## Учреждение образования

## Белорусский государственный технологический университет

Кафедра полиграфического оборудования и

системы обработки информации

**ЛАБОРАТОРНАЯ РАБОТА № 5**

по дисциплине «Стандартизация и сертификация

информационных систем и технологий»

Тема

«Особенности технологий *Java EE*»

Выполнила студентка

4 курса 10 группы

Козловская А.М.

Проверил

доц. Беляев В.П.

Минск 2020

**Лабораторная работа № 5**

**ОСОБЕННОСТИ ТЕХНОЛОГИЙ *JAVA EE***

***Цель* *работы*:** Изучение особенностей технологий *Java EE*.

Привести в отчете развернутые ответы по следующим пунктам:

* Какие версии *Java EE* существуют?
* Объясните понятия *containers* и *application servers*.
* Какие технологии включает *Java* EE? Приведите список технологий, их назначение и соответствующие спецификации, а также эталонную реализацию для каждой.
* Объясните понятия *JCP* и *JSR.*

Приведите краткие выводы по содержанию работы.

**Какие версии *Java EE* существуют**

*Java EE* или *Java Enterprise Edition* представляет платформу для создания корпоративных приложений на языке *Java*. Прежде всего это сфера веб-приложений и веб-сервисов.

На рисунке 1 представлен список версий *Java EE*.



Рисунок 1 – версии Java EE

Основой *Java EE* был проект *JPE Project*, который стартовал в мае 1998 года. В декабре 1999 года вышел релиз платформы *Enterprise Java Platform (J2EE 1.2)*, которая объединяла такие компоненты как сервлеты, *JSP*, *EJB*, *JMS*.

В 2006 году с выходом 5-й версии она была переименована в *Java Enterprise Edition (JEE)*. С тех пор периодически выходят новые версии платформы. Последняя текущая версия - *Java EE* 8 вышла в сентябре 2017 года. В 2017 году произошла новая веха в развитии платформы: *Oracle* передал контроль над развитием *Java EE* организации *Eclipse Foundation*. В апреле 2018 года *Java EE* была переименована в *Jakarta EE*.

В начале 2019 года ожидается выход новой версии *Jakarta/Java EE*.

**Объясните понятия *containers* и *application servers.***

*Java EE* сервер приложений – продукт промежуточного уровня, который реализует набор *J2EE* технологий (спецификаций), тем самым предоставляя среду для развертывания, управления и выполнения *Java EE* компонентов.

*Java EE* компоненты выполняются в контейнерах сервера приложений, которые управляют их выполнениям.

Контейнеры являются коллекциями компонентов и обеспечивают интерфейс между компонентом и низкоуровневыми платформо-независимыми функциональными возможностями, поддерживающими компонент. До того, как веб-компонент, корпоративный компонент или компонент клиентского приложения может быть выполнен, он должен быть скомпонован в приложение *Java EE* и размещен внутри своего контейнера.

Различают следующие виды контейнеров *Java EE*:

* Сервер *Java EE* является частью времени исполнения приложения *Java EE*; предоставляет *EJB*- и веб-контейнеры.
* Контейнер *Enterprise JavaBeans (EJB)* управляет исполнением корпоративных компонентов для приложений *Java EE*. Корпоративные компоненты и их контейнер выполняются на сервере *Java EE*.
* Веб-контейнер управляет исполнением *JSP*-страниц и сервлетов для приложения *Java EE*. Веб-компоненты и их контейнер выполняются на сервере *Java EE*.
* Контейнер клиентского приложения управляет исполнением компонентов клиентского приложения. Клиентские приложения и их контейнер выполняются на стороне клиента.
* Контейнер апплетов: управляет выполнением апплетов. Состоит из веб-браузера и *Java plug-in*, выполняющихся совместно на клиенте.

Сервер приложения включает в себя как минимум следующие сервисы:

* *EJB*-контейнер, который поддерживает автоматическую синхронизацию *Java* объектов с базой данных (*CMP – container managed persistence, BMP – bean managed persistence*);
* *JMS* – сервис доставки сообщений между компонентами и серверами;
* управление ресурсами (доступ к СУБД, файловой системе, почтовому серверу и т. д.);
* безопасность и защита данных;
* поддержка транзакций (в том числе и распределённых, двухфазных). См. *Java Transaction API.*
* веб-сервер и сервлет-сервер;
* поддержка веб-сервисов;
* *JSF*.

**Какие технологии включает *Java EE*? Приведите список технологий, их назначение и соответствующие спецификации, а также эталонную реализацию для каждой.**

***Servlet*** – Обслуживание запросов веб-клиентов.

Приведенный ниже код реализует простой сервлет *HelloWorld.java*, возвращающий статическую *HTML*-страницу браузеру. В примере сервлет наследует свойства *HttpServlet*, реализующего интерфейс *Servlet*.

