Servlety

Jakub Szlenk

Struktura aplikacji webowej

- / – katalog zawierający wszelkie widoki, szablony, a także wszystkie pliki, które mają być dostępne publicznie. Serwer nie udostępnia jedynie plików zawartych w podkatalogach WEB-INF i META-INF

- /WEB-INF – katalog zawierający dodatkowe zasoby niezbędne do prawidłowego działania aplikacji, m.in. skompilowane klasy Java i biblioteki (JAR)

- /WEB-INF/web.xml – deskryptor aplikacji webowej – plik zawierający wszystkie istotne ustawienia, mający kluczowe znaczenie dla działania całej aplikacji

Struktura aplikacji webowej

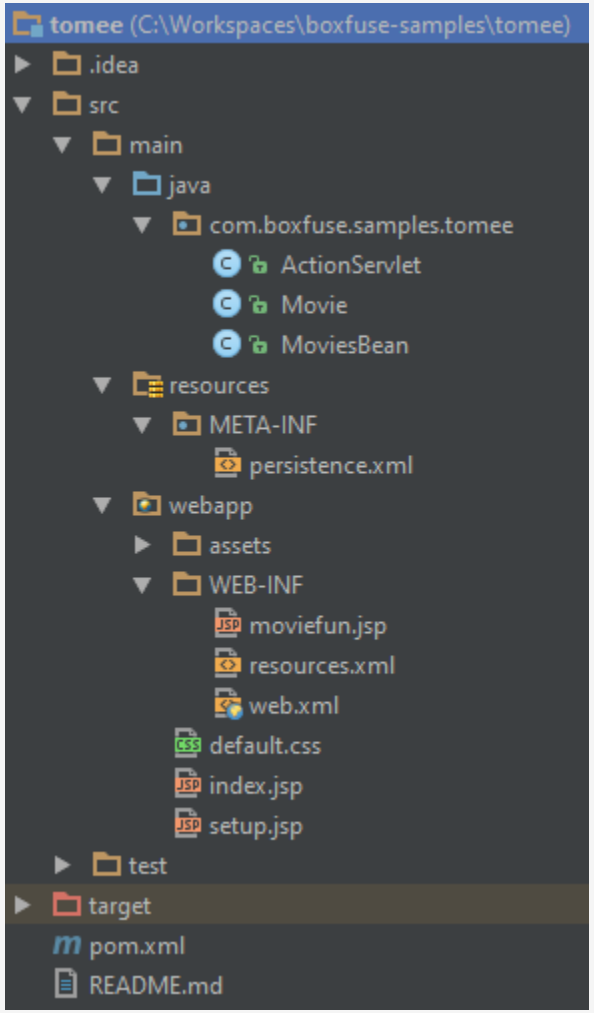
- /WEB-INF/classes – katalog zawierający skompilowane klasy
- /WEB-INF/lib – katalog zawierający dodatkowe biblioteki

- /META-INF – zawiera plik manifest.mf – jego obecność wynika z faktu, że każda aplikacja webowa przygotowana do wdrożenia ma postać pojedynczego archiwum WAR (Web ARchive), będącego specyficznym przypadkiem zwykłego pliku JAR

- Opcjonalnie: odwołania do usług sieciowych, plik persistence.xml (plik konfiguracyjny stosowany w technologii JPA)

Struktura aplikacji webowej

Hierarchia folderów



Plik konfiguracyjny web.xml

- web.xml jest to plik konfiguracyjny aplikacji internetowych w Javie. Nakazuje kontenerowi serwletów, które klasy załadować, jakie parametry ustawić w kontekście i jak przechwytywać żądania pochodzące z przeglądarek.

