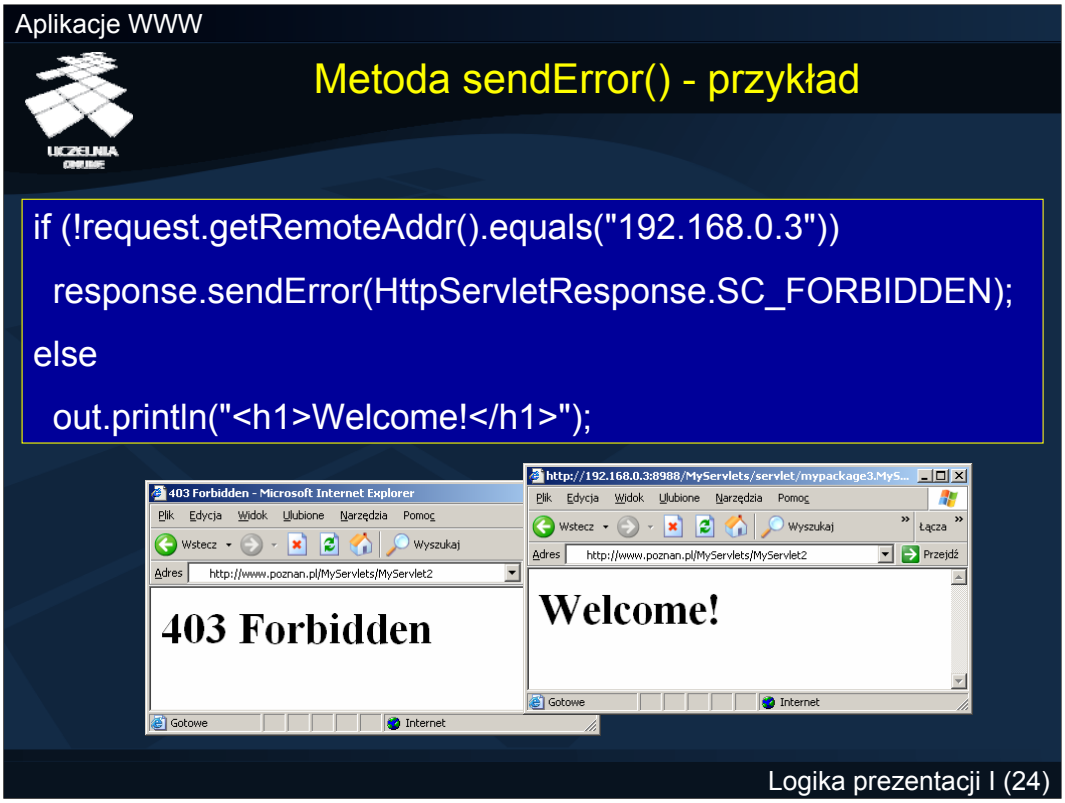
• void flushBuffer() - wymusza wysłanie buforowanej odpowiedzi HTTP do klienta HTTP; jeśli ta metoda nie zostanie wywołana, wtedy odpowiedź zostanie wysłana po zakończeniu metody doGet() lub doPost().

Opis pozostałych metod interfejsu HttpServletResponse znajduje się w dokumentacji Java EE SDK.

