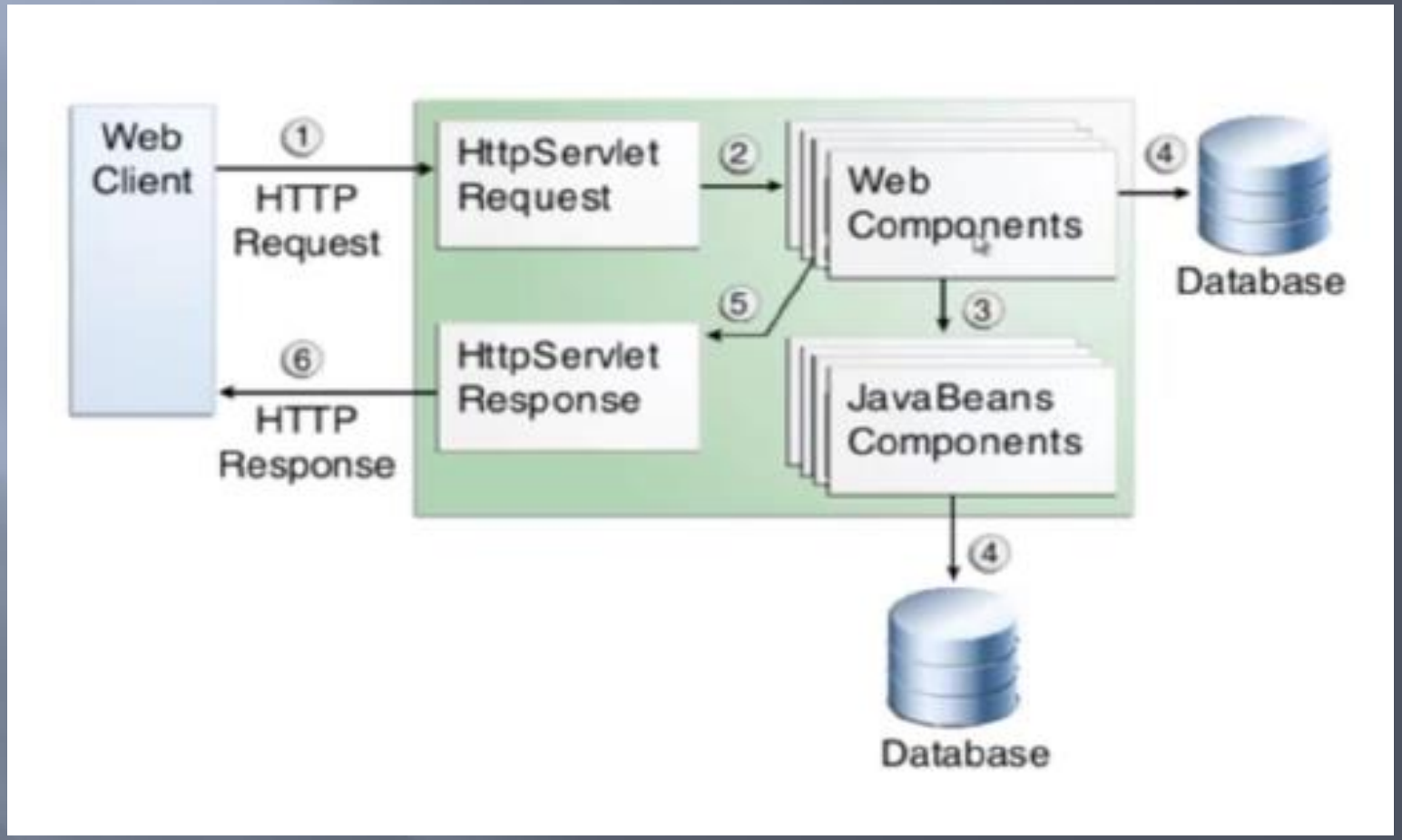
ВСТУП ДО JAVA EE

БАЗОВА АРХІТЕКТУРА JAVA WEB APPLICATION



ВИКОРИСТАННЯ SERVLET-КОНТЕЙНЕРІВ ТА JAVA WEB СЕРВЕРІВ

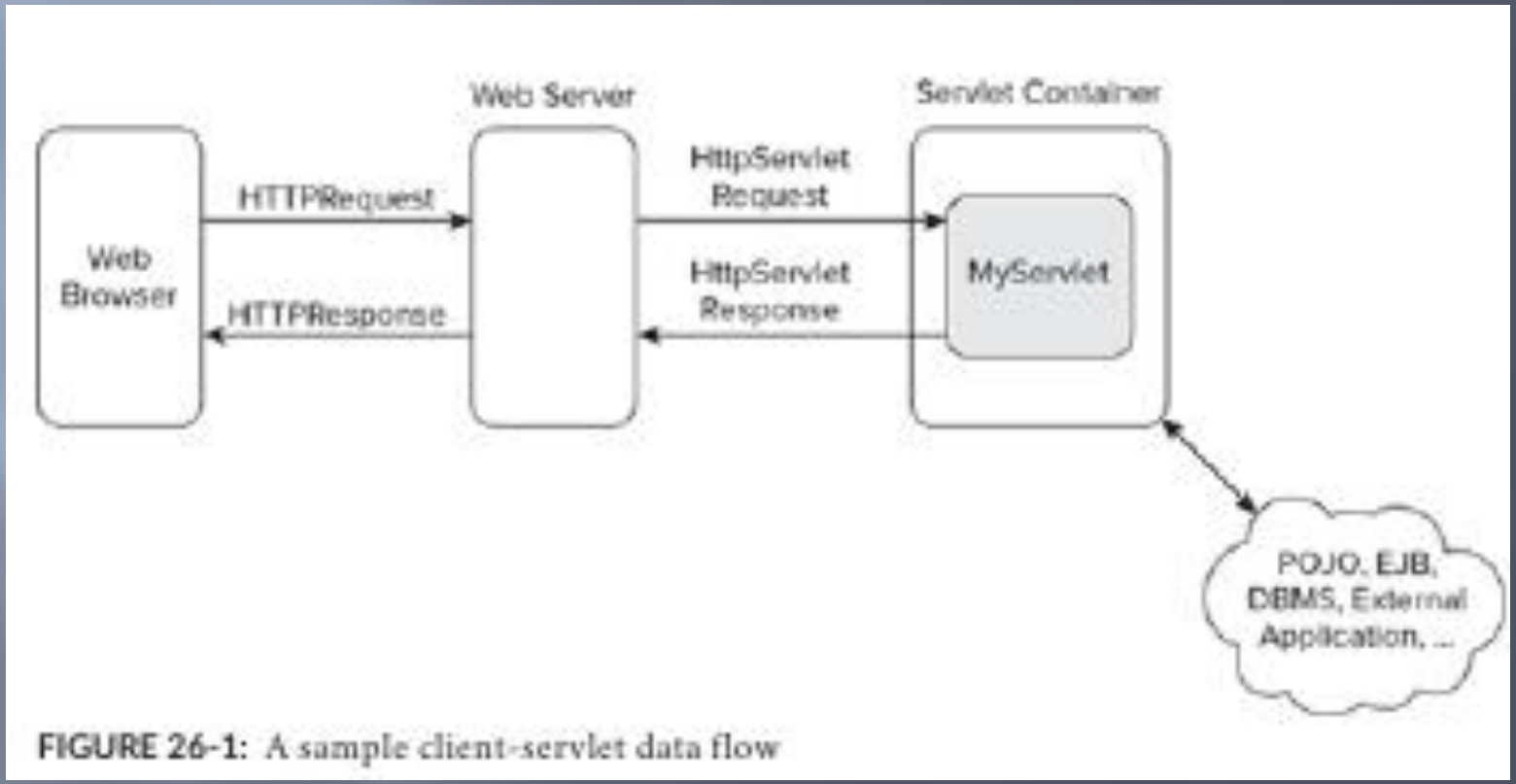


FIGURE 26-1: A sample client-servlet data flow

ПЕРЕВАГИ СЕРВЛЕТІВ

- Логіку роботи з сервером (прийом запиту і відправка результату) бере на себе Java
- Програмісту потрібно лише реалізувати обробку запиту
- Кросплатформність (як і у будь-якій Java програмі)

