# Розділ 1
# Сервлети та JSP

## Лекція 1
Сервлети

## Визначення сервлета

Servlet (сервлет) — це Java-клас, призначений для разширення HTTP (веб) сервера.

Необхідні інтерфейси знаходяться в пакетах:
**javax.servlet**
**javax.servlet.http**

Java Servlet API — частина Java EE (Enterprise edition)
Актуальна версія - Java Servlet 4.0 API з підтримкою HTTP/2

Сервлети можуть працювати за різними прококолами

Сервлет - це скомпільований байт-код Java.
Тому для внесення змін у додатом необхідно компілювати сервлет повторно та завантажувати його на сервер.

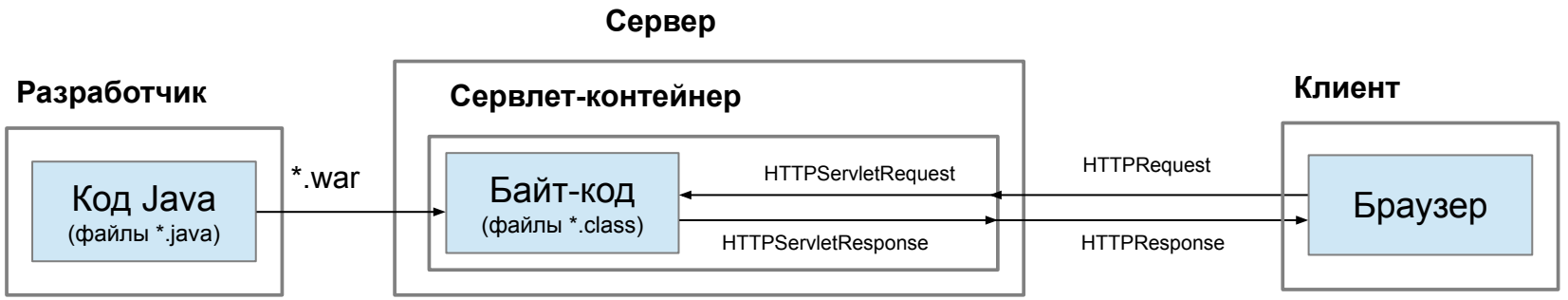
## Визначення JSP

JSP — це технологія Java, що дозволяє розробляти інтернет-ресурси в стилі скриптових сторінок.