*import java.io.IOException;*

*import java.io.OutputStream;*

*import javax.servlet.ServletConfig;*

*import javax.servlet.http.HttpServlet;*

*import javax.servlet.ServletException;*

*import javax.servlet.http.HttpServletRequest;*

*import javax.servlet.http.HttpServletResponse;*

*public class HelloWorld extends HttpServlet*

*{*

*private ServletConfig config;*

*@Override*

*public void init (ServletConfig config) throws ServletException*

*{*

*this.config = config;*

*}*

*@Override*

*public void destroy() {}*

*@Override*

*public ServletConfig getServletConfig()*

*{*

*return config;*

*}*

*@Override*

*public String getServletInfo()*

*{*

*return "A Simple Servlet";*

*}*

*public void service (HttpServletRequest request, HttpServletResponse response)*

*throws ServletException, IOException*

*{*

*response.setContentType( "text/html" );*

*String text = "<!DOCTYPE html PUBLIC " +*

*"\"-//W3C//DTD HTML 4.01 Transitional//EN\" " +*

*"\"http://www.w3.org/TR/html4/loose.dtd\"> " +*

*"<html><head>" +*

*"<meta http-equiv=\"Content-Type\" " +*

*"content=\"text/html; charset=UTF-8\"> " +*

*"<title>Пример сервлета!</title>" +*

*"</head>" +*

*"<body>" +*

*"<h1>Здравствуй, %s %s!</h1>" +*

*"</body></html>";*

*text = String.format(text, config.getInitParameter("name"),*

*config.getInitParameter("mname"));*

*OutputStream outStream = response.getOutputStream();*

*outStream.write(text.getBytes("UTF-8"));*

*outStream.flush();*

*outStream.close();*

*}*

*}*

При обращении к сервлету вызывается метод *service*, который формирует страницу ответа и передает ее в браузер через *HttpServletResponse*. В первых строках ответа сервера определен заголовок *Content-Type* с указанием кодировки *UTF*-8. Дополнительно в шаблон ответа подставляются имя и отчество, получаемые из дескриптора приложения web.xml. То есть, выполняется чтение параметров конфигурации сервлета.

Для того, чтобы можно было бы возвращать страницу с кириллицей, используется *OutputStream*. В выходной поток можно вывести любую информацию (в нашем примере - простой *HTML*-код).

Ниже представлен листинг дескриптора приложения *web*.*xml*.

*<?xml version="1.0" encoding="UTF-8"?>*

*<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*

*xmlns="http://java.sun.com/xml/ns/j2ee"*

*xmlns:web="http://xmlns.jcp.org/xml/ns/javaee"*

*xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee*

*http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd*

*http://java.sun.com/xml/ns/j2ee*

*http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"*

*id="WebApp\_9" version="2.4">*

*<display-name>Servlet Hello World</display-name>*

*<servlet>*

*<servlet-name>hello</servlet-name>*

*<servlet-class>example.HelloWorld</servlet-class>*

*<init-param>*

*<param-name>name</param-name>*

*<param-value>Мария</param-value>*

*</init-param>*

*<init-param>*

*<param-name>mname</param-name>*

*<param-value>Юрьевна</param-value>*

*</init-param>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>hello</servlet-name>*

*<url-pattern>/hello</url-pattern>*

*</servlet-mapping>*

*</web-app>*

***EJB****. Enterprise JavaBeans* – спецификация технологии серверных компонентов, содержащих бизнес-логику.

***JSTL***. *JavaServer* *Pages* *Standard* *Tag* *Library* – представляет собой набор тегов в стиле HTML, позволяющих обращаться к объектам *Java* и выполнять многие из конструкций языка *Java*.

Подключение библиотеки тегов.

*// Основные теги создания циклов, определения условий, вывода информации на страницу и т.д.*

*<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>*

*// Теги для работы с XML-документами*

*<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>*

*// Теги для работы с базами данных*

*<%@ taglib prefix="s" uri="http://java.sun.com/jsp/jstl/sql" %>*

*// Теги для форматирования и интернационализации информации (i10n и i18n)*

*<%@ taglib prefix="f" uri="http://java.sun.com/jsp/jstl/fmt" %>*

Тэг *<c:out>* используется для вывода данных на экран. Следующий код отобразит выражение, представленное в атрибуте *value*.

*// отображение текста 16+64\*2*

*<c:out value="16+64\*2" />*

Чтобы было вычислено выражение атрибута *value* его необходимо оформить как выражение *EL (Expression Language*) в виде "${16+64\*2} .

*// отображение значения 144*

*<c:out value="${16+64\*2}" />*

***JSP****. JavaServer Pages* – динамическая генерация веб-страниц на стороне сервера.

Ниже представлен исходный код *JSP* страницы *index.jsp*

*<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">*

*<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>*

*<%@ page import="java.util.\*, java.text.\*" %>*

*<%!*

*String getFormattedDate()*

*{*

*SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy hh:mm:ss");*

*return sdf.format(new Date());*

*}*

*%>*

*<html>*

*<head>*

*<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">*

*<title>Добро пожаловать, JSP!</title>*

*</head>*

*<body>*

*<h1>Добро пожаловать!</h1>*

*<i>Сегодня <%= getFormattedDate() %></i>*

*</body>*

*</html>*

Исходный код сгенерированной страницы будет иметь следующий вид.

*<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">*

*<html>*

*<head>*

*<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">*

*<title>Добро пожаловать, JSP!</title>*

*</head>*

*<body>*

*<h1>Добро пожаловать!</h1>*

*<i>Сегодня 25.02.2020 21:08:56</i>*

*</body>*

*</html>*

***Security.*** *Java EE Security API*

Использование явной реализации представлено ниже

*<dependencies>*

*<dependency>*

*<groupId>javax.security.enterprise</groupId>*

*<artifactId>javax.security.enterprise-api</artifactId>*

*<version>1.0</version>*

*</dependency>*

*</dependencies>*

Пример обеспечения безопасности сервлетов, исопльзуя технологию *Java EE Security* *API* представлен ниже

*@WebServlet("/secured")*

*@ServletSecurity(*

*value = @HttpConstraint(rolesAllowed = {"admin\_\_role"}),*

*httpMethodConstraints = {*

*@HttpMethodConstraint(*

*value = "GET",*

*rolesAllowed = {"user\_\_role"}),*

*@HttpMethodConstraint(*

*value = "POST",*

*rolesAllowed = {"admin\_\_role"})*

*})*

*public class SecuredServlet extends HttpServlet {*

*}*

**Объясните понятия *JCP* и *JSR.***

*Java Community Process (JCP)* – сформированный в 1998 году формальный процесс, который позволяет заинтересованным