МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем

та мереж

**Звіт**  до лабораторної роботи №4

з дисципліни “Екстремальне програмування ”

**Servlet API**

Виконав: студент гр. КН-311 Туз О.І.

Прийняв:

Щербак С.С.

Львів – 2020

Теоретичні відомості

Java Servlet API — стандартизований [API](https://uk.wikipedia.org/wiki/API) для створення [динамічного контенту](https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D1%96%D1%87%D0%BD%D0%B0_%D0%B2%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0) до [веб-сервера](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), використовуючи платформу [Java](https://uk.wikipedia.org/wiki/Java). Сервлети — аналог технологій [PHP](https://uk.wikipedia.org/wiki/PHP), [CGI](https://uk.wikipedia.org/wiki/CGI) і [ASP.NET](https://uk.wikipedia.org/wiki/ASP.NET). Сервлет може зберігати інформацію між багатьма транзакціями, використовуючи [HTTP](https://uk.wikipedia.org/wiki/HTTP) cookies, сесії або через редагування [URL](https://uk.wikipedia.org/wiki/URL).

Servlet API, що міститься в пакеті javax.servlet, описує взаємодію веб-контейнера і сервлета. Веб-контейнер — це компонент веб-сервера, що створений для взаємодії з сервлетами. Він відповідає за управління життєвим циклом сервлетів, перетворення URL у певний сервлет та забезпечення того, щоб клієнт, який зробив URL запит, мав відповідні права доступу.

Сервлети, [інтерфейси](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) та базові класи, протоколи роботи з ними, робоче оточення, описуються у відповідних специфікаціях компанії [Sun Microsystems](https://uk.wikipedia.org/wiki/Sun_Microsystems).

Для полегшення розробки [HTTP](https://uk.wikipedia.org/wiki/HTTP) сервлетів, у специфікації описано [абстрактний клас](https://uk.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BB%D0%B0%D1%81) HttpServlet, від якого розробникам пропонується [успадковувати](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) свої сервлети.

Схема роботи та застосування сервлетів:

1. Клієнт (наприклад, [Веб-оглядач](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4%D0%B0%D1%87)), відвідує веб-сторінку та надсилає HTTP запит на сервер.
2. Web-сервер отримує запит та передає його контейнеру сервлетів. Контейнер сервлетів може виконуватись в тому ж самому процесі, що і веб-сервер, в окремому процесі на тій же системі, що і веб-сервер, або взагалі в окремому процесі на іншій системі.
3. Контейнер сервлетів з'ясовує який сервлет слід викликати, виходячи з інформації про конфігурацію утримуваних сервлетів, та викликає його, передаючи як параметри об'єктні представлення запиту та відповіді.
4. Сервлет використовує об'єкт запиту для отримання інформації про віддаленого користувача, параметри HTTP запиту тощо. Сервлет виконує запрограмовані в ньому дії та надсилає результати роботи через об'єкт відповіді.
5. Після того, як сервлет припиняє обробку запиту, контейнер сервлетів перевіряє коректність відправки відповіді, та повертає управління до головного веб-сервера.

Сервлети, також, використовуються в технології [JSP](https://uk.wikipedia.org/wiki/JSP). Шаблони сторінок [транслюються](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) у вихідні тексти Java-класів успадкованих від стандартних класів сервлетів. Java-компілятор [компілює](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%82%D0%BE%D1%80) ці вихідні тексти в Java-байт коди. Отримані скомпільовані класи можуть використовуватись в сервлет-контейнері. Як правило, сервлет-контейнери виконують усі ці допоміжні дії автоматично.

Для практичної реалізації сервлетів використовується Tomcat.