НЕДОЛІКИ СЕРВЛЕТІВ

Складний супровід коду.

Плутанина в коді великих проектів.

Обмежені можливості, зокрема, незручно створювати дизайн.

ЖИТТЄВИЙ ЦИКЛ СЕРВЛЕТА

- Відправка запиту клієнтом (по URL).
- Визначення сервлета, відповідального за обробку даного запиту (через файл настройок web.xml).
- Створення екземпляра потрібного сервлету.
- Обробка одного або декількох запитів.
- Знищення примірника сервлета.
- Сервлет ініціалізується 1 раз і компілюється в Java файл - кожне наступне звернення відбувається до вже скомпільованого файлу.
- Якщо сервлет зміниться - файл компілюється заново.

СЕРВЛЕТИ І БАГАТОПОТОЧНІСТЬ

За замовчуванням веб контейнер створює для кожного запиту окремий потік (1 запит - 1 потік).

Кожен створений потік звертається до одного і того ж екземпляру сервлету - в цьому і є небезпека.

Статичні змінні і змінні класу не є потокобезпечними, - потокобезпечні тільки змінні всередині методів doGet і doPost.

ПАКЕТИ ДЛЯ JAVA WEB

javax.servlet базовий пакет сервлетів

javax.servlet.http разширення для web-сторінок

javax.sevlet.jsp класи для JSP-сторінок

javax.sevlet.jsp.tagetxt розширення JSP- сторінок для бібліотек тегів

Пакети java web

- **javax.servlet** базовий загальний пакет сервлетів;
- **javax.servlet.http** спеціальні розширення для web-сторінок;
- **javax.sevlet.jsp** класи для створення JSP;
- **javax.sevlet.jsp.tagetxt** розширення класів JSP- сторінок, що дають можливість створювати бібліотеки тегів.

Інтерфейс сервлет

Интерфейс Servlet містить опис методів:

- **init(ServletConfig config)** викликається при ініціалізації сервелта;
- **destroy()** викликається перед вивантаженням сервлета з контейнера;
- **service(ServletRequest req, ServletResponse res)** спрацьовує при виклику сервлета оголошений в інтерфейсі **Servlet** реалізований в класі HttpServlet у вигляді дипетчера, що передає запити на **doGet, doPost** і т.д.

Клас HTTpServlet

- ***void doGet (HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException***; метод для обробки GET запитів
- ***void doPost (HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException***; метод для обробки POST запитів

За допомогою інтерфейсу HttpServletRequest опрацьовують HTTP-запити користувачів (параметри, заголовки запитів і т.д.).

За допомогою інтерфейсу HttpServletResponse формується HTTP-відповідь користувачеві (HTML або XML, відправка контенту)