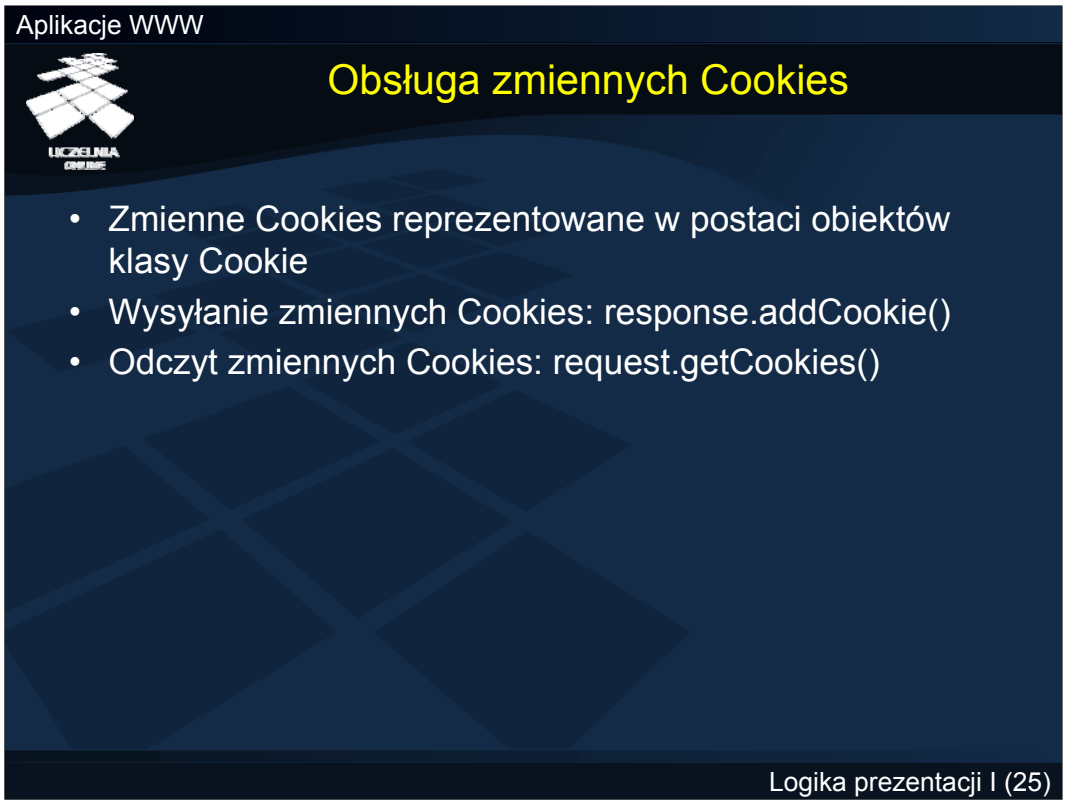
Metoda sendError() - kody zwrotne

SC_BAD_REQUEST	(400) Niewłaściwy format żądania
SC_UNAUTHORIZED	(401) Wymagana autoryzacja klienta
SC_FORBIDDEN	(403) Odmowa dostępu
SC_NOT_FOUND	(404) Dokument nie znaleziony
SC_GONE	(410) Dokument zmienił adres, a nowa lokalizacja nie jest znana
SC_SERVICE_UNAVAILABLE	(503) System jest chwilowo przeciążony lub niedostępny
SC_NOT_MODIFIED	(304) Dokument nie uległ modyfikacji od poprzedniego dostępu

Metoda sendError() obiektu response służy do określenia kodu zwrotnego umieszczanego w odpowiedzi HTTP. Argumentem tej metody może być liczba całkowita lub nazwa stałej zadeklarowanej przez interfejs HttpServletResponse. Na slajdzie przedstawiono nazwy przykładowych stałych reprezentujących najczęściej wykorzystywane kody zwrotne. Opis pozostałych stałych interfejsu HttpServletResponse znajduje się w dokumentacji Java EE SDK.



Na slajdzie przedstawiono fragment kodu źródłowego serwletu Java, który wykorzystuje metodę sendError() w celu poinformowania klienta HTTP o braku autoryzacji. Jeżeli adres IP klienta HTTP jest równy "192.168.0.3", to wysyłany jest dokument HTML zawierający napis "Welcome". W przeciwnym przypadku do odpowiedź HTTP zawiera kod zwrotny 403 (Forbidden), oznaczający odmowę dostępu do dokumentu.



Technologia serwletów Java umożliwia programiście wygodne operowanie zmiennymi Cookies (patrz wykład pt. "Protokół HTTP"). Każda zmienna Cookie może być reprezentowana przez obiekt bibliotecznej klasy Cookie (javax.servlet.http.Cookie). Do wysyłania zmiennych Cookies do klienta HTTP służy metoda addCookies() obiektu response, natomiast do odczytu zmiennych Cookies odebranych od klienta HTTP służy metoda getCookies() obiektu request. Obie te metody w istocie operują na nagłówkach żądań i odpowiedzi HTTP.

Klasa Cookie

`Cookie(n,s)`	Tworzy zmienną o nazwie *n* i wartości *s*
`String getDomain()` `void setDomain(s)`	Odczytuje/ustawia adres domenowy, dla którego przeznaczona jest zmienna
`String getMaxAge()` `void setMaxAge(v)`	Odczytuje/ustawia czas życia zmiennej, liczony w sekundach (-1 -> ulotne)
`String getName()` `void setName(s)`	Odczytuje/ustawia nazwę zmiennej
`String getPath()` `void setPath(s)`	Odczytuje/ustawia prefiks ścieżki URL na serwerze, dla której przeznaczona jest zmienna
`String getValue()` `void setValue(s)`	Odczytuje/ustawia wartość zmiennej

Klasa Cookie służy do reprezentowania zmiennych Cookies za pomocą obiektów Java. Każdy obiekt implementuje metody przedstawione na slajdzie. Za pomocą tych metod programista może odczytać lub określić: nazwę, wartość, czas życia oraz zasięg zmiennej. Podczas wysyłania zmiennej Cookie do klienta HTTP następuje serializacja obiektu Java do postaci łańcucha znakowego umieszczanego w nagłówku odpowiedzi HTTP, w polu Set-Cookie. Odczyt zmiennej Cookie polega na zbudowaniu obiektu Java na podstawie pola Cookie nagłówka żądania HTTP.

Na slajdzie przedstawiono fragment przykładowego kodu źródłowego serwletu Java, który wysyła do klienta HTTP dwie nowe zmienne Cookies: zmienną "Imie" o wartości "Maciej" i zmienną "Miasto" o wartości "Poznań". Ponadto, dla zmiennej "Imie" określono czas życia wynoszący 24 godziny (24x60x60 sekund). Czas życia zmiennej "Miasto" jest domyślny, co oznacza, że jest ona zmienną ulotną, traconą po zakończeniu pracy klienta HTTP. Znaczenie wierszy kodu źródłowego jest następujące:

1. Tworzony jest obiekt o nazwie myCookie1, reprezentujący zmienną Cookie o nazwie "Imie". Jednocześnie określana jest wartość zmiennej.