Apache Tomcat — це контейнер [сервлетів](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82), розроблений [Apache Software Foundation](https://uk.wikipedia.org/wiki/Apache_Software_Foundation). Повністю написаний мовою програмування [Java](https://uk.wikipedia.org/wiki/Java) та реалізує специфікацію [сервлетів](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%BB%D0%B5%D1%82) і [Java Server Pages](https://uk.wikipedia.org/wiki/Java_Server_Pages) від [Sun Microsystems](https://uk.wikipedia.org/wiki/Sun_Microsystems), що є стандартами для розробки [веб-застосунків](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) на [Java](https://uk.wikipedia.org/wiki/Java).

Члени ASF і незалежні добровольці розвивають та підтримують Tomcat. Користувачі мають вільний доступ до вихідного коду Tomcat за умовами [Apache License](https://uk.wikipedia.org/wiki/Apache_License). Першою релізною версією Tomcat стала версія 3.0.x (попередні версії були випущені Sun для внутрішнього користування).

Виконання лабораторної роботи

Даний проект виконаний за допомогою мови програмування Java сервлетів і запущений за допомогою Apache Tomcat.

Частина коду html з проекту:

<**div class="reservation"**>  
 <**h1 id="reservation"**>Reservation</**h1**>  
 <**div class="inblock\_reserv"**>  
 <**form method="post" name="myemailform" action="notification" class="reservation\_form"**>  
 <**input type="text" name="name" placeholder="Your name" class="pole"**>  
 <**br**><**br**>  
 <**input type="text" name="surname" placeholder="Your surname" class="pole"**>  
 <**br**><**br**>  
 <**input type="email" name="email" placeholder="Your e-mail" class="pole"**>  
 <**br**><**br**>  
 <**input type="tel" name="phone" placeholder="Your phone number" class="pole"**>  
 <**br**><**br**>  
 <**input type="date" name="date" placeholder="Which day" class="pole"**>  
 <**br**><**br**>  
 <**input type="number" name="person\_amount" placeholder="Amount of person" class="pole"**>  
 <**br**><**br**><**br**>  
 <**input type="checkbox" name="special\_occasion" value="Special occasion"**>Special occasion  
 <**br**><**br**>  
 <**input type="checkbox" name="separate\_room" value="Separate room"**>Separate room  
 <**br**><**br**>  
 <**p**>Which hall do you prefer?</**p**>  
 <**select name="hall" class="pole"**>  
 <**option value="main"**>Main</**option**>  
 <**option value="outdoor"**>Outdoor</**option**>  
 <**option value="underground"**>Underground</**option**>  
 <**option value="terrace"**>Terrace</**option**>  
 </**select**>  
 <**br**><**br**>  
 <**textarea name="message" rows="7" cols="40" placeholder="Any advise?"**></**textarea**>  
 <**br**><**br**>  
 <**input type="submit" name='submit' value="Ok" class="for\_submit"**>  
 </**form**>  
 </**div**>

Клас Notification.java

**package** service;  
  
**import** model.Hall;  
**import** model.Order;  
**import** model.User;  
  
**import** javax.servlet.ServletException;  
**import** javax.servlet.annotation.WebServlet;  
**import** javax.servlet.http.HttpServlet;  
**import** javax.servlet.http.HttpServletRequest;  
**import** javax.servlet.http.HttpServletResponse;  
**import** java.io.IOException;  
**import** java.io.PrintWriter;  
  
  
@WebServlet(**"/notification"**)  
**public class** Notification **extends** HttpServlet {  
 @Override  
 **protected void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  
 String name = request.getParameter(**"name"**);  
 String surname = request.getParameter(**"surname"**);  
 String email = request.getParameter(**"email"**);  
 String phone = request.getParameter(**"phone"**);  
 String date = request.getParameter(**"date"**);  
 String personAmount = request.getParameter(**"person\_amount"**);  
 String specialOccasion = (**"Special occasion"**.equals(request.getParameter(**"special\_occasion"**))) ? **"Yes"** : **"No"**;  
 String separateRoom = (**"Separate room"**.equals(request.getParameter(**"separate\_room"**))) ? **"Yes"** : **"No"**;  
 String hall = request.getParameter(**"hall"**);  
 String message = request.getParameter(**"message"**);  
  