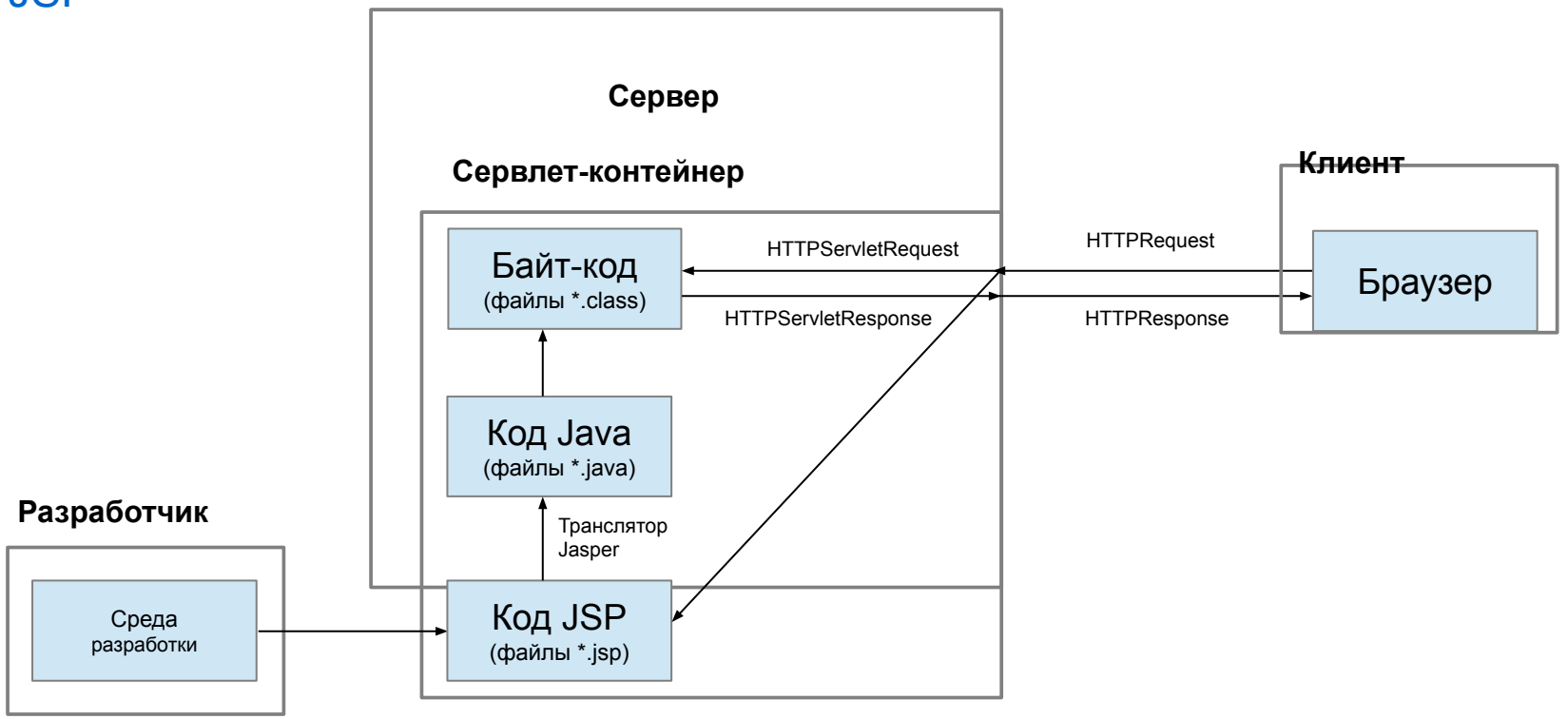
При першому зверненні до такої сторінки вона автоматично компілюється в сервлет і виконується.

Компіляція відбувається під час кожного виклику сторінки, якщо вона була змінена. Якщо зміни не відбувались – використовується скомпільована (попередньо) сторінка (іншими словами це вже буде сервлет)

## Servlet


Сервер
Разработчик
Сервлет-контейнер
Клиент
Код Java
(файлы *.java)
*.war
Байт-код
(файлы *.class)
HTTPServletRequest
HTTPServletResponse
HTTPRequest
HTTPResponse
Браузер

## JSP


Сервер
Сервлет-контейнер
Клиент
Байт-код
(файлы *.class)
HTTPServletRequest
HTTPServletResponse
HTTPRequest
HTTPResponse
Браузер
Код Java
(файлы *.java)
Транслятор
Jasper
Разработчик
Среда
разработки
Код JSP
(файлы *.jsp)

# Сервлет-контейнер

Сервлет-контейнер призначений для виконання класа сервлета в контексті веб-сервера

## Некомерційні

- Apache Tomcat (раніше Jakarta Tomcat)
- Apache Geronimo
- Glassfish (Sun microsystems, opensource)

## Комерційні

- Java System web-server/Application server (Sun microsystems)
- IBM Websphere (IBM)
- Oracle Application server (Oracle)
- JRun (Adobe)
- WebObjects (Apple)
- Borland Enterprise Server (Borland)
- BEA WebLogic (BEA Systems)
- JBoss (Red Hat, opensource)

# Структура додатку

**Програма ПОВИННА містити**

- Основну (базову) директорію (ім'я директорії - це ім'я додатку)
- **WEB-INF** директорію
- **web.xml** файл конфігурації (Deployment Descriptor - дескриптор розгортання)