- Podstawowe zastosowanie pliku web.xml to powiązanie wzorców adresów URL z konkretnymi klasami serwletów.
- Powiązanie odbywa się dwuetapowo:

1. nazwa klasy serwletu <-> nazwa serwletu
2. nazwa serwletu <-> jeden lub wiele wzorców UR

Struktura aplikacji webowej

Pierwszy etap: przydzielenie klasie serwletu przyjaznej nazwy:

```
<servlet>
  <servlet-name>AddCookieServlet</servlet-name>
  <servlet-class>AddCookieServlet</servlet-class>
</servlet>
```

- Nazwa może zawierać spacje
- W pozostałej treści pliku web.xml należy odwoływać się do serwletu tylko za pomocą nazwy

Struktura aplikacji webowej

Drugi etap: przypisanie serwletowi przynajmniej jednego wzorca URL:

```
<servlet-mapping>
  <servlet-name>AddCookieServlet</servlet-name>
  <url-pattern>/servlets/servlet/AddCookieServlet</url-pattern><servlet-mapping>
</servlet-mapping>
```

- W tym momencie użytkownik po wejściu na stronę http://localhost/servlets/serwlt/AddCookieServlet wywoła AddCookieServlet

Struktura aplikacji webowej

Jako wartość dla znacznika url-pattern można podać następujące rodzaje wzorców:

- wzorzec dokładny – np. /serwlety/Serwlet
- wzorzec wieloznaczny – np. /serwlety/* – zostaną dopasowane wszystkie żądania, które zawierają fragment /serwlety/
- wzorzec rozszerzeń – *.xhtml – zostaną dopasowane wszystkie żądania, które odwołują się do zasobów o danym rozszerzeniu

Serwlet może otrzymać parametry inicjalizacyjne:

- Są one umieszczane w znaczniku <servlet>
- Dzięki nim można wpływad na działanie aplikacji bez rekompilacji plików źródłowych

Struktura aplikacji webowej

Przykład:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
      http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
      version="3.1">
  <servlet>
    <servlet-name>AddCookieServlet</servlet-name>
    <servlet-class>AddCookieServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>GetCookiesServlet</servlet-name>
    <servlet-class>GetCookiesServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>DataServlet</servlet-name>
    <servlet-class>DataServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>AddCookieServlet</servlet-name>
    <url-pattern>/servlets/servlet/AddCookieServlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>GetCookiesServlet</servlet-name>
    <url-pattern>/servlets/servlet/GetCookiesServlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>DataServlet</servlet-name>
    <url-pattern>/servlets/servlet/DataServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Co to jest Servlet ?

Rys historyczny

Kiedyś serwer mógł dynamicznie generować stronę, tworząc osobne procesy do obsługi wszystkich żądań ze strony klientów. Procesy te otwierały połączenia z bazą danych, aby uzyskać potrzebne informacje, oraz komunikowały się z serwerem WWW za pośrednictwem interfejsu znanego jako CGI (*Common Gateway Interface*). Pozwalał on na odczytywanie danych z żądania w formacie HTTP i generowanie odpowiedzi. Do pisania programów obsługujących interfejs CGI wykorzystywane było różne języki programowania, m.in. C, C++ i Perl.

Jednak technologia CGI powodowała duże problemy z wydajnością. Wymagała ona dużej mocy obliczeniowej i pamięci pozwalającej na tworzenie osobnych procesów dla każdego żądania. Przez to była bardzo droga. Poza tym programy obsługujące interfejs CGI nie mogły być uruchamiane na innych platformach. Z tych powodów stworzone zostały inne techniki, m.in. **serwlety**.

Dlaczego servlety okazały się lepszym rozwiązaniem ?

Serwlety mają wiele zalet w porównaniu z interfejsem CGI.

- Po pierwsze, ich wydajność jest znacznie lepsza. Są one uruchamiane w obrębie przestrzeni adresowej serwera WWW. Nie ma konieczności tworzenia osobnych procesów dla każdego żądania.
- Po drugie, ponieważ są one napisane w języku Java, istnieje możliwość uruchamiania ich na różnych platformach.
- Po trzecie, menedżer bezpieczeństwa Javy wprowadza ograniczenia, które pozwalają na zabezpieczenie zasobów umieszczonych na serwerze.
- W końcu serwlet ma możliwość wykorzystania wszystkich funkcji bibliotek języka Java. Może on komunikować się innym oprogramowaniem za pośrednictwem gniazd i mechanizmów RMI, które już poznałeś.