 User user = **new** User(name, surname, email, phone);  
 Order order = **new** Order(user, date, Integer.*valueOf*(personAmount), specialOccasion, separateRoom, Hall.*valueOf*(hall.toUpperCase()), message);  
  
 PrintWriter writer = response.getWriter();  
  
 **if** (order.checkFreeDate()) {  
 Mail.*sendMail*(order);  
 writer.println(AdditionalInfo.*getSuccessResponse*(order));  
 } **else** {  
 writer.println(AdditionalInfo.*getFailureResponse*());  
 }  
 }  
}

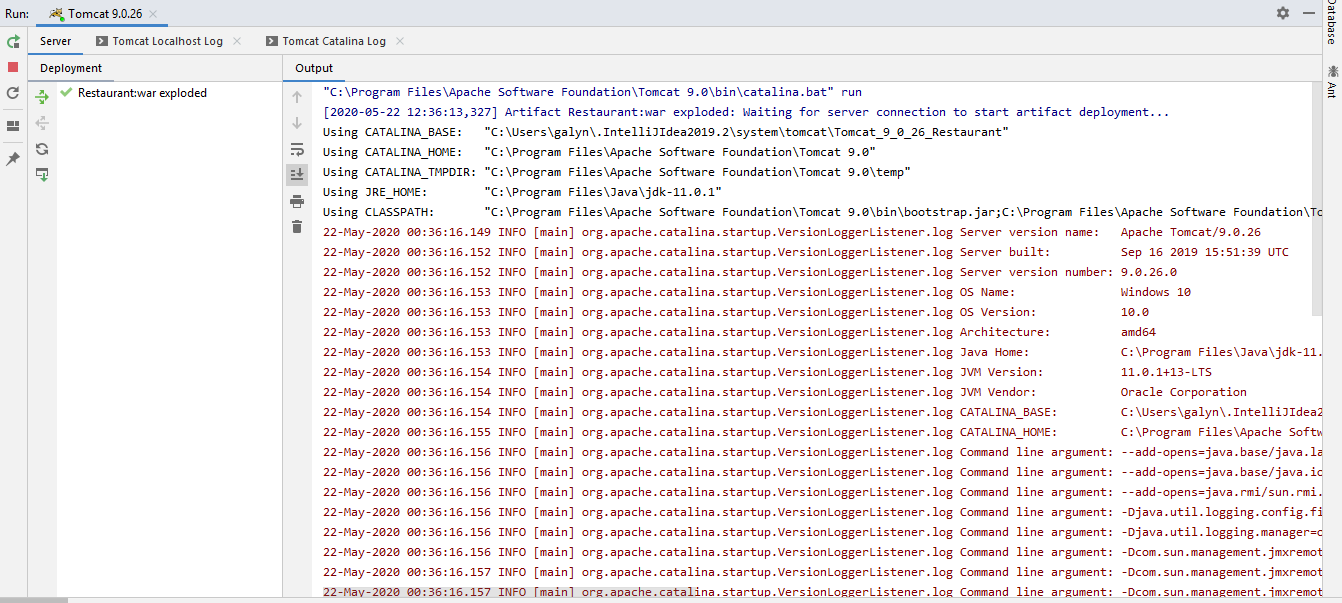


Рис.1 Завантаження Apache Tomcat

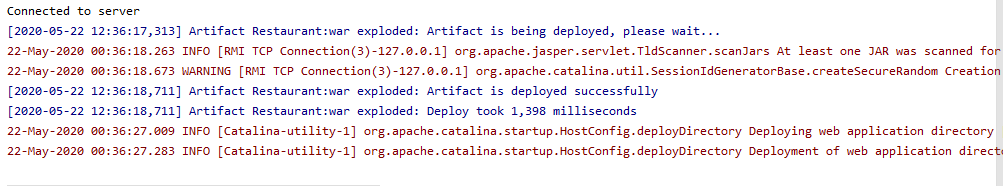


Рис.2 Результат приєднання до серверу

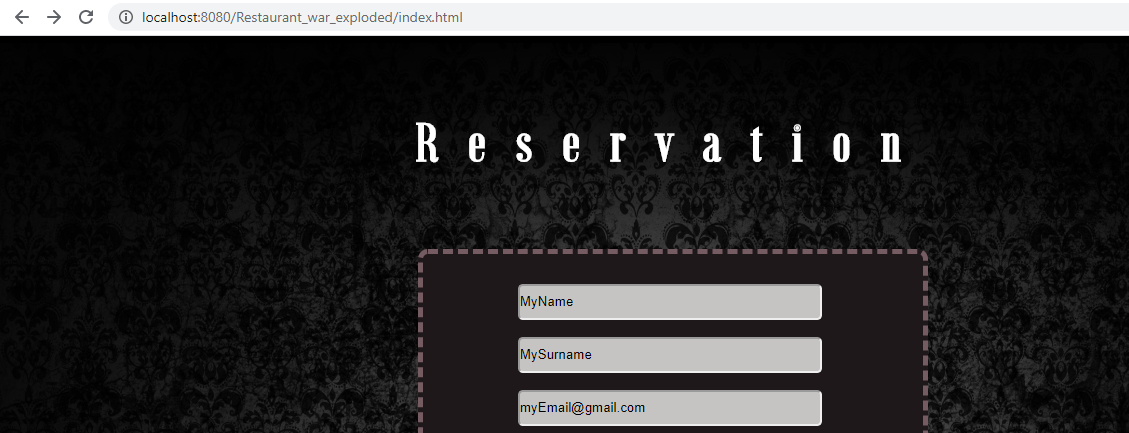


Рис.3 Початкова форма

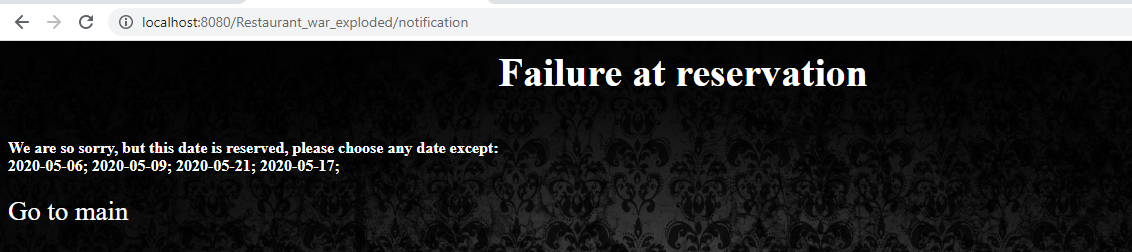


Рис.4 Результат відправлення форми

При запуску проекту запускається початкова сторінка із кореневим мапінгом ‘/’. На початковій html сторінці знаходиться форма із параметром action=”notification”, що означає, що при підтвердженні цієї форми програма буде шукати сервлет із мапінгом notification. З боку бекенду створюється клас Notification.java як розширює клас HttpServlet. Також необхідно задати сервлету відповідний мапінг, в нашому випадку /notification. Це можна зробити двома способами: за допомогою xml файлу, або за допомогою анотації @WebServlet. В даній лабораторній роботі використано сучасніший метод – анотація. Також в методі перевизначено метод doPost, який має два параметри: request і response. З request було витягнуто всі дані, передані з форми, а response – це відповідь, яку сервер віддасть кінцевому клієнту.

При запуску програми автоматично будується war архів проекту і запускається за допомогою Apache Tomcat. Інтерфейс доступний на localhost:8080.

Висновок: у даній лабораторній роботі було ознайомлено з базовими поняттями Servlet, а також реалізація частини коду веб-сайту за допомогою Servlet API і Apache Tomcat.