**Програма МОЖЕ містити:**

- Servlets (що знаходяться у папці **WEB-INF\classes**)
- JSP файли
- HTML, JS, CSS та ін.
- Файли забражень, мультимедійні файли та файли інших типів
- Бібліотеки класів (jar-файли) (зазвичай в директорії **WEB-INF\lib**)

Базовая структура веб-додатку повинна містити:

- Базову (кореневу) директорію
- WEB-INF директорію (всередині базової)
- дескриптор розгортання web.xml (всередині WEB-INF)

Назва базової директорії буде частиною URL, що вказує на один із ресурсів, котрий міститься у додатку і використовується у якості назви додатку.

Наприклад, виклик index.jsp файла, який знаходиться в каталозі websample, може бути таким:

http://localhost:8080/**websample**/index.jsp

**web.xml** - конфігураційний файл використовується для:

- Оголошення класів servlet  та JSPs
- Відображення servlets та JSPs в URL шаблони
- Визначення welcom-сторінок
- Встановлення безпеки вмісту, ролей і методів аутентифікації

Executable класи додатку повинні бути розташовані у папці **WEB-INF\CLASSES**:

- Servlets
- Java Beans (використовуються у JSP)
- Tag libraries classes (використовуються у JSP)
- Helper classes

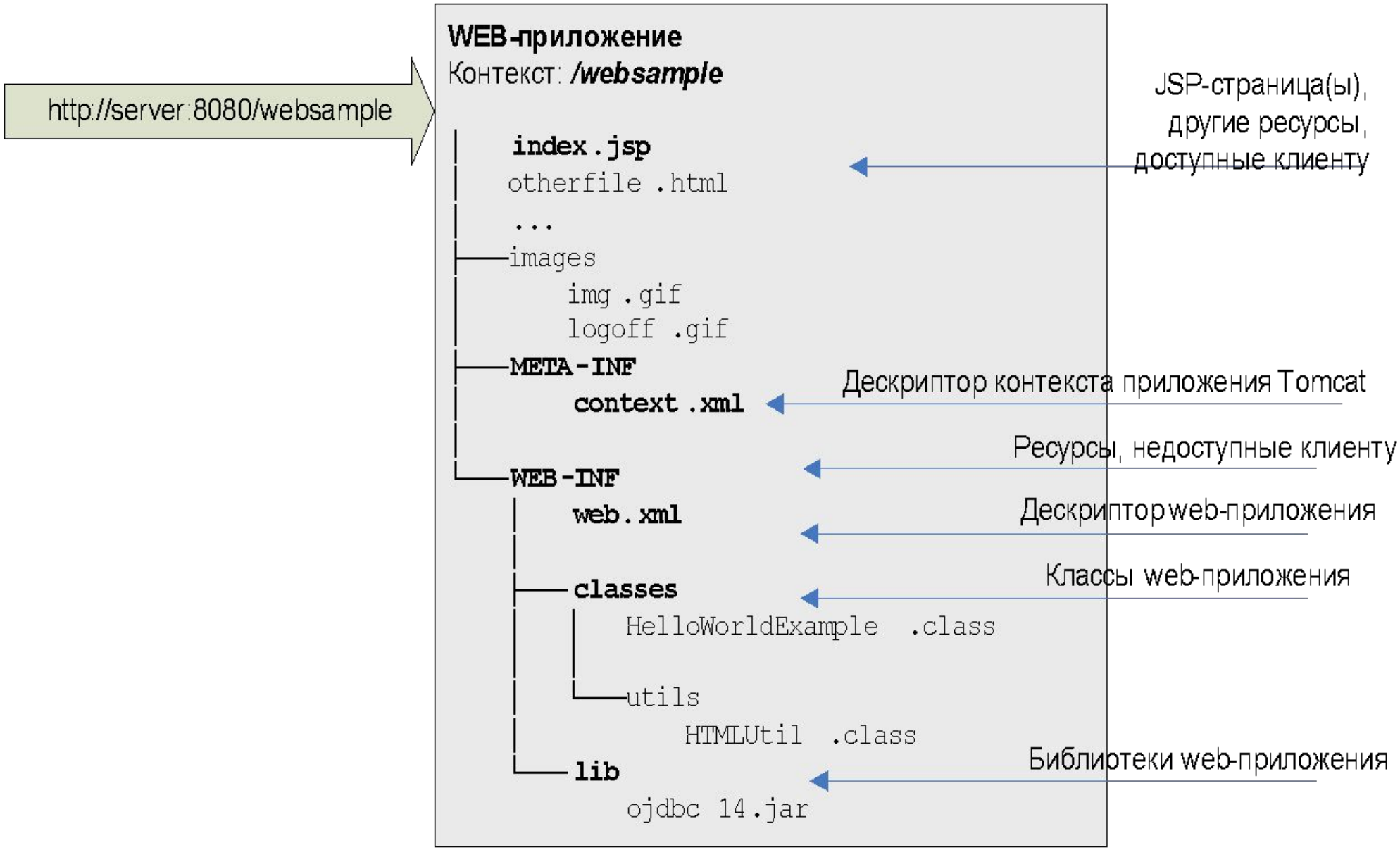
Всі класи, які розташовані у директорії classes, можуть бути згруповані у пакети.