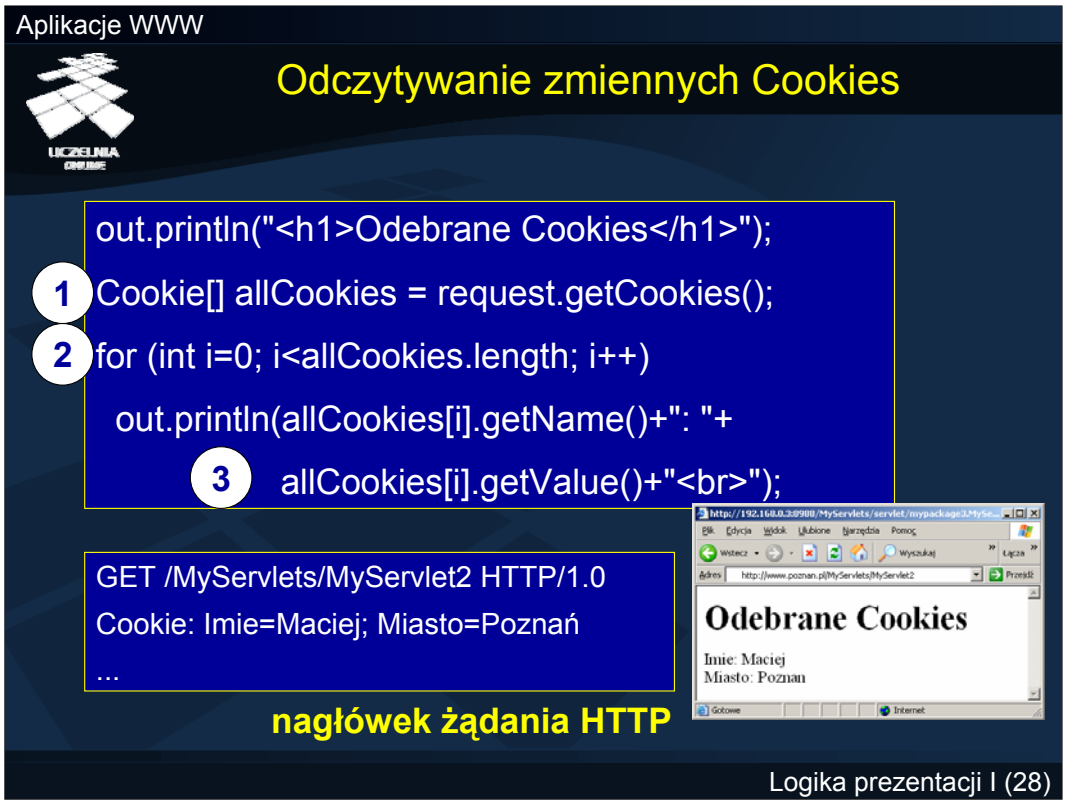
2. Tworzony jest obiekt o nazwie myCookie2, reprezentujący zmienną Cookie o nazwie "Miasto". Jednocześnie określana jest wartość zmiennej.

3. Dla obiektu myCookie1, reprezentującego zmienną "Imie", wywoływana jest metoda setMaxAge(), służąca do określenia czasu życia zmiennej.

4. Do nagłówka odpowiedzi HTTP dodawana jest zmienna "Imie", reprezentowana przez obiekt myCookie1.

5. Do nagłówka odpowiedzi HTTP dodawana jest zmienna "Miasto", reprezentowana przez obiekt myCookie1.

W dolnej części slajdu przedstawiono fragment nagłówka odpowiedzi HTTP wygenerowanego przez omawiany serwlet Java. W nagłówku występują pola zawierające definicje zmiennych Cookies. Zmienne te zostaną zapamiętane przez klienta HTTP i będą dołączane do przyszłych żądań HTTP.



Na slajdzie przedstawiono fragment przykładowego kodu źródłowego serwletu Java, który wyświetla nazwy i wartości wszystkich zmiennych Cookies otrzymanych od klienta HTTP w nagłówku żądania HTTP. W dolnej części slajdu zamieszczono fragment przykładowego nagłówka żądania HTTP, zawierającego definicję dwóch zmiennych Cookies: zmiennej "Imie" o wartości "Maciej" i zmiennej "Miasto" o wartości "Poznań".

