

Serwlety na platformie Java EE

Co to jest serwlet?

- kawałek kodu w Javie po stronie serwera HTTP
- rozszerza możliwości serwera
- CGI, w Javie, wzbogacone o biblioteki ułatwiające życie programistycie (np. utrzymywanie sesji, wpólne zasoby serwletów, ciasteczka, obsługa GET/POST/PUT)
- serwlety mogą korzystać ze standardowych klas Javy (z VM), klas wchodzących w skład Servlet API: javax.servlet.* i javax.servlet.http.* oraz ewentualnie z dodatkowych bibliotek zainstalowanych na serwerze.
- serwlety nie mają interfejsu użytkownika, komunikują się z przeglądarką (za pośrednictwem serwera WWW wspierającego serwlety) za pomocą protokołu HTTP

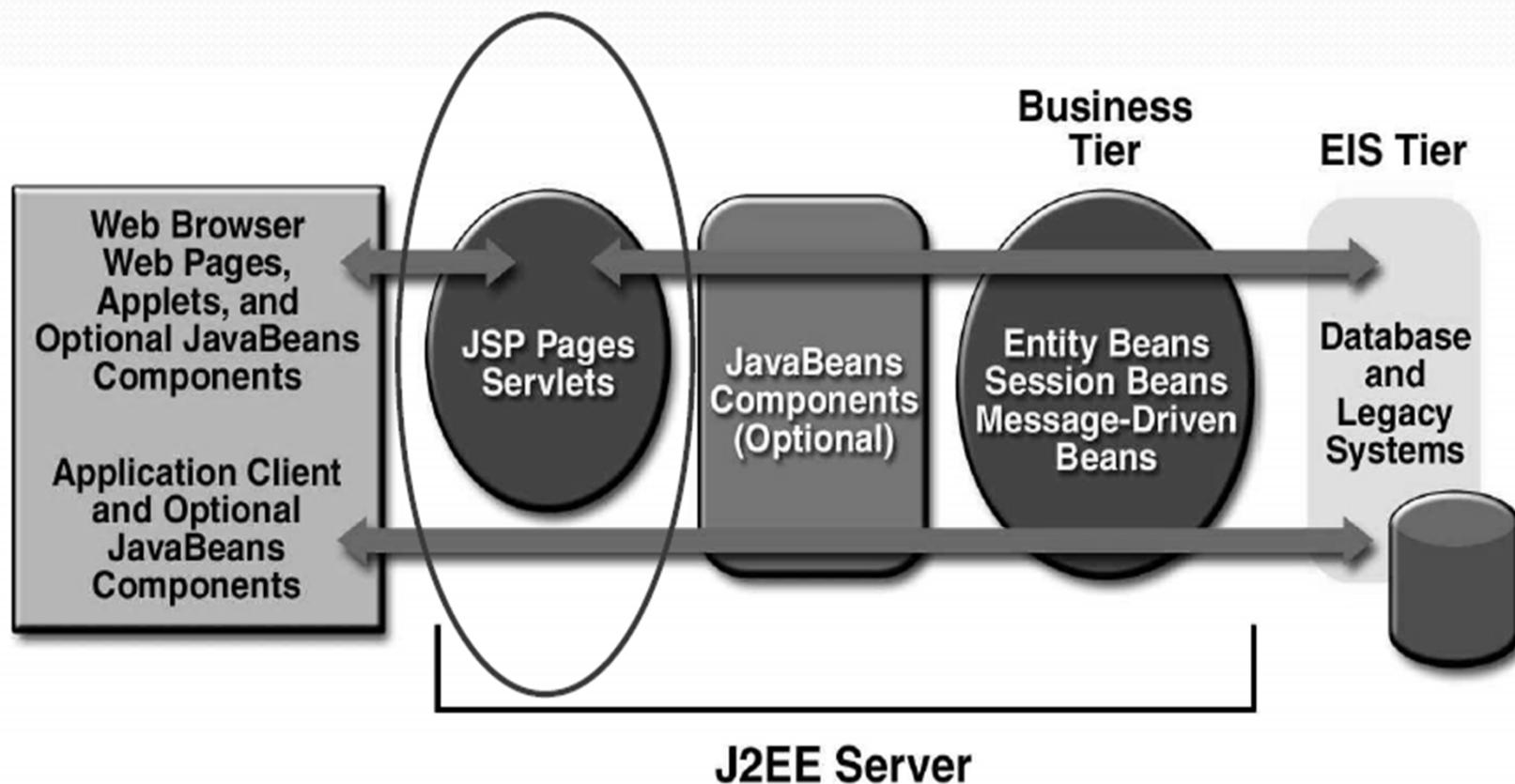
Do czego służy serwlet?