Файли JSPs та статичні файли можут бути позміщені будь-де відносно базової директорії, але їх місцезнаходження буде відображено в URL, котрий використовується для їхнього виклику.

Наприклад, якщо index.html сторінка знаходиться у **websample\myproject** , URL буде:

**http://localhost:8080/websample/myproject/index.html**

# Структура додатку



# Структура сервлетів

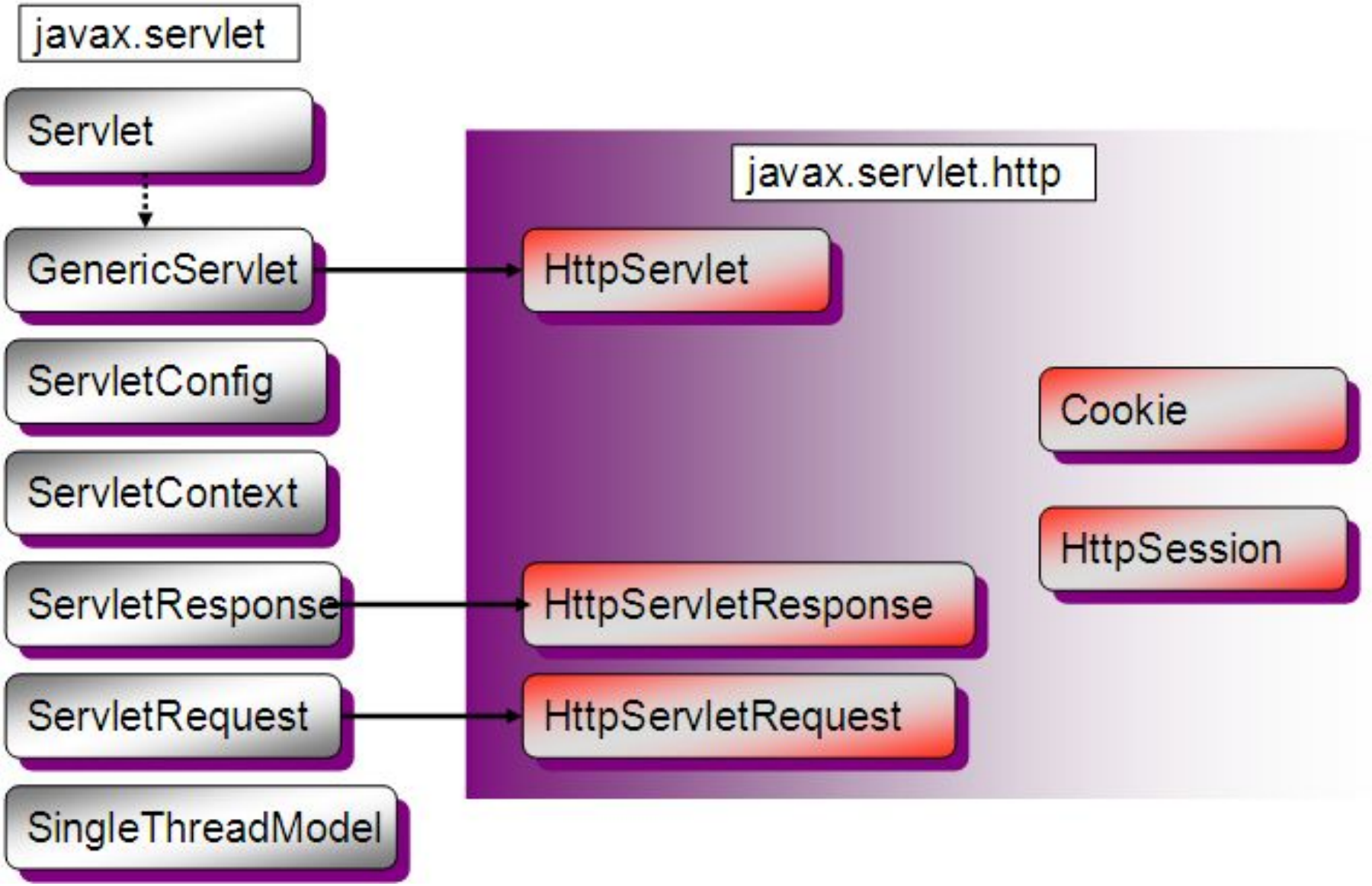
Необхідні інтерфейси роботи сервлетів знаходяться у пакетах javax.servlet, javax.servlet.http