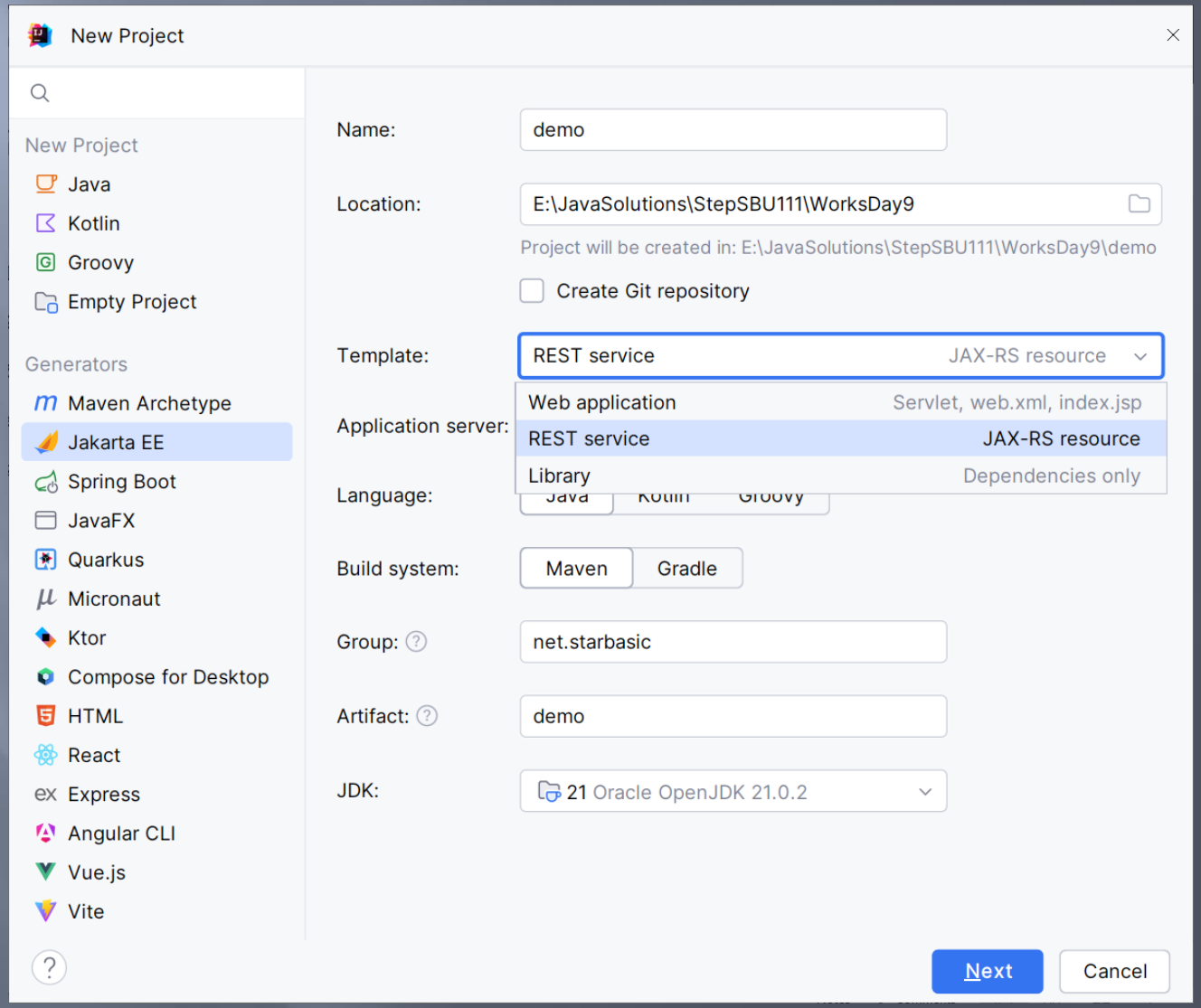
HTTPServlet, приклад

```
public class HelloServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter writer = response.getWriter();
        writer.println("Hello, World!");
        writer.close();
    }
}
```