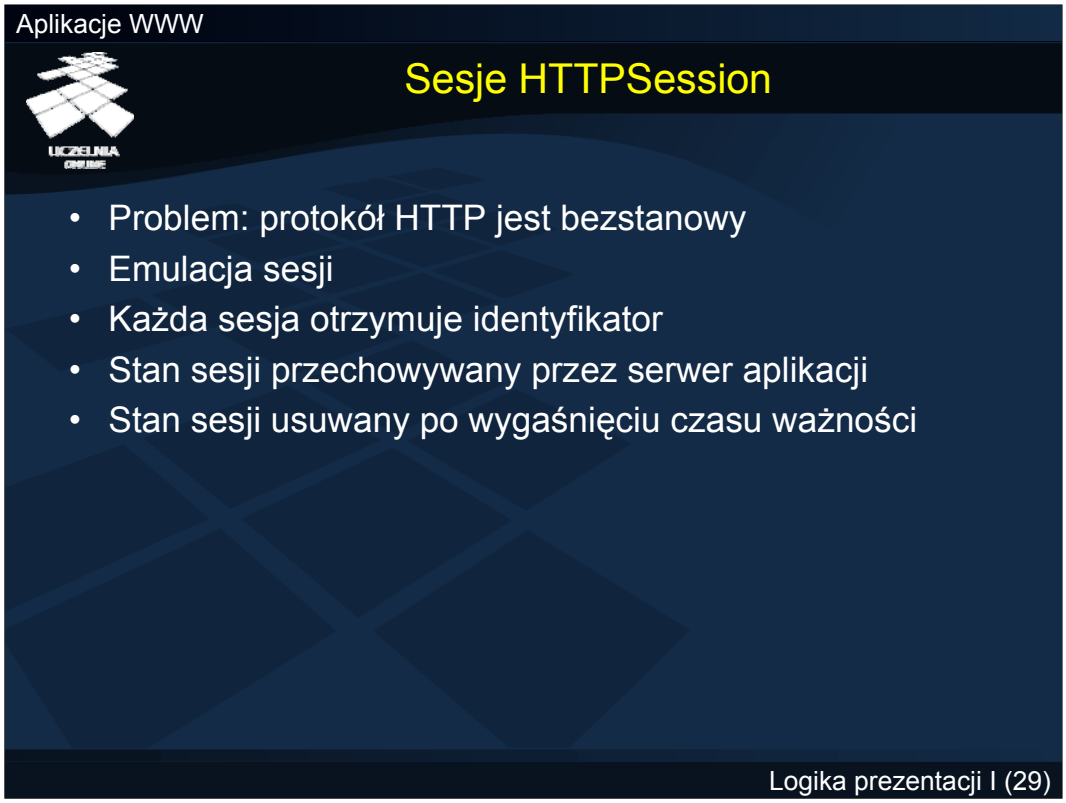
Odczyt wszystkich zmiennych Cookies otrzymanych od klienta HTTP jest możliwy za pomocą metody getCookies() obiektu request. Metoda ta zwraca tablicę obiektów klasy Cookie. Niestety, obiekt request nie oferuje możliwości odczytu pojedynczej zmiennej Cookie o podanej nazwie. Zmienną taką należy wyszukać programowo w tablicy zwróconej przez metodę getCookies().

Znaczenie wierszy przedstawionego kodu źródłowego jest następujące:

1. Wywołanie metody getCookies() obiektu request w celu pobrania tablicy obiektów klasy Cookie, reprezentujących wszystkie zmienne przekazane przez klienta HTTP w nagłówku żądania HTTP.

2. Pętla programowa krocząca po wszystkich elementach otrzymanej tablicy obiektów. W każdej iteracji pętli wybierany jest pojedynczy obiekt z tablicy, reprezentujący pojedynczą zmienną przekazaną przez klienta HTTP.

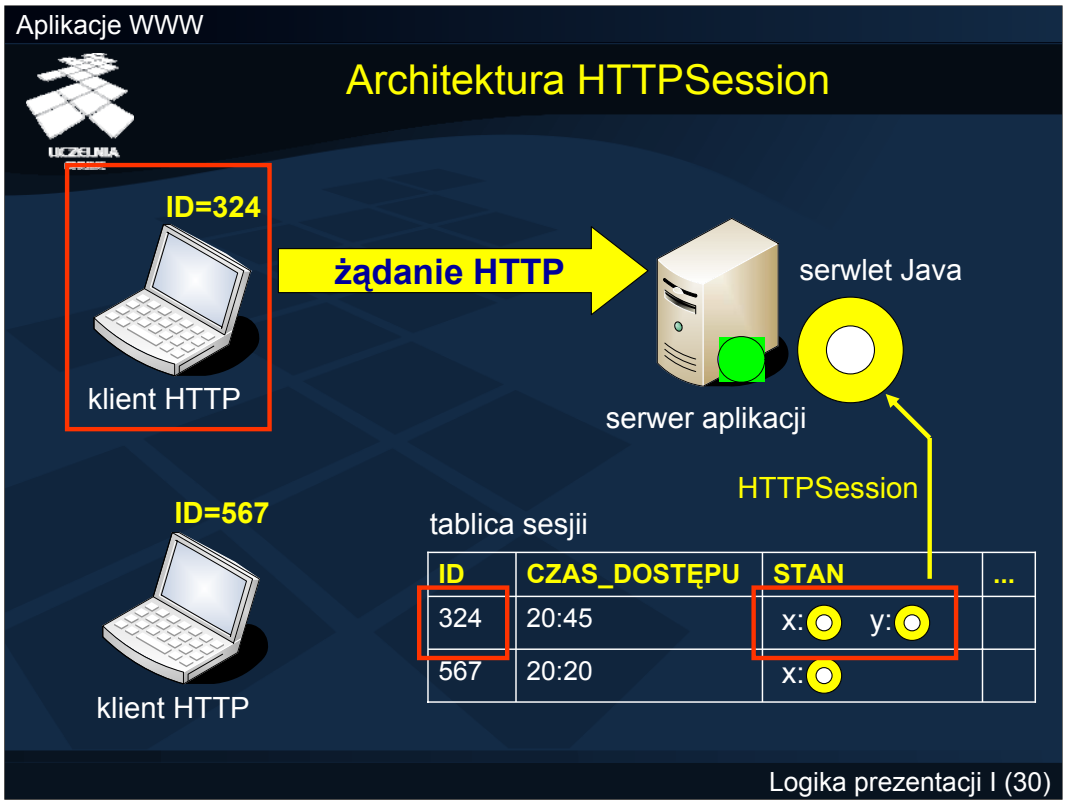
3. Dla bieżącego obiektu pobierana jest nazwa i wartość reprezentowanej przez niego zmiennej Cookie. Nazwa i wartość zmiennej są umieszczane w wynikowym dokumencie HTML.



