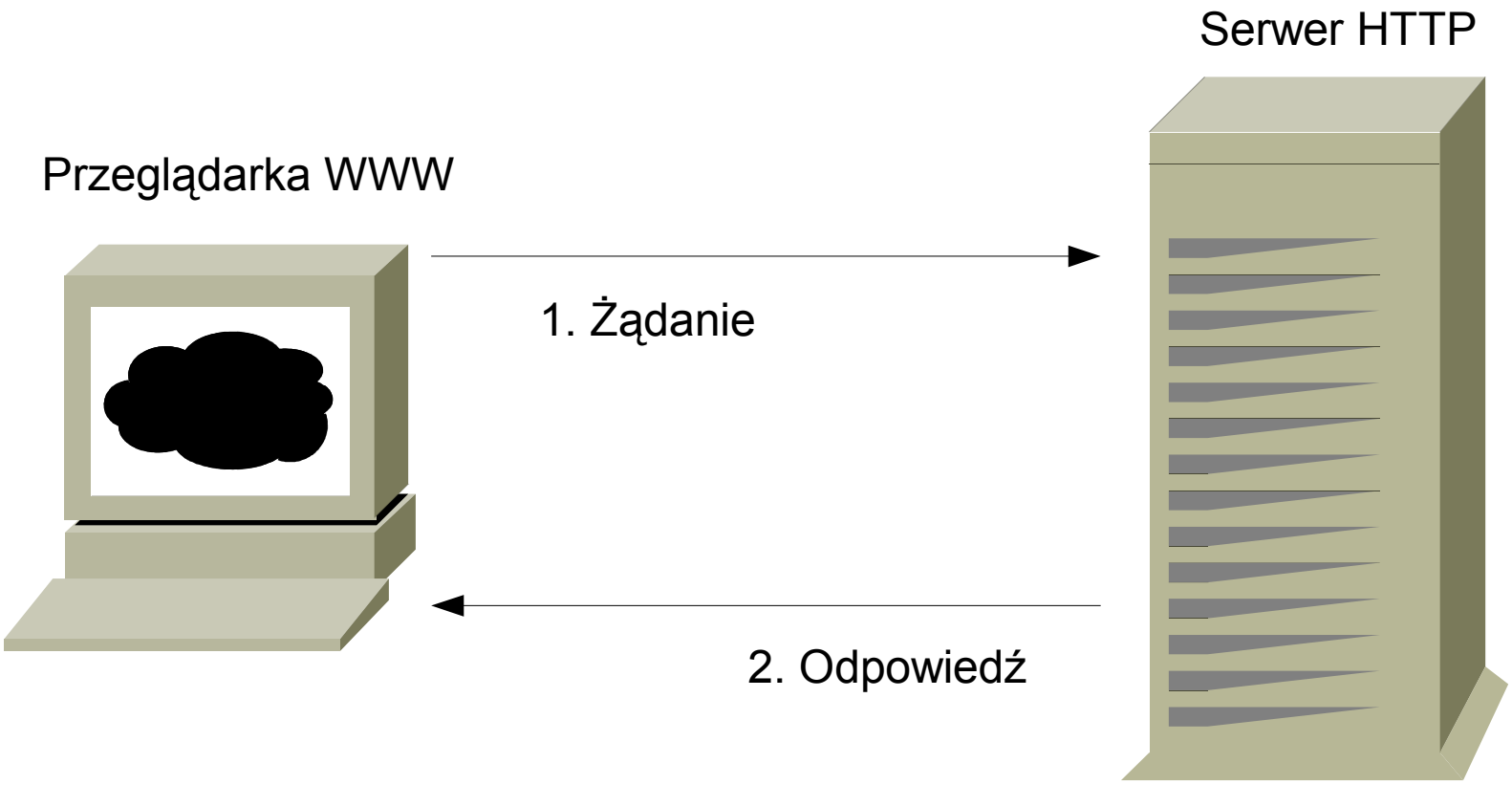
Piotr Laskowski
Krzysztof Stefański

Java™ Servlets

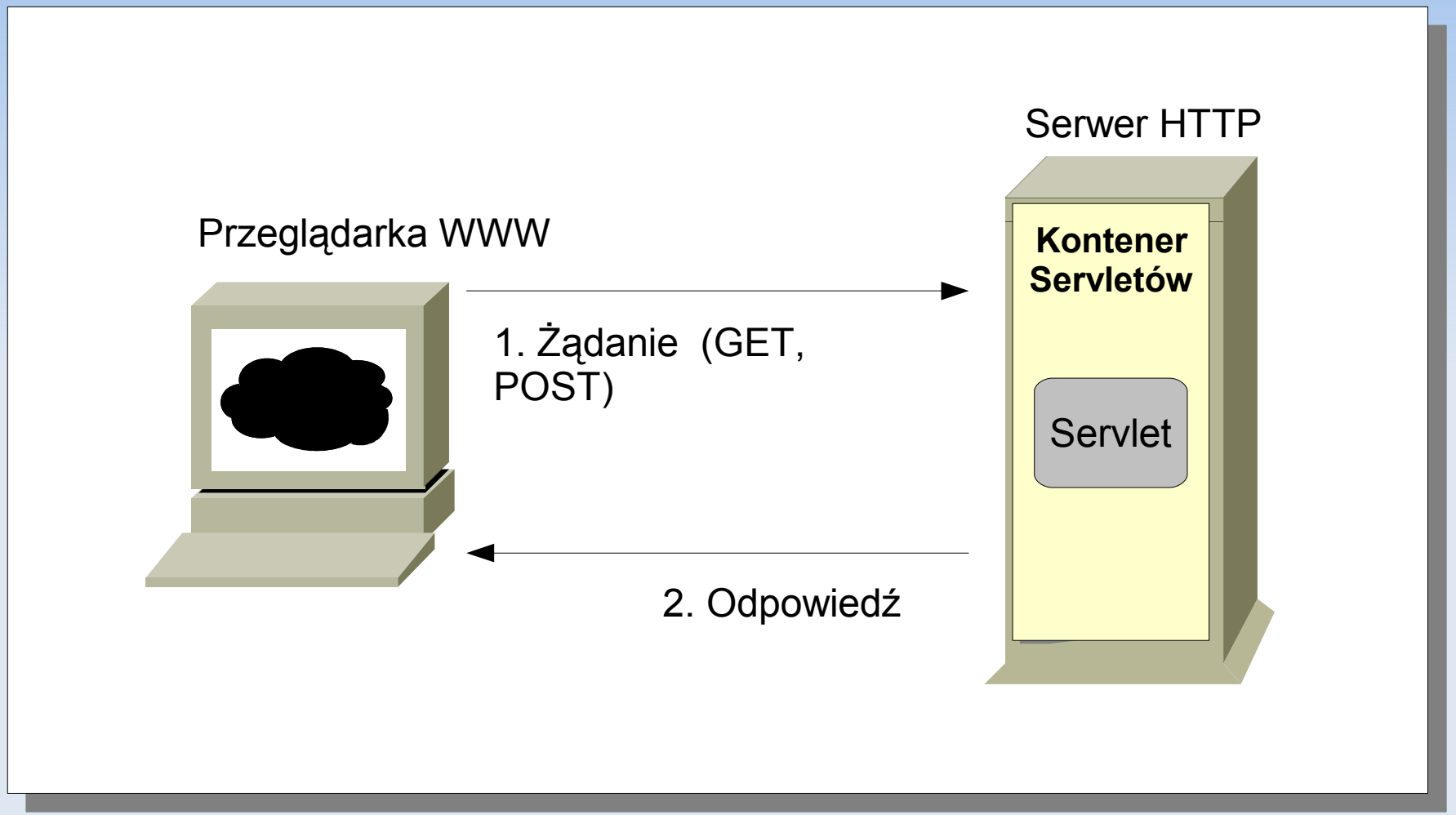
Java Servlets

- Technologia dynamicznego generowania treści dla aplikacji WWW
- Wyspecyfikowana przez Sun, obecnie przez Java Community Process
- Pierwsza formalna specyfikacja 2.1: 1998 r.
- Aktualna wersja 2.5 (w planach 3.0)