Пакет **javax.servlet** забезпечує інтерфейси та класи для написання сервлетів. У ньому визначений інтерфейс **Servlet**. Всі сервлети реалізують цей інтерфейс.

Для работи по протоклу HTTP використовують розширення - **HttpServlet**.

Інтерфейс **Servlet** оголошує, але не реалізує методи, які управляють сервлетом і його взаємодією з клієнтами.

# Структура сервлетів



# Структура сервлетів

Приймаючи запит від клієнта, сервлет отримує два об'єкти:

- **ServletRequest**, який інкапсулює зв'язок клієнта з сервером связь клиента с сервером.
- **ServletResponse**, який інкапсулює зворотній зв'язок сервера з клієнтом.

Для протоколу HTTP ці об'єкти відповідно розширяються у:

- **HttpServletRequest –** інкапсулює інформацію ВІД клієнта
- **HttpServletResponse –** інкапсулює відповідь  клієнту

# Життєвий цикл сервлетів

Сервлети виконуються у на платформі веб-серверу, як частина того ж процесу, що і сам веб-сервер. Веб-сервер відповідає за ініціалізацію,  виклик та знищення кожного екземпляру сервлета. Веб-сервер взаємодіє з сервлетом через інтерфейс: **javax.servlet.Servlet.**

**Інтерфейс javax.servlet.Servlet складається з трьох основних методів:**

## init()

Підготовча робота з налаштування оточення сервлета перед виконанням клієнтського запиту.  Викликається перед будь-яким іншим зверненням до сервлети

## service()

Зчитує запит і формує відповідь за допомогою 2-ох аргументів – об'єктів **ServletRequest і ServletResponse**

## destroy()

Викликається для звільнення всіх ресурсів (наприклад, відкриті файли чи з'єднання з базою даних) перед завершенням роботи сервлета. Виклик може бути ініційований подією або за таймаутом сесії.