Protokół HTTP został zaprojektowany jako bezstanowy, co oznacza, że serwer HTTP nie potrafi rozpoznać czy określone żądania HTTP pochodzą od tego samego użytkownika końcowego, czy od użytkowników niezależnych. Ponieważ wiele zastosowań serwletów Java wymaga korelowania żądań HTTP, dlatego zaproponowano mechanizm emulacji sesji nazwany HTTPSession.

Idea mechanizmu HTTPSession opiera się na założeniu, że serwer aplikacji generuje dla każdego klienta HTTP niepowtarzalny identyfikator, nazywany identyfikatorem sesji. Identyfikator ten jest zwykle zapisywany w zmiennej Cookie, dzięki czemu powraca do serwera aplikacji przy każdym kolejnym żądaniu HTTP zgłaszanym przez tego samego klienta HTTP. Po stronie serwera aplikacji znajduje się tzw. tablica sesji, w której z każdą wartością identyfikatora sesji związany jest zbiór programowych obiektów Java reprezentujących stan sesji. Serwlety Java mają możliwość odczytu i zapisu obiektów w tym zbiorze za pośrednictwem tzw. obiektu sesji, będącego obiektem interfejsu javax.servlet.http.HTTPSession. Powiązanie wywołanego serwletu Java ze zbiorem obiektów sesji konkretnego klienta HTTP jest automatycznie realizowane przez serwer aplikacji.

Stan sesji klienta HTTP jest przechowywany przez serwer aplikacji aż do momentu jego jawnego usunięcia przez serwlet Java, bądź po przekroczeniu czasu nieaktywności zdefiniowanego przez administratora serwera aplikacji.