Autor:

Servlety | tworzenie, konfiguracja, wdrożenie

- Serwlet jest klasą, która dziedziczy po klasie javax.servlet.GenericServlet
- Zadaniem serwletu jest obsługa żądań (request) i generowanie odpowiedzi (response)
- Serwlet nie jest samodzielną aplikacją – jest on uruchamiany przez kontener webowy
- Mimo istnienia uniwersalnej architektury, znaczna większość serwletów dziedziczy po klasie HttpServlet, przystosowanej rzecz jasna do obsługi żądań i odpowiedzi HTTP
- Po utworzeniu w odpowiedni sposób klasy serwletu i dodaniu niezbędnego kodu XML do pliku web.xml (lub adnotacji), serwlet jest gotowy do obsługi żądań (oczywiście po uruchomieniu serwera)

Servlety | tworzenie, konfiguracja, wdrożenie

Klasa serwletu, dziedzicząc po klasie HttpServlet, musi zawierać definicję co najmniej jednej z metod obsługujących żądania HTTP:

doGet(), doPost(), doDelete(), doPut()

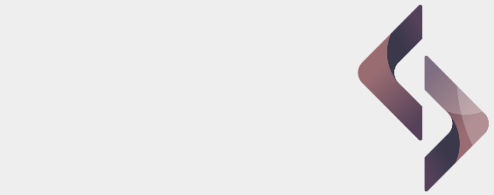
Każda z tych metod obsługuje żądania o określonej metodzie HTTP (GET/POST/PUT/DELETE)

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, java.io.IOException

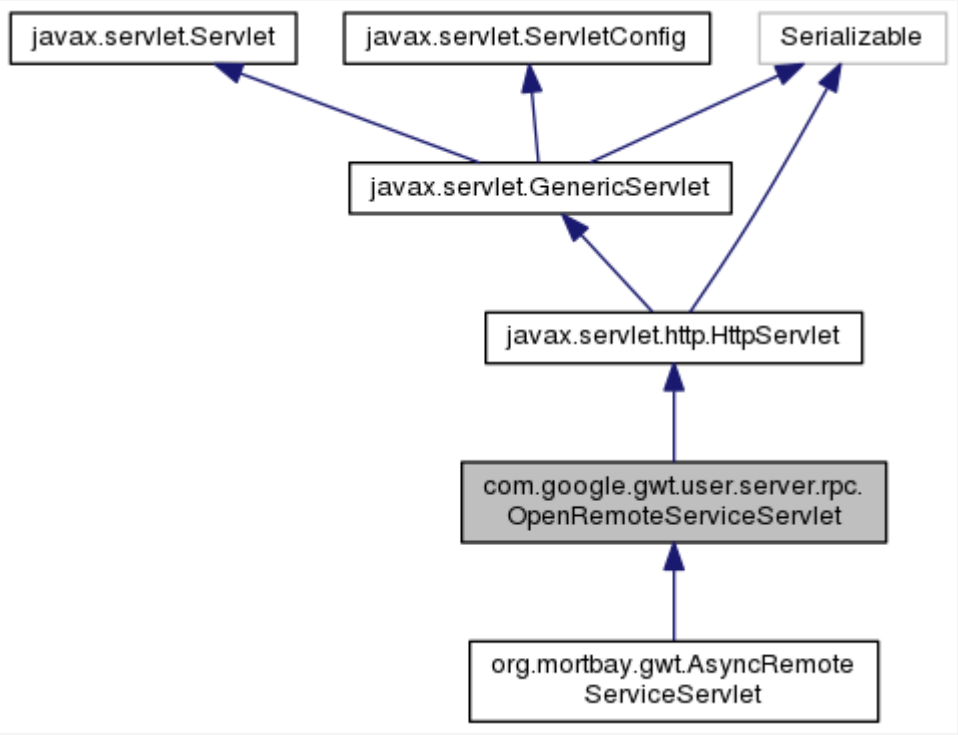
Wszystkie te metody mają identyczne sygnatury (poza nazwami).

Parametr req reprezentuje obiekt żądania, podczas gdy resp – obiekt odpowiedzi

Servlety | tworzenie, konfiguracja, wdrożenie



- Hierarchia klas:



Servlety | tworzenie, konfiguracja, wdrożenie

Działanie serwletu w dużej mierze można uprościć do schematu:

- pobierz niezbędne informacje z żądania
- wykonaj logikę biznesową
- wygeneruj odpowiedź, korzystając ze strumieni dostępnych w obiekcie odpowiedzi

Servlety | tworzenie, konfiguracja, wdrożenie

Przykład:

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

public class HelloServlet extends GenericServlet {

  public void service(ServletRequest request, ServletResponse response) throws ServletException,
IOException {
    response.setContentType("text/html");
    response.setCharacterEncoding("UTF-8");
    PrintWriter pw = response.getWriter();
    pw.println("<B>Witaj SDA!");
    pw.close();
  }
}
```

Servlety | tworzenie, konfiguracja, wdrożenie

- Przy użyciu obiektu klasy PrintWriter można wygenerować dane tekstowe, które trafią do przeglądarki internetowej klienta

- W związku z tym w wywołaniach metod print/println można przekazywać kod HTML/CSS lub w dowolnym innym pożądanym języku obsługiwanym przez przeglądarkę (np. SVG)

- Do określenia typu przesyłanych danych należy stosować metodę setContentType odpowiedzi

- Jeśli zamiast obiektu klasy PrintWriter zostanie wykorzystany obiekt klasy ServletOutputStream, pobrany za pomocą metody getOutputStream odpowiedzi, można przesyłać dane binarne

Servlety | tworzenie, konfiguracja, wdrożenie

Przykład:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
   byte[] bufor = ... // pobranie pliku z dysku;
   response.setContentType("image/png");
   response.addHeader("Content-Disposition", "attachment;  filename=obrazek.png");
   response.setContentLength(bufor.length);
   OutputStream strumien = response.getOutputStream();
   strumien.write(b);
   strumien.flush();
}
```

Servlety | tworzenie, konfiguracja, wdrożenie

- Istnieje możliwość dołączenia do odpowiedzi nagłówków i ciasteczek:

void addHeader(String nazwa, String wartość)

void addCookie(Cookie ciastko)

- Aby pobrać parametry przesłane w żądaniu (czy to za pomocą metody GET, czy też POST), należy skorzystać z metody typu HttpServletRequest:

- String getParameter(String nazwa)

Servlety | definicja

Servlet to klasa, których celem jest przetwarzanie żądań HTTP i generowanie zawartości która będzie odesłana w odpowiedzi na te żądania. Innymi słowy, servlety służą do implementacji dynamicznych aplikacji WWW.

- Technologia Java Servlets (obok JSP) jest integralną częścią Java EE
- Umożliwia przetwarzanie żądań HTTP i generowanie odpowiedzi
- Servlety to klasy Javy implementujące interfejs HttpServlet (Dzięki nim (m.in.) separuje się warstwę logiki biznesowej od warstwy prezentacji)
- Do uruchomienia servlety wymagają serwera webowego, np. Tomcata
- Konfigurację servletu można zawrzeć w adnotacji @WebServlet lub w pliku web.xml
- Na serwerze występuje zazwyczaj jedna instancja servletu, z której korzysta wiele wątków jednocześnie

Servlety | cykl życia servletu

• Podczas startu aplikacji lub pierwszego żądania, kontener web:

- ładuje klasę servletu
- tworzy instancję klasy servletu
- wywołuje metodę init() servletu

• Przetwarzanie żądań - wywoływanie metod obsługi żądań: metoda service() woła odpowiednią metodę doGet(), doPost, itd

• Zwalnianie zasobów

- wywołanie metody destroy()

Servlety | przetwarzanie żądania

1. Przeglądarka wysyła żądanie (request) HTTP do serwera
2. Serwer rozpoznaje żądanie do servletu (inicjalizuje go jeśli trzeba)
3. Żądanie klienta jest przetwarzane przez servlet
4. Servlet przygotowuje odpowiedź dla serwera
5. Odpowiedź przekazywana jest do serwera
6. Serwer odsyła odpowiedź (response) HTTP do przeglądarki

Servlet API

Klasy i interfejsy potrzebne do tworzenia serwletów są zawarte w dwóch pakietach: javax.servlet i javax.servlet.http.

Razem tworzą one podstawę Servlet API. Należy pamiętać, że wymienione pakiety nie są częścią podstawowych pakietów języka Java i jako takie nie należą do pakietu Java SE.

Okazuje się jednak, że są dołączane do serwera Tomcat i pakietu Java EE.