**та двох допоміжних методів:**

getServletConfig()  - забезпечує доступ до конфігурації сервера
GetServletInfo()  - повертає інформацію про сервлет: автор, версія та ін.

# Робота сервлетів

В класі HTTPServlet визначені методи doGet() и doPost(), які перевизначаються у сервлеті та забезпечують отримання даних від клієнта, використовуючи відповідний метод.

```
package servlets;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyFirstServlet extends HttpServlet {
@Override
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        PrintWriter out = response.getWriter();
       response.setContentType("text/html");
       out.println("<HTML><HEAD><TITLE>");
       out.println("ServletTest");
       out.println(""</TITLE></HEAD><BODY>");
       out.println(«Чи є життя після карантину? =^_^=");
       out.println("</BODY></HTML>");
       out.close();

    }
}
```

# Робота сервлетів

Як видно з прикладу клас MyFirstServlet розширює клас HTTPServlet та перевизначає його метод doGet().

Метод doGet() використовує Writer із об'єкта HttpServletResponse для того, щоб повертати клієнту текстову інформацію. Перед тим, як отримати доступ до Writer, встановлюється заголовок content-type. В кінці метода doGet(), після того як був надісланий запит, Writer закривається.

Під час кожного виклика методи doGet() і doPost() класу HttpServlet приймають параметром об'єкт, що реалізує інтерфейс HttpServletRequest. Веб-сервер, який виконує сервлет, створює об'єкт HttpServletRequest і передає його методу service() сервлета (який в сою чергу передає його методу doGet() або doPost()). Цей об'єкт містить запит від клієнта.

Як ми бачимо, виводити інформацію великою кількістю out.println досить незручно, оскільки великі та складні відповіді будуть містити сотні out.println

# Ключові методи об'єкта HttpServletRequest

| **Метод** | **Описание** |
|---|---|
| String getParameter(String name) | Отримання із запиту значення параметру. Ім'я параметру – вхідний параметр. |
| Enumeration getParameterNames() | Отримання із запиту всіх імен параметрів. |
| String[ ] getParameterValues(String name) | Для параметру з декількома значеннями метод повертає масив String. |
| Cookie[ ] getCookies () | Отримання масиву об'єктів Cookie, які збурігаються на комп'ютері клієнта. Cookie можуть використовуватись для унікальної фдентифікації клієнта сервером. |
| HttpSession getSession(boolean create) | Повертає об'єкт HttpSession сеансу клиента. Якщо параметр create дорівнює true та об'єкт HttpSession не існує, то створюється новий об'єкт HttpSession. |

# Ключові методи об'єкт HttpServletResponse

| Метод | Описание |
|---|---|
| void addCookie (Cookie cookie) | Використовується для додання Cookie в заголовок відповіді клієнту. |
| ServletOutputStream getOutputStream() | Отримання бінарного потоку виводу для відправки бінарних даних клієнту. |
| PrintWriter getWriter | Отримання символьного потоку виводу для відправки текстових даних клієнту. |
| void setContentType(String type) | Визначення MIME-типу відповіді браузеру. MIME-тип |

# Контекст сервера

Інформація про контекст сервера доступна в будь-який час через об'єкт **ServletContext**. Сервлет може отримати цей об'єкт викликавши метод getServletContext() об'єкта ServletConfig.

**Найбільш поширені методи контекста сервлета**

| **Метод** | **Описание** |
|---|---|
| GetMimeType () | Повертає тип MIME цього файла. |
| getRealPath () | Перетворює відносний або віртуальний шлях в новий шлях відносно кореневої директорії HTML-документів сервера. |
| getServlet () | Повертає об'єкт Servlet зазначеного імені. Корисний при доступі до служб інших сервлетів. |
| getServletNames () | Повертає список імен сервлетів, які доступні в поточному просторі імен. |
| log () | Записує інформацію у файл реєстрації сервлета. Імя файла реєстрації і його формат залежать від сервера. |