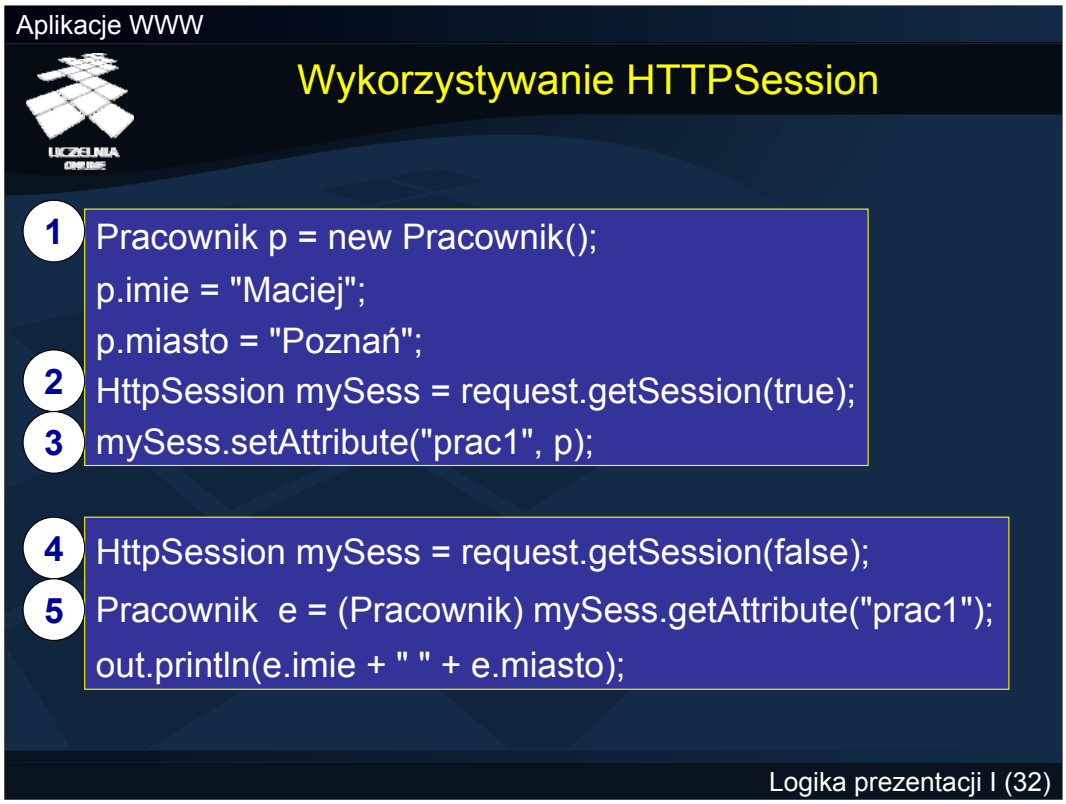


Na slajdzie przedstawiono architekturę mechanizmu HTTPSession. Każdy klient HTTP posiada niepowtarzalny identyfikator sesji. Identyfikatory te są zwykle przydzielane podczas pierwszego kontaktu klienta HTTP z serwerem aplikacji. Górny klient posiada identyfikator o wartości "324", a dolny - "567". Serwer aplikacji przechowuje tablicę sesji, w której każdy wiersz odpowiada jednej wartości identyfikatora sesji, czyli jednemu klientowi HTTP. Gdy klient HTTP wysyła żądanie HTTP do serwera aplikacji, do nagłówka żądania dołącza swój identyfikator sesji. Na tej podstawie serwer aplikacji udostępnia serwletowi Java specjalny obiekt, nazywany obiektem HTTPSession, zawierający zbiór obiektów reprezentujących stan danej sesji. Obiekty stanu należące do sesji innych klientów HTTP są dla serwletu niedostępne ze względów bezpieczeństwa. Serwlet Java może obiekty stanu odczytywać, modyfikować, tworzyć i usuwać. Z każdym obiektem związana jest alfanumeryczna etykieta, np. "x", "y".

Interfejs HTTPSession

`Object getAttribute(n)` `void setAttribute(n,o)`	Zapamiętuje na czas sesji obiekt pod podaną nazwą / odczytuje obiekt o podanej etykiecie
`long getCreationTime()`	Odczytuje czas rozpoczęcia sesji, liczony w ms od 1.01.1970
`String getId()`	Odczytuje jednoznaczny identyfikator sesji
`long getLastAccessedTime()`	Odczytuje czas ostatniej operacji wykonanej w ramach sesji (1.01.1970)
`void invalidate()`	Zamyka sesję i usuwa wszystkie jej obiekty

Serwer aplikacji udostępnia serwletowi Java tzw. obiekt HTTPSession, który jest obiektem Java implementującym metody interfejsu javax.servlet.http.HTTPSession. Na slajdzie przedstawiono najważniejsze metody interfejsu HTTPSession.



Slajd przedstawia fragmenty przykładowego kodu źródłowego serwletów Java wykorzystujących mechanizm HTTPSession. Górny fragment kodu źródłowego umieszcza nowy obiekt stanu sesji. Dolny fragment kodu źródłowego odczytuje zapisany wcześniej obiekt stanu. Znaczenie przedstawionych wierszy kodu źródłowego jest następujące:

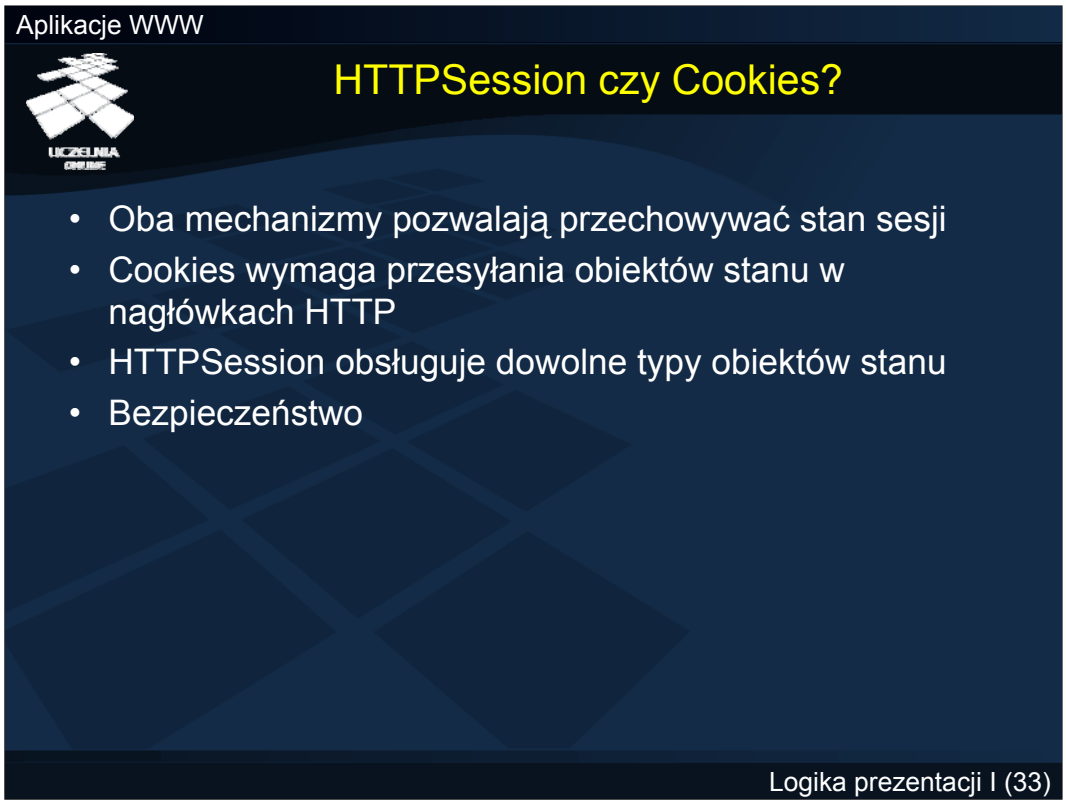
1. Serwlet Java tworzy przykładowy obiekt, który ma reprezentować część stanu sesji. Jest to obiekt klasy Pracownik, posiadający dwa atrybuty, "imie" i "miasto".