Servlet API

Pakiet javax.servlet:

Interfejs	Opis
Servlet	Zawiera deklaracje metod dotyczących cyklu życia serwletu
ServletConfig	Umożliwia serwletom dostęp do parametrów inicjalizacji
ServletContext	Umożliwia serwletom rejestrowanie zdarzeń i zapewnia dostęp do informacji na temat ich środowiska
ServletRequest	Interfejs używany do odczytywania danych z żądań
ServletResponse	Interfejs używany do zapisywania danych w odpowiedziach

Servlet API

Pakiet javax.servlet:

Klasa	Opis
GenericServlet	Implementuje interfejsy Servlet i ServletConfig
ServletInputStream	Hermetyzuje strumień wejściowy pozwalający na odczyt żądań
ServletOutputStream	Hermetyzuje strumień wyjściowy pozwalający na zapisywanie odpowiedzi
ServletException	Oznacza, że wystąpił błąd serwletu
UnavailableException	Oznacza, że serwlet jest niedostępny

Servlet API

Pakiet javax.servlet.http:

Interfejs	Opis
HttpServletRequest	Umożliwia odczytywanie danych z żądania HTTP
HttpServletResponse	Umożliwia zapisywanie danych w odpowiedzi HTTP
HttpSession	Pozwala na odczytywanie i zapisywanie danych dotyczących sesji

Servlet API

Pakiet javax.servlet.http:

Klasa	Opis
Cookie	Pozwala na przechowywanie informacji na komputerze klienta
HttpServlet	Zawiera metody pozwalające na obsługę żądań i odpowiedzi

API | *javax.servlet.GenericServlet*

Jest to podstawowa implementacją servletu, niezależna od protokołu.

Implementuje ona również interfejs Servlet i ServletConfig.

Raczej nie będziemy, z niej korzystać, ale gdybyśmy tego bardzo chcieli to powinniśmy ją rozszerzyć i zaimplementować w niej metodę:

service(ServletRequest req, ServletResponse res).

API | *Interfejs Servlet*

Wszystkie serwlety muszą implementować interfejs Servlet. Zawiera on deklaracje metod init(), service() i destroy(), wywoływanych przez serwer. Dostępna jest również metoda pozwalająca na pobranie przez serwlet parametrów inicjalizacji.

Metoda	Opis
void destroy()	Metoda wywoływana w celu usunięcia serwletu z pamięci
String getServletInfo()	Zwraca ciąg znaków opisujący serwlet
void init(ServletConfig *sc*) throws ServletException	Metoda wywoływana w momencie inicjalizacji serwletu. Parametr sc oznacza parametry inicjalizacji.
void service(ServletRequest *req*, ServletResponse *res*) throws ServletException, IOException	Metoda wywoływana w celu przetworzenia żądania przekazanego w parametrze *req*. Odpowiedź może być zapisana w obiekcie określonym przez parametr *res*.

API | *javax.servlet.http.HttpServlet*

Klasa HttpServlet poszerza klasę GenericServlet. Jest ona często wykorzystywana w sytuacjach, w których tworzone są serwlety pobierające i przetwarzające żądania HTTP.

Metody tej klasy to:

- doGet,
- doPost,
- doPut,
- doDelete,
- init and destroy,
- service.

Pozostałe jej metody przeciążamy rzadziej.

API | *javax.servlet.ServletConfig*

Interfejs ServletConfig umożliwia uzyskanie danych konfiguracyjnych podczas ładowania serwletu. Jego metody to:

Metoda	Opis
ServletContext getServletContext()	Zwraca kontekst dla danego serwletu
String getInitParameter(String *param*)	Zwraca wartość parametru inicjalizacji param
String getServletName()	Zwraca nazwę serwletu
Enumeration<String> getInitParameterNames()	Zwraca listę nazw wszystkich parametrów inicjalizacji

API | *javax.servlet.ServletRequest*

Interfejs ServletRequest zapewnia serwletowi dostęp do informacji na temat żądania.

Metoda	Opis
Object getAttribute(String *attr*)	Zwraca wartość atrybutu określonego w parametrze *attr*
String getCharacterEncoding()	Zwraca sposób kodowania żądania
String getContentType()	Zwraca typ żądania
String getParameter(String *pname*)	Zwraca wartość parametru określonego przez *pname*
Enumeration<String> getParameterNames()	Zwraca wyliczenie zawierające nazwy parametrów żądania

API | *javax.servlet.ServletResponse*

Interfejs ServletResponse umożliwia serwletowi generowanie odpowiedzi dla klienta.