- Do wyświetlania dynamicznych stron WWW (alternatywa do popularnego PHP)
- przetwarzanie formularzy
- forwarding
- wyświetlania wyników zapytań do b. d. (tu zaleca się strony JSP)

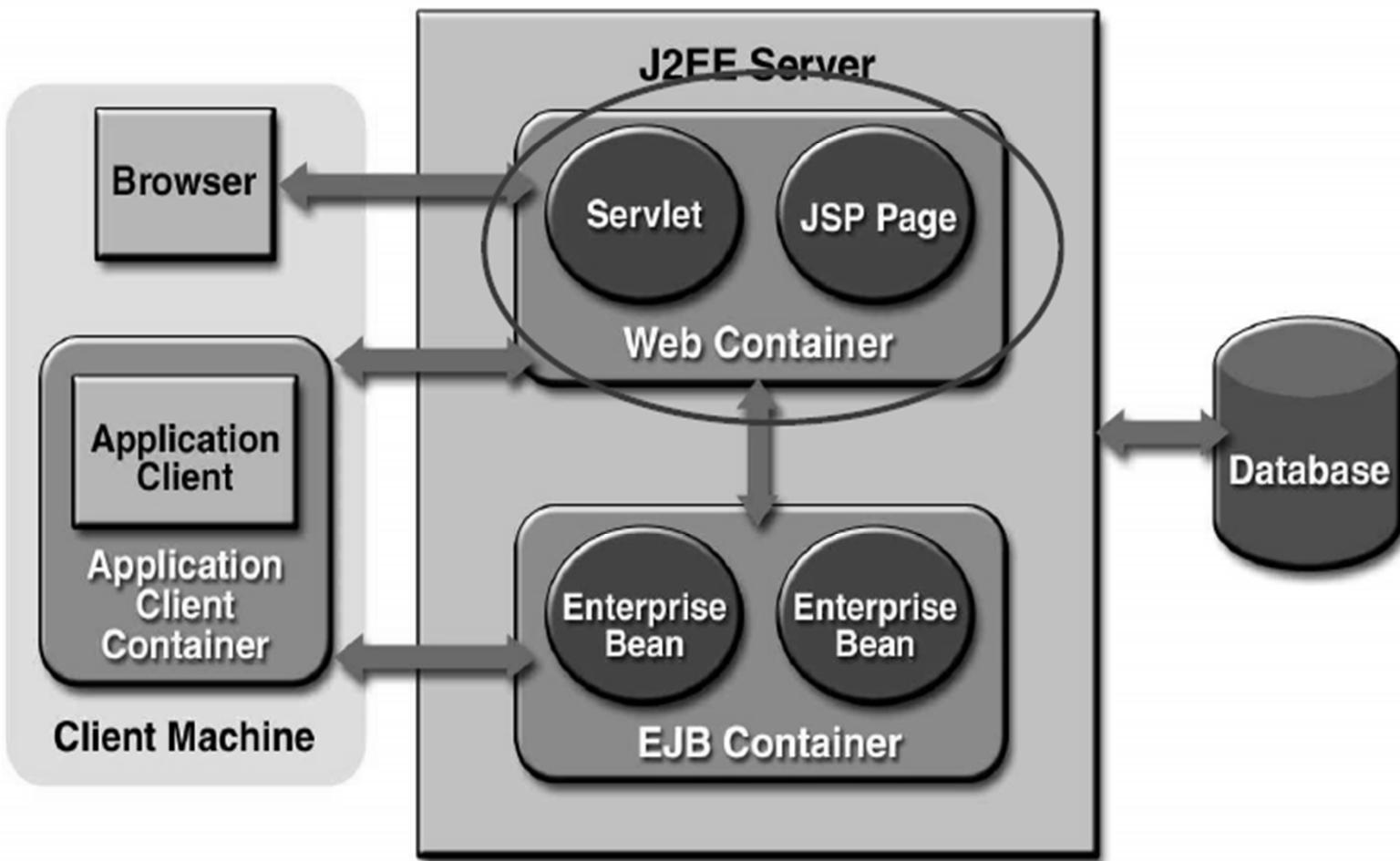
Miejsce serwletów w aplikacjach J2EE

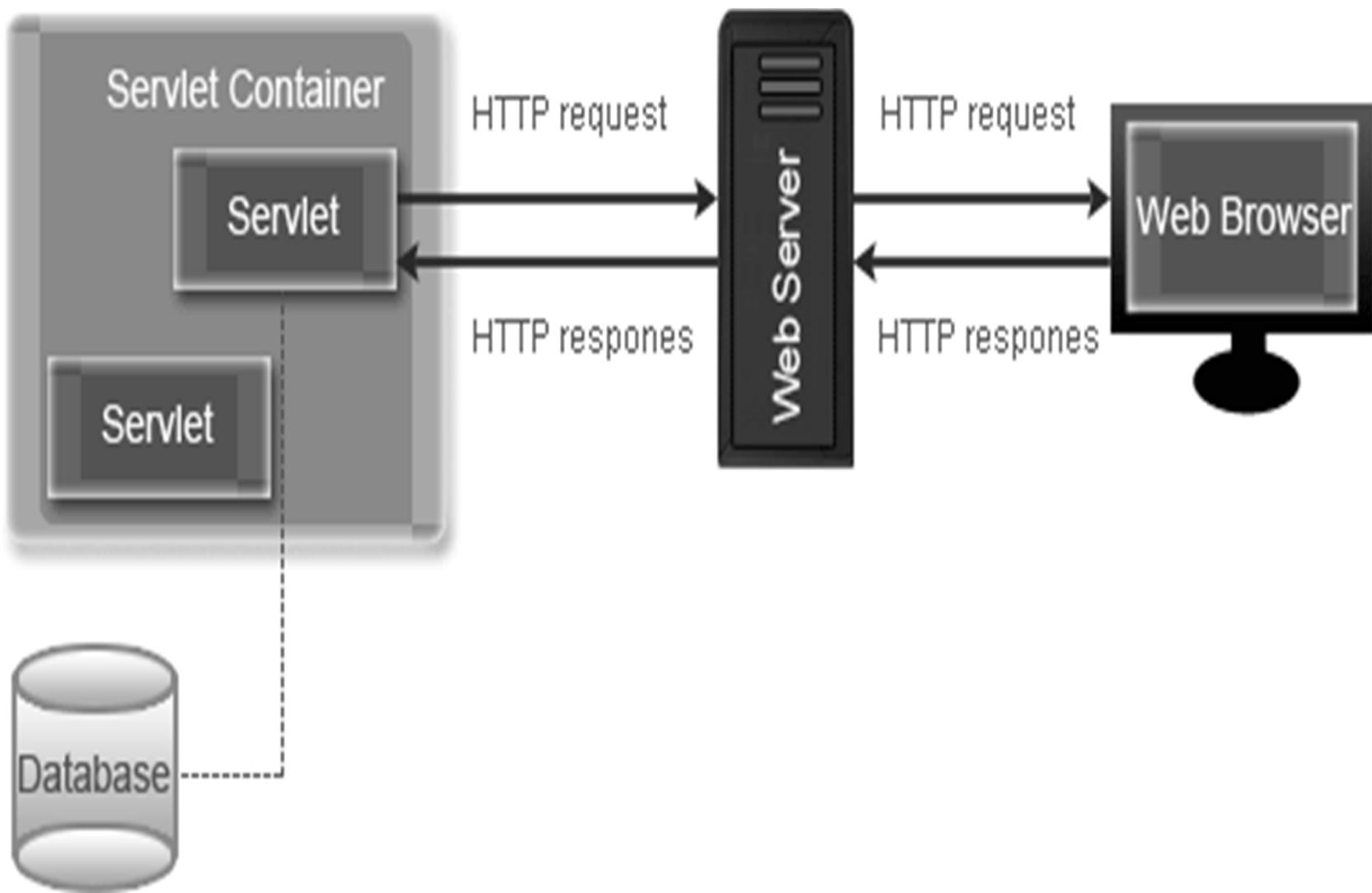
- serwlety to rozszerzenie serwera WWW
- blisko interfejsu użytkownika: zorientowane na żądanie/odpowiedź (request/response)
- serwlety działają w odpowiedzi na żadania klienta
- są interfejsem do serwera aplikacyjnego/bazy danych/EJB
- pełnią rolę kontrolera w schemacie Model-View-Controller, za widok odpowiedzialne są strony JSP, model to EJB
- w serwletach zaleca się umieszczanie tylko lekkich funkcji biznesowych, ale w małych aplikacjach cała logika może być w serwletach i JSP