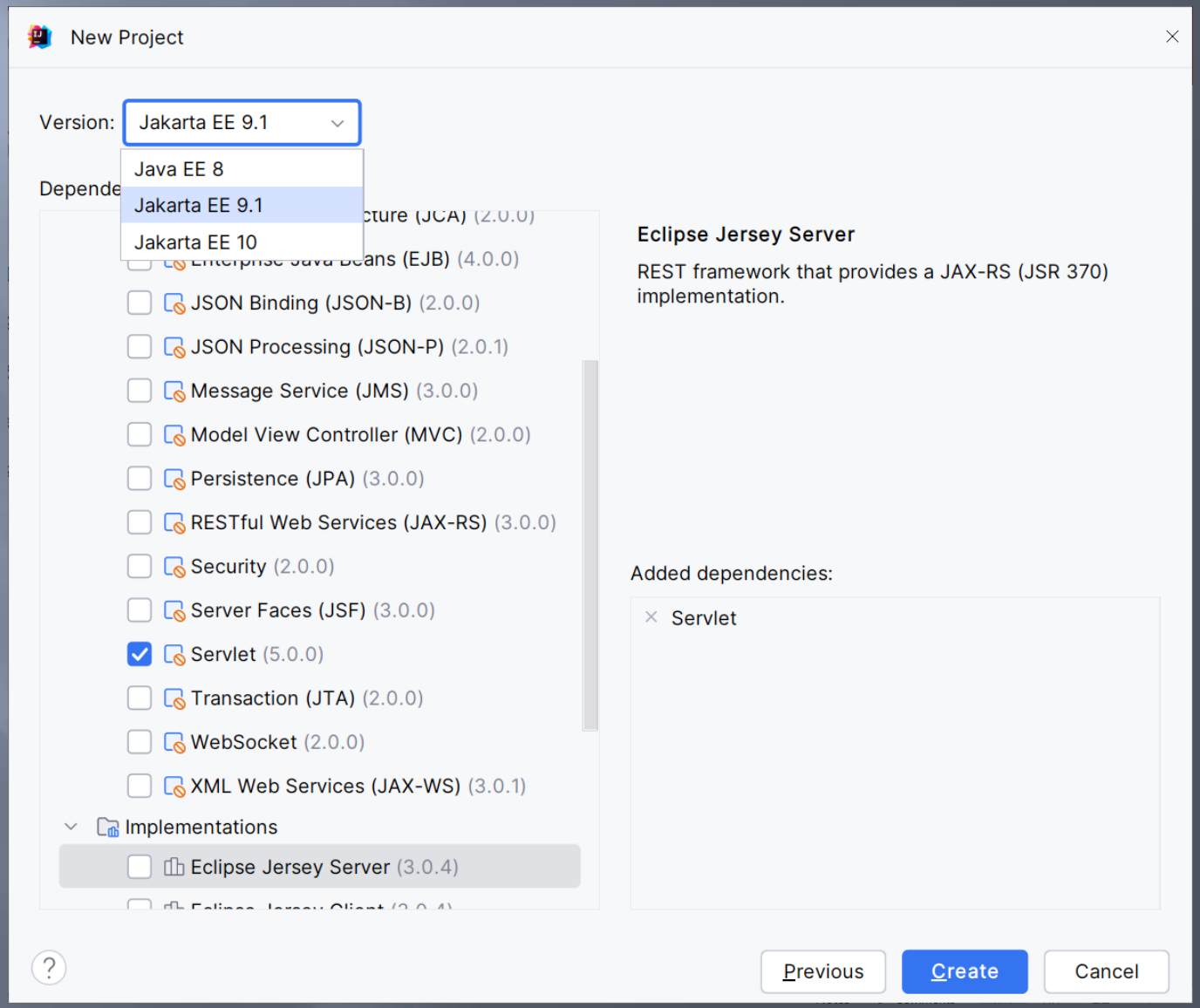
Реєстрація Servlet-а в Web.xml

```
<servlet>
<servlet-name>hello</servlet-name>
<servlet-class> net.firstweb.app.HelloServlet
</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>hello</servlet-name>
<url-pattern>/hello</url-pattern>
</servlet-mapping>
```

Java Webв Intellij Idea 2024



Java Webв Intellij Idea 2024



JSP

Jakarta Server Pages (JSP; раніше Java Server Pages) - це сукупність технологій, які допомагають розробникам програмного забезпечення створювати динамічно генеровані веб-сторінки на основі HTML, XML, SOAP або інших типів документів. Випущений у 1999 році Sun Microsystems, JSP схожий на PHP та ASP, але використовує мову програмування Java.