**Приклад запису інформації в лог**

```
import java.io.*;
 import javax.servlet.*;
 public MyFirstServlet implements Servlet
 {
     private ServletConfig config;
     public void init (ServletConfig config) throws ServletException
     {
         this.config = config;
         ServletContext sc = config.getServletContext();
         sc.log( "Started OK!" );
     }
}
```

# Об'єкт  Cookie

**Основними методами цього об'єкта є:**

getName()
getValue()
setValue()

**Приклад**

```
//Выбор значений всех ключиков
Cookie cookies [] = request.getCookies ();
for (int i = 0; i < cookies.length; i++) {
    out.println(cookies[i].getName()+"="+cookies[i].getValue()+"<br>");
}
```

```
Cookie cookie = new Cookie("key","value");
cookie.setMaxAge(60*60*24);
response.addCookie(cookie);
```

Деякі параметри сервера можна змінити в конфігураційному файлі – **web.xml**.

Приклад зміни часу сесії та встановлення шляху зберігання тимчасових файлів

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" 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">
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <context-param>
        <description>Location to store uploaded file</description>
        <param-name>file-upload</param-name>
        <param-value>
            C:\Program Files\Apache Software Foundation\Apache Tomcat 8.0.27\temp\
         </param-value>
    </context-param>
</web-app>
```

**Отримання в JSP в змінну fileTemp шляху, де зберігаються тимчасові файли**

```
<%
    ServletContext context = pageContext.getServletContext();
    String fileTemp = context.getInitParameter("file-upload");
%>
```

Для того, щоб сервлет правильно працював на сервері, необхідно зробити опис сервлета в файлі web.xml. Теги файлу web.xml наведені нижче

**<servlet>** - блок,  що описує сервлеты
- **<display-name>** - назва сервлета
- **<description>**  - текстовий опис сервлета
- **<servlet-name>** - ім'я сервлета
- **<servlet-class>** - клас сервлета
  - **<init-param>** - блок, що описує параметри  ініціалізації сервлета
    - **<param-name>**  - назва параметра
    - **<param-value>**  - значення параметра

**<servlet-mapping>** - блок, що описує відповідний url сервлета
- **<servlet-name>** - ім'я сервлета
- **<url-pattern>** - url-шаблон

**<session-config>** - блок, що описує параметри сессии
- **<session-timeout>** - максимальний час життя сесії

**<login-config>** - параметри авторизації копистувача
- **<auth-method>** - метод авторизації (BASIC, FORM, DIGEST, CLIENT-CERT)

**<welcome-file-list>** - блок, що описує імена файлів індексних файлів(будуть відкриватися при запиті директорії, без назви смого файлу). Сервер буде шукати перший існуючий файл зі списку.
- **<welcome-file>** - ім'я файла

**<error-page>** - блок, що описує помилки та завантаження відповідних їм сторінок
- **<error-code>** - код помилки
- **<exception-type>** - тип помилки
- **<location>** -  файл який завантажується

**<taglib>** - блок, що описує відповідність  JSP Tag library descriptor з URI-шаблоном
- **<taglib-uri>** - назва uri-шаблона
- **<taglib-location>** - розміщення шаблона

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" 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">
    <servlet>
        <servlet-name>authorisation</servlet-name>
        <servlet-class>servlets.authorisation</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>NewServlet</servlet-name>
        <servlet-class>servlets.NewServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>authorisation</servlet-name>
        <url-pattern>/user/profile/auth.php</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>NewServlet</servlet-name>
        <url-pattern>/NewServlet</url-pattern>
        <url-pattern>/TestServlet?Year=2020</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <context-param>
        <description>Location to store uploaded file</description>
        <param-name>file-upload</param-name>
        <param-value>
            C:\Program Files\Apache Software Foundation\Apache Tomcat 8.0.27\temp\
         </param-value>
    </context-param>
</web-app>
```