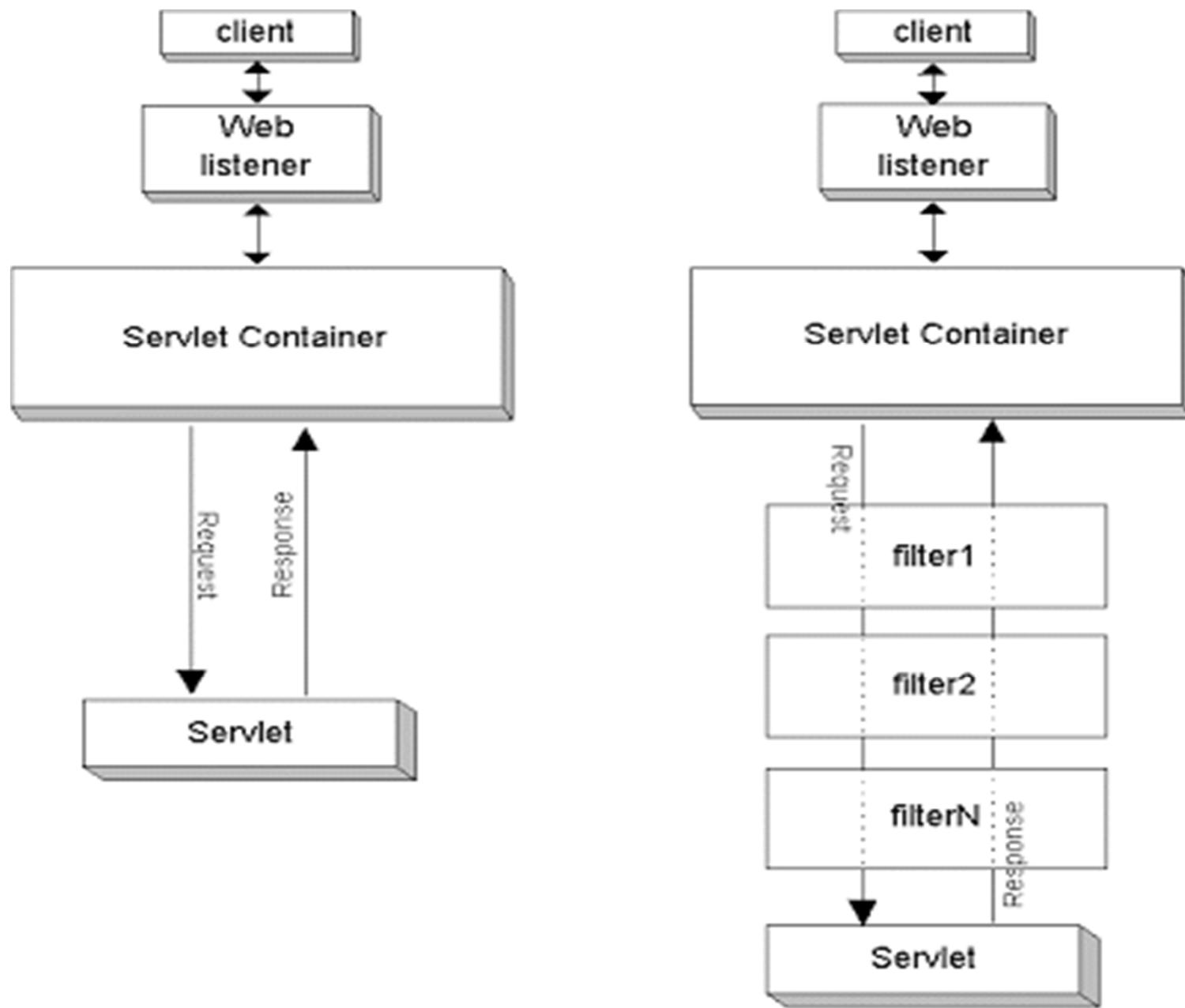
Model J2EE



Kontenery w modelu J2EE







Komunikacja z serwerem

- przeddefiniowanie metod doGet, doPost, doPut, doDelete. Te metody są wywoływanie przez metodę service zdefiniowaną w Servlet przy każdym żądaniu do serwera dotyczącym serwletu.
- dane od klienta (przeglądarki) są w obiekcie HttpServletRequest.
- dane do klienta symbolizuje obiekt HttpServletResponse

Ważne metody obiektu request:

- *getParameter* - zwraca parametr, np. z formularza
- *getCookies* - daje ciasteczka
- *getSession* - podaje obiekt sesji
- *getReader* (dla danych tekstowych)