Model Request-Response



Kontener servletów



- Kontener odbiera żądanie i zleca jego obsługę wybranemu servletowi

Ontologia servletu

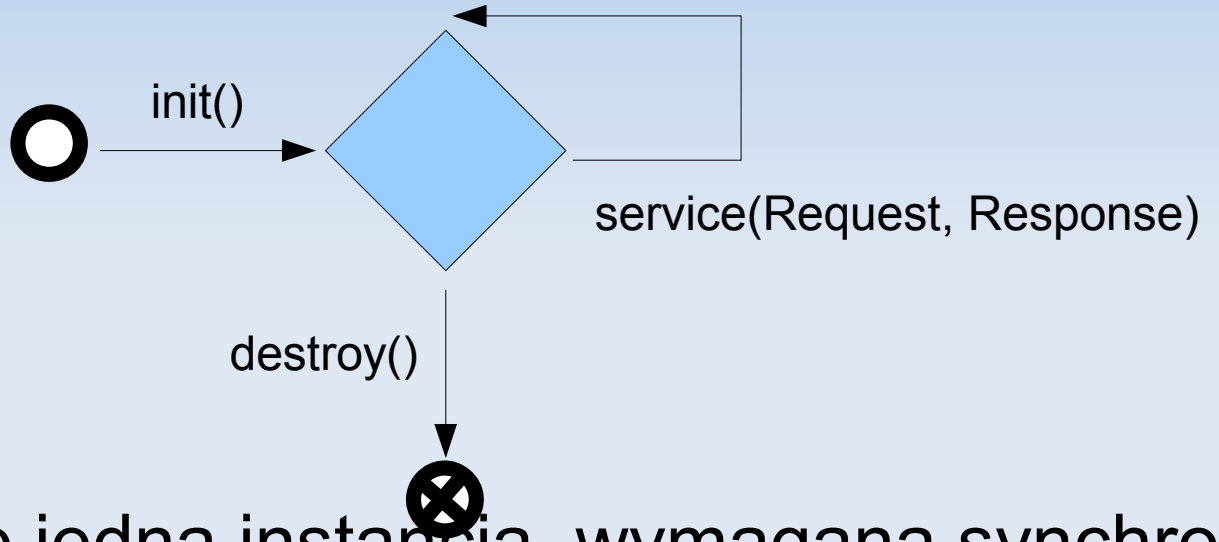
- Pakiet javax.servlet i javax.servlet.http
- Klasa GenericServlet, HTTPServlet

```
public abstract class HTTPServlet extends GenericServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
        response) throws IOException, ServletException
    {
        PrintWriter out = response.getWriter();
        ...
        out.println(”Witaj gościu na mojej stronie!”);
    }
    public void doPost(HttpServletRequest request, HttpServletResponse
        response) throws IOException, ServletException;
    ...
}
```

- Interfejsy HttpServletRequest, HttpServletResponse

Cykl życia servletu

- Realizowany przez zestaw metod **GenericServlet**



- Zwykle jedna instancja, wymagana synchronizacja
- Interfejs SingleThreadServlet (Deprecated)

Pobieranie danych z żądania

- Metody HTTPServletRequest:
  - getParameter(String name)
  - getParameterMap, getParameterValues itp.
- Jednolity interfejs dla danych GET i POST
- Bezpośrednio odczytując treść żądania (metoda getInputStream)
  - Przydatne przy wgrywaniu plików

Obsługa nagłówków żądania

- Wygodne API udostępnione przez obiekt żądania
  - getHeader(String name)
  - Skróty dla najpopularniejszych: np. getMethod(), getRequestURI()
- Możliwa również drobiazgowa obsługa (getHeaderNames, getHeaders)
  - Przykładowo: obsługa Accept-Language
- Można też ręcznie

Nagłówki odpowiedzi

- Obsługiwane przez API obiektu **HTTPServletResponse**
- setStatus(int statusCode); stałe SC_*
- Skróty: sendError, sendRedirect(String url)
- setHeader(String name, String value)
- Skróty: setContentType(String value)

Wdrażanie aplikacji - web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <context-param>
      <param-name>name_of_context_initialization_parameter</param-name>
      <param-value>value_of_context_initializtion_parameter</param-value>
    </context-param>
    <servlet>
      <servlet-name>welcomeServlet</servlet-name>
      <servlet-class>com.foo-bar.somepackage.TheServlet</servlet-class>
      <init-param>
        <param-name>foo</param-name>
        <param-value>bar</param-value>
      </init-param>
    </servlet>
    <servlet-mapping>
      <servlet-name>welcomeServlet</servlet-name>
      <url-pattern>/welcome</url-pattern> <!-- relatywne wzgledem URI wdrż. -->
    </servlet-mapping>
</web-app>
```

Utrzymywanie sesji

- Dane sesji przechowywane są po stronie serwera
- Identyfikatory sesji przechowywane są w ciasteczkach po stronie klienta
- Gdy klient ma wyłączoną obsługę ciasteczek, identyfikator sesji przekazujemy za pomocą mechanizmu GET.
- Czas, po jakim sesja wygasa definiuje się w pliku web.xml

Dostęp do sesji

- Sesja jest reprezentowana przez obiekt klasy HttpSession
- Obiekt otrzymujemy wywołując metodę getSession() na obiekcie request
- Atrybuty zapisujemy metodą

  setAttribute(String, Object), i pobieramy metodą getAttribute(String)
- invalidate – jawne zniszczenie sesji

Sesja - przykład

```
public class Shopping extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
              response) throws IOException, ServletException
    {
        response.setContentType(”text/html”);
        PrintWriter out = response.getWriter();

        HttpSession session = request.getSession();