Metoda	Opis
void setContentLength(int *size*)	Definiuje długość odpowiedzi (określoną w parametrze *size*)
void setContentType(String *type*)	Definiuje typ odpowiedzi (określony w parametrze *type*)
PrintWriter getWriter() throws IOException	Zwraca obiekt typu PrintWriter, który może zostać użyty do zapisania ciągu znaków w odpowiedzi.

API | *javax.servlet.ServletContext*

Interfejs ServletContext umożliwia serwletom uzyskiwanie informacji na temat ich środowiska.

Metoda	Opis
Object getAttribute(String *attr*)	Zwraca wartość atrybutu o nazwie określonej przez parametr *attr*
String getMimeType(String *file*)	Zwraca typ MIME elementu określonego przez parametr *file*
String getServerInfo()	Zwraca informacje dotyczące serwera
void setAttribute(String *attr*, Object *val*)	Przyporządkowuje wartość *val* atrybutowi określonemu przez parametr *attr*

Ćwiczenie

Odczytywanie parametrów serwletu

Interfejs ServletRequest zawiera metody, które pozwalają na odczytywanie nazw i wartości parametrów zawartych w żądaniu. Za chwilę utworzymy przykładowy serwlet, który ilustruje sposób ich zastosowania. Zostały w nim wykorzystane dwa pliki. Strona WWW jest zdefiniowana w pliku *PostParameters.html*, zaś sam serwlet w pliku *PostParametersServlet.java*.

Servlety | obsługa sesji

Sesja jest obiektem ze stanem aplikacji dla konkretnego użytkownika (łączenie przeglądarki z użytkownikiem następuje dzięki ciasteczku sesyjnemu).

Aby pobrać sesję w serwlecie należy wykonać metodę: „getSession()”.

Dodatkowo jeżeli podamy w niej parametr „true” to w wypadku braku sesji zostanie ona utworzona.

Aby pobrać atrybut w sesji wystarczy wykonać na niej metodę „getAttribute()”.

Servlety | obsługa sesji

- Dostępna za pomocą metody getSession() obiektu żądania
- Reprezentowana za pomocą klasy HttpSession
- Kluczowe metody:

1. getAttribute() i setAttribute()
2. invalidate() – czyści sesję o isNew() – określa, czy sesja została utworzona w ramach aktualnego żądania

W aplikacjach Java EE istnieje możliwość umieszczania danych w różnych zasięgach

Oprócz tego można deklarować atrybuty w zasięgach żądania i kontekstu aplikacji

Servlety | atrybuty i zasięgi

• W przypadku żądania dostęp do atrybutów regulują metody get/setAttribute typu HttpServletRequest

• Kontekst aplikacji to specjalny obiekt, który udostępnia informacje na temat całej aplikacji i umożliwia dynamiczne modyfikowanie struktury aplikacji (np. dodawanie serwletów)

• Atrybuty umieszczone w zasięgu kontekstu są dostępne w obrębie całej aplikacji!

• Dostęp do atrybutów jest możliwy za pomocą metody get/setAttribute typu ServletContext

• Instancja typu ServletContext jest dostępna na różne sposoby, m.in. przy użyciu obiektu żądania (req.getServletContext()) lub samego serwletu w metodach doGet()/doPost()

Servlety | atrybuty i zasięgi

Korzystając z atrybutów o zasięgu kontekstu należy pamiętac o możliwości współbieżnego dostępu przez różnych klientów

• Z tego względu warto synchronizowad metody/fragmenty kodu, które modyfikują mapę atrybutów

Ćwiczenie

Obsługa żądań GET protokołu HTTP

Za chwilę napiszemy serwlet obsługujący żądania GET protokołu HTTP. Będzie on wywoływany w momencie wysyłania formularza umieszczonego na stronie WWW. (Parametry URL)

Obsługa żądań POST protokołu HTTP

W tym punkcie napiszemy serwlet obsługujący żądanie POST protokołu HTTP. Jest on wywoływany w momencie, w którym następuje wysłanie formularza ze strony WWW.

Ćwiczenie

Korzystanie ze znaczników kontekstu użytkownika

Teraz napiszemy serwlet, który ilustruje sposób wykorzystania znaczników kontekstu użytkownika (cookies). Zostanie on wywołany w momencie wysłania formularza ze strony WWW.