- *getInputStream* (dla danych binarnych)

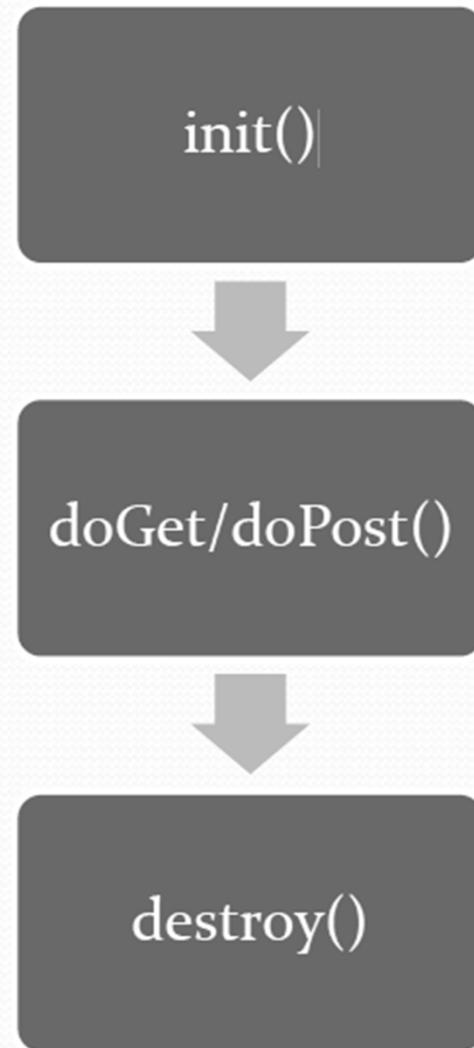
Ważne metody obiektu response:

- *getWriter* (dane tekstowe)
- *getOutputStream* (dane binarne)

Serwlet Java

- Serwlet musi:
 - dziedziczyć po klasie HttpServlet
 - obsługiwać jedną z metod:
 - doGet(HttpServletRequest request, HttpServletResponse response)
 - doPost(HttpServletRequest request, HttpServletResponse response)
- opcjonalnie (wywoływane tylko raz!):
 - *public void init(ServletConfig config)*
 - *public void destroy()*

Cykl życia serwletu



Cykl życia serwletu

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.