        Cart cart;
        synchronized(session) {
            cart = (Cart) session.getAttribute(”cart”);
            if (cart == null) {
                cart = new Cart();
                session.setAttribute(”cart”, cart);
            }
        }
        out.println(”Masz ” + cart.getAmount() + ” rzeczy w koszyku.”);
        out.println(”<p><a href=\” +
               response.encodeURL(request.getContextPath() + "/payment")
               + ”\”>Kup</a></p>”);
    }
}
```

Sesja – funkcjonalności

- Z obiektu sesji można otrzymać różne informacje, które jej dotyczą
- np. getAttributeNames(), getLastAccessedTime(), ...
- Więcej: http://java.sun.com/webservices/docs/1.6/api/javax/servlet/http/HttpSession.html

Cookies

- Oprócz identyfikatora sesji, możemy po stronie klienta trzymać inne informacje (tekstowe)
- Do ciasteczek dostęp mamy poprzez klasę Cookie
- HttpServletRequest.getCookies() zwraca tablicę obiektów klasy cookie
- HttpServletResponse.addCookie(Cookie) dodaje ciasteczko

Cookies - przykład

```
public class Shopping extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
           response) throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        Cookie[] cookies = request.getCookies();
        Integer amount = null;
        for (Cookie c : cookies) {
            if (c.getName().equals(”amount”))
                amount = Integer.parseInt(c.getValue());
        }

       if (amount == null) {
            amount = 0;
            Cookie c = new Cookie(”amount”, amount.toString());
            response.addCookie(c);
        }

         out.println(”Masz ” + amount.toString() + ” rzeczy w koszyku.”);
    }
}
```

ServletContext

- Czasem chcemy przechowywać dane nie związane z konkretną sesją, lecz wspólne
- Można w tym celu wykorzystać obiekt klasy ServletContext, który jest związany z daną aplikacją
- Dostęp do niego jest zapewniony poprzez metodę getServletContext() klasy GenericServlet
- Trzeba pamiętać o synchronizacji!

ServletContext - przykład

```
public class Counter extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
           response) throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        ServletContext context = this.getServletContext();
        Counter counter = (Counter)context.getAttribute("hitCounter");
        out.println(”Jesteś ” + counter.getAndInc() +
             ” osobą na tej stronie”);
    }
}
```

Przekierowanie

- Jeśli chcemy przekierować zapytanie na zupełnie inną stronę, można użyć metody ServletResponse.sendRedirect(String URL), która wyśle do przeglądarki prośbę o przekierowanie
- Jeśli robimy przekierowanie wewnątrz własnego serwisu, zwykle lepiej jest użyć klasy RequestDispatcher, która przekazuje prośbę wygenerowania strony dalej

RequestDispatcher - przykład

```
…

String page = request.getParameter(”page”);
RequestDispatcher dispatcher;

if (page.equals(”home”))
    dispatcher = this.getServletContext().getRequestDispatcher("/home");
else dispatcher =
    this.getServletContext().getRequestDispatcher("/underconstruction");

if (dispatcher != null) {
    dispatcher.forward(request, response);
}

…
```

Przekazywanie informacji

- Czasem robiąc przekierowanie do innej strony za pomocą forward() chcielibyśmy przekazać jej jakieś dodatkowe parametry
- W obiekcie ServletRequest możemy przechowywać obiekty-atrybuty w taki sam sposób, jak w sesji bądź w kontekście aplikacji.

Filtry

- Jeśli chcemy doczepić coś do strony generowanej przez servlet (lub w jakiś inny sposób ją zmodyfikować), możemy użyć do tego mechanizmu filtrów
- Możemy również chcieć ”oszukać” servlet, podmieniając mu dane żądania

Filtry

- Dla danego servletu możemy w pliku web.xml zdefiniować łańcuch filtrów, które będą po kolei uruchamiane. Ostatni w łańcuchu jest servlet.
- Czynności wykonywane przez filtr:
  - Otrzymuje obiekt response i request i wstępnie je przetwarza (lub podmienia)
  - Wywołuje metodę doFilter(request, response), uruchamiając swojego następnika w łańcuchu
  - Robi końcowe przetwarzanie (np. dodaje baner reklamowy)

Część praktyczna

- http://students.mimuw.edu.pl/~ks250429/servlety.tar.gz
- http://students.mimuw.edu.pl/~ks250429/tomcat.tar.gz
- Uruchomienie kontenera Apache Tomcat.
- Uruchomienie servletu wypisującego datę i godzinę. Dodanie informacji o adresie IP klienta (HttpServletRequest.getRemoteAddr()).
- Uruchomienie servletu przygotowującego listę produktów. Dodanie obsługi sytuacji, kiedy klient ma wyłączoną obsługę ciasteczek.

Pytania

Czy są jakieś pytania?

Bibliografia

- http://java.sun.com/javaee/5/docs/tutorial/doc/
- http://java.sun.com/javaee/5/docs/api/

Koniec

Dziękujemy za uwagę