Для розгортання та запуску серверних сторінок Jakarta потрібен сумісний веб-сервер із контейнером сервлетів, такий як Apache Tomcat або Jetty.

Життєвий цикл JSP-сторінки

1. **Translation** Сервер додатків "парсить" код JSP і формує клас сервлета.
2. **Compilation** Сервер додатків компілює вихідний код JSP класу і створює клас (class).
3. **Class Loading** Контейнер JSP завантажує class сервлет в пам'ять.
4. **Instantiation** Впровадження конструкторів без параметрів для ініціалізації класу в пам'яті.
5. **Initialization** Викликається метод init об'єкта JSP. Після цієї фази JSP сервлет готовий до обробки запитів користувачів.
6. **Request Processing** обробка запитів клієнта JSP сторінкою. Багатопотона, аналогічно сервлетам.
7. **Destroy** Видалення JSP класу з пам'яті. Зазвичай відбувається при зупинці сервера.

JSP-Теги

У JSP 1.2 Було оголошено 5 основних типів тегів:

- <% @ директива атрибут = "значення"%>

<Jsp: directive.директива імяАтрибута = "значення" />

- Коментар <% -- Текст коментаря --%>

видимість коментара html і jsp різна, наприклад

<! - Початок коментаря <% = expression%> продовження коментаря ->

- Оголошення <%! одна або кілька декларацій%>
- Вирази <% = один вислів%>

<Jsp: expression> текст вирази </ jsp: expression>

- скріплет <% скріплет%>

<Jsp: scriptlet> текст скріптлета </ jsp: scriptlet>

JSP-Action

Tag	Purpose
<jsp:include>	Includes a file at the time the page is requested.
<jsp:useBean>	Finds or instantiates a JavaBean.
<jsp:setProperty>	Sets the property of a JavaBean.
<jsp:getProperty>	Inserts the property of a JavaBean into the output.
<jsp:forward>	Forwards the requester to a new page.
<jsp:plugin>	Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin.
< jsp:text >	Used to write template text in JSP pages and documents.

Неявні об’єкти JSP-сторінки

request - *екземпляр HttpServletRequest, що опрацьовується. Область видимості - запит. Основні методи: getAttribute, getParameter, getParameterNames, getParameterValues Об’єкт дає доступ до параметрів запиту через метод getParameter, типу запиту (GET, POST, HEAD, і т.д.), і тим, хто HTTP заголовках (cookies, Referer і т.д.).*

response - *відповідь сервера, екземпляр HttpServletResponse. Область видимості - сторінка.*

out - *об’єкт PrintWriter дає можливість запису в вихідний потік. Область видимості - сторінка. Основні методи: clear, clearBuffer, flush, getBufferSize, getRemaining. out використовується практично виключно скріплет, оскільки вираження JSP автоматично розміщуються в потік виводу.*

Неявні об’єкти JSP-сторінки

pageContext - *вміст JSP-сторінки. Область видимості - сторінка. Забезпечує доступ до специфічних об'єктів методами: getSession, getPage, findAttribute, getAttribute, getAttributeScope, getAttributeNamesInScope, getException.*

session - о*б'єкт типу Session. Область видимості - сторінка. Основні методи getId, getValue, getValueNames, putValue. змінна session існує навіть якщо немає посилань на вхідні сесії.*

application - *контекст сервлетв, отриманий з об'єкта конфігурації сервлета при виклику методів getServletConfig і getContext.*

config - *екземпляр ServletConfig поточної сторінки JSP.*

Exception - *екземпляр Throwable, виведений в сторінку помилок error page. Область видимості - сторінка. Основні методи: printStackTrace, toString, getMessage, getLocalizedMessage.*