Cykl życia serwletu

- na początku system woła metodę init(), w niej można np. połączyć się z bazą danych
- jeżeli coś pójdzie nie tak to należy rzucić wyjątek UnavailableException
- można wołać metodę getInitParameter, interpretowaną przez serwer (kontener), np. jako init-param z pliku web.xml w Tomcat'cie. W takim parametrze może być zapisany np. URL do bazy danych.
- gdy serwer zdecyduje wyrzucić serwlet z pamięci to wywołuje metodę destroy. Uwaga! Jeżeli metoda do[Get|Post] wykonuje się długo, to trzeba zadbać o jej zakończenie. Jeżeli serwlet uruchamiał wątki, to też trzeba je przerwać.

Wzmianka o wielowątkowości

- Serwlety wykonują się w środowisku wielowątkowym, więc może się zdarzyć, że wiele kopii serwletu będzie działać naraz. A może być tak, że będzie jedna instancja serwletu, ale wiele wątków wykonujących metodę service. Jeżeli w serwacie jest odwołanie do zasobu, który wymaga wyłączności to:
- trzeba zapewnić wykluczanie ręcznie, np. używając zmiennych/metod synchronized
- zadeklarować, że serwlet implementuje interfejs SingleThreadModel, wtedy serwer uruchomi jednocześnie tylko jedną instancję metody service, np. public class ReceiptServlet extends HttpServlet implements SingleThreadModel. Ten interfejs nie zawiera żadnych metod, jest tylko znacznikiem dla kontenera.

Utrzymywanie stanu (session tracking)

Cechy:

- Obiekty nie są nigdzie składowane, tylko istnieją w pamięci kontenera. Metody operują na referencjach. Po odczytaniu referencji metodą `getAttribute` operujemy już na oryginalnym (jedynym!) obiekcie.
- Po odczytaniu atrybutu trzeba sprawdzić czy jest równy null, jeżeli tak to trzeba utworzyć nowy obiekt (uwaga na wielowątkowość).
- Sesje działają standardowo na ciasteczkach. W ciasteczku jest tylko id sesji, reszta siedzi w pamięci kontenera. Jeżeli przeglądarka nie obsługuje ciasteczek to trzeba używać metody `encodeURL()`.
- Sesje nie są związane z jednym serwletem, różne serwlety korzystają z tej samej sesji

Sesje

- Problem: protokół HTTP jest bezstanowy
- Emulacja sesji
- Każda sesja otrzymuje identyfikator
- Stan sesji przechowywany przez serwer aplikacji
- Stan sesji usuwany po wygaśnięciu czasu ważności

Sesje przykład

- *HttpSession session = request.getSession(true);*
(czytanie/pisanie)
- *Klasa ref_obiekту1 = new Klasa(); (pisanie)*
- *session.putAttribute("dluga.nazwa_atr", ref_obiekту1);*
(pisanie)
- *Klasa ref_obiekту2 = (Klasa)*
session.getAttribute("dluga.nazwa_atr"); (czytanie)

Ciasteczka

- Zmienne Cookies reprezentowane w postaci obiektów klasy Cookie
- Wysyłanie zmiennych Cookies: response.addCookie()
- Odczyt zmiennych Cookies: request.getCookies()

Ciasteczka

- Wysyłanie

```
String wartosc = „wartosc”;  
Cookie ciacho = new Cookie("Nazwa", wartosc);  
response.addCookie(ciacho);
```

- Odbieranie

```
Cookie[] cookies= request.getCookies();  
String wartosc;  
for (int i = 0; i < cookies.length; i++) {  
    if(cookies[i].getName().equals("Nazwa"))  
        wartosc = thisCookie.getValue();  
}
```

HTTPSession czy Cookies?

- Oba mechanizmy pozwalają przechowywać stan sesji
- Cookies wymaga przesyłania obiektów stanu w nagłówkach HTTP
- HTTPSession obsługuje dowolne typy obiektów stanu
- Bezpieczeństwo