Śledzenie sesji

Piszemy serwlet ilustruje sposób wykorzystania stanów sesji.

Filtry

Filtr to mechanizm, który pozwala na wykonanie pewnej czynności przed przekazaniem sterowania do odpowiedniego serwletu

- Filtry przekształcają zawartość żądania i/lub odpowiedz
- Są klasami javy implementującymi interfejs Filter
- Dzięki nim oddziela się powtarzalną logikę od specyficznej logiki konkretnego zasobu
- Umożliwiają wprowadzenie aspektu do specyficznej logiki
- Są realizacją wzorca projektowego Decorator
- Przetwarzanie/dopasowywanie filtrów jest natomiast realizacją wzorca projektowego Chain of Responsibility
- Konfigurację filtra można zawrzeć w adnotacji @Filter lub w pliku web.xml

Filtry

Filtry są stosowane wszędzie tam, gdzie konieczna jest wstępna obróbka żądania, np.:

do rejestrowania informacji w dzienniku

do szyfrowania/deszyfrowania

do kompresji

do przekazania sterowania do serwlet

do wywołania zdarzenia dostępu do zasobów

do konwersji obrazków

do logowania i audytu

do obsługi MIME-TYPE

do obsługi XSLT (transformacji XMLi)

Filtry

- Filtry, podobnie jak serwlety, mogą być aplikowane do wzorców URL:

```
<filter>
        <filter-name>Filtr</filter-name>
        <filter-class>sda.filtry.Filtr</filter-class>
</filter>
 <filter-mapping>
        <filter-name>Filtr</filter-name>
        <url-pattern>/serwlety/*</url-pattern>
</filter-mapping>
```

Filtry | cykl życia

- Podczas pierwszego żądania, kontener web

- ładuje klasę filtra

- tworzy instancję klasy filtra

- wywołuje metodę init() filtra

- Przetwarzanie żądań

- jeśli filtr podłączony jest pod żądany servlet, wykonywana jest logika filtra zawarta w metodzie doFilter()

- Zwalnianie zasobów

- wywołanie metody destroy() filtra

Filtry

Przykład:

```
public class FiltrExample implements Filter {
   public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws
IOException, ServletException {
      System.out.println("Przed obsługą");
      if (request.getParameter("user") != null) chain.doFilter(request, response);
      System.out.println("Po obsłudze");
   }

   public void init(FilterConfig fConfig) throws ServletException {
   }

   @Override
   public void destroy() {
   }
}
```

Ćwiczenie Servlet 4.0

Przekazywanie parametrów + sesja + filtr przechwytujący

Autor:

Ćwiczenie 1 do samodzielnego rozwiązania

Ćwiczenie 1:

Treść: Utwórz serwlet, który zwróci do klienta zdanie w formacie text/plain z zachowaniem polskich znaków.

Instrukcja rozwiązania: 1. Utwórz klasę dziedziczącą po klasie HttpServlet.

2. Zadeklaruj w niej metodę doGet(), w której znajdzie się treśd odpowiedzi.

3. Wygeneruj odpowiedź korzystając z obiektu typu PrintWriter (res.getWriter()), pamiętając o ustawieniu nagłówków.

4. Nie zapomnij o wpisie w pliku konfiguracyjnym web.xml lub adnotacji.

Ćwiczenie 2 i 3 do samodzielnego rozwiązania

Ćwiczenie 2:

Treść: Skonstruuj serwlet, który pozwoli na odrębną reakcję w zależności od tego, czy użytkownik odwiedził stronę po raz pierwszy, czy też nie.

Instrukcja rozwiązania: 1. Dodaj nowy serwlet do aplikacji lub skorzystaj z poprzedniego.

2. Skorzystaj z sesji, aby rozróżnie zachowanie serwletu.

Ćwiczenie 3:

Treść: Skonstruuj serwlet, który zwróci odpowiedź, jeśli w żądaniu będzie zawarte ciasteczko o pewnej przyjętej nazwie i wartości.

Instrukcja rozwiązania: 1. Dodaj nowy serwlet do aplikacji lub skorzystaj z poprzedniego.

2. Skorzystaj z kolekcji ciasteczek żądania, aby znaleźd i zweryfikowad pożądane ciasteczko.

Dziękuję

Niech Bytecode będzie z Tobą ☺