2. Serwlet Java pobiera od serwera aplikacji obiekt HTTPSession. Obiekt ten został nazwany mySess. Do pobrania obiektu HTTPSession służy metoda getSession() obiektu request. Parametrem metody getSession() jest wartość logiczna wskazująca, czy klientowi HTTP powinien zostać nadany identyfikator sesji w sytuacji, gdyby było to jego pierwsze żądanie.

3. Do obiektu HTTPSession zapisywany jest nowoutworzony obiekt o etykiecie "prac1". Obiekt ten zostanie przez serwer aplikacji zapisany w tablicy sesji, w wierszu opisanym identyfikatorem sesji bieżącego klienta HTTP.

4. Serwlet Java pobiera od serwera aplikacji obiekt HTTPSession.

5. Z obiektu HTTPSession odczytywany jest obiekt o etykiecie "prac1" i jest on umieszczany w zmiennej e. W tym celu serwer aplikacji wybiera z tablicy sesji wiersz opisany identyfikatorem sesji bieżącego klienta. Następnie wartości atrybutów tego obiektu są umieszczane w wynikowym dokumencie HTML.

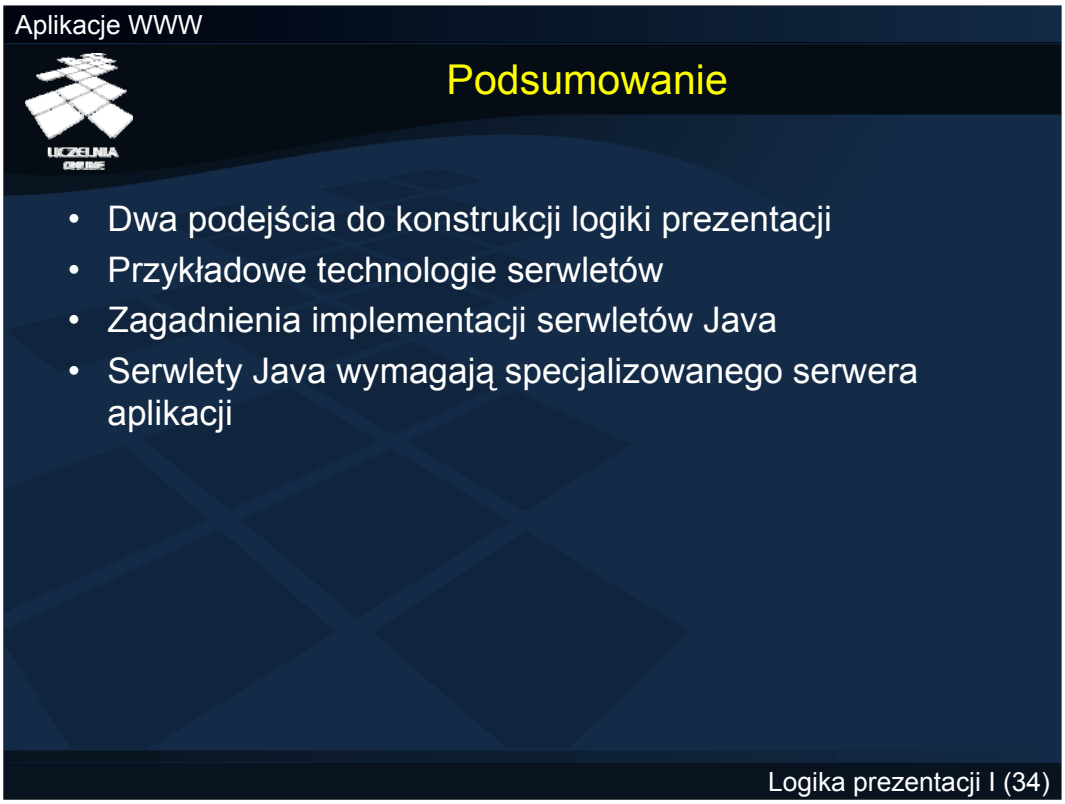


Przedstawiliśmy dotąd dwie alternatywne metody umożliwiające programistom przechowywanie stanu sesji klienta HTTP: metodę opartą na zmiennych Cookies i metodę opartą na sesjach HTTPSession. Dokonajmy teraz ich funkcjonalnego porównania:

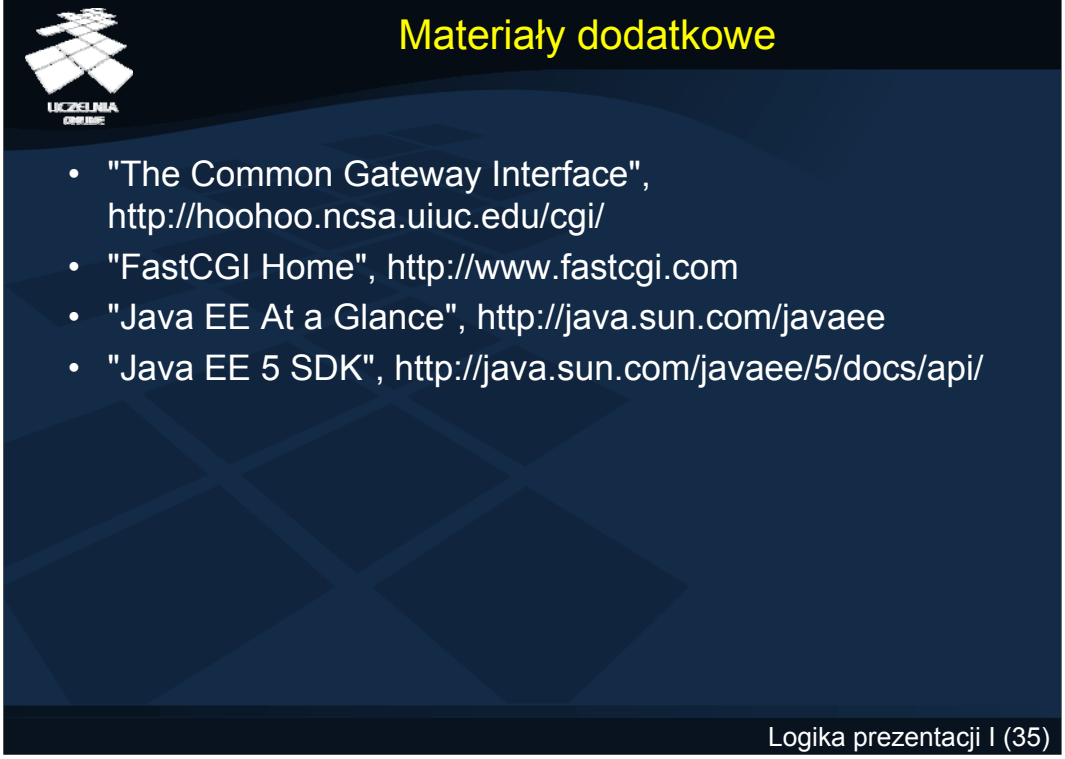
1. Mechanizm zmiennych Cookies wymaga przesyłania wszystkich obiektów stanu w nagłówkach HTTP, natomiast mechanizm HTTPSession przesyła wyłącznie identyfikator sesji. Ma to określone implikacje związane z bezpieczeństwem systemów.

2. Mechanizm zmiennych Cookies obsługuje tylko jeden typ obiektów stanu - łańcuchy znaków alfanumerycznych. Z kolei HTTPSession umożliwia wykorzystywanie praktycznie dowolnych typów obiektowych.

3. Mechanizm HTTPSession może funkcjonować nawet wówczas, gdy klient HTTP nie obsługuje Cookies. Wtedy serwer aplikacji przekazuje identyfikatory sesji np. metodą URL Rewriting.



Istnieją dwa główne podejścia do konstrukcji logiki prezentacji: technologie serwletów i technologie szablonów. Najważniejszymi przykładami technologii serwletów są: programy CGI i serwlety Java. Implementacja programu CGI polega na utworzeniu programu wykonywalnego w systemie operacyjnym. Implementacja serwletu Java polega na utworzeniu klasy Java zawierającej standardowe metody. Serwlety Java mogą korzystać z bogatych bibliotek umożliwiających manipulację nagłówkami HTTP, przetwarzanie zmiennych Cookies, obsługę stanu sesji, itp. Serwlety Java wymagają obecności specjalizowanego serwera aplikacji, nazywanego serwerem Java EE.



•"The Common Gateway Interface", http://hoohoo.ncsa.uiuc.edu/cgi/
•"FastCGI Home", http://www.fastcgi.com
•"Java EE At a Glance", http://java.sun.com/javaee
•"Java EE 5 SDK", http://java.sun.com/javaee/5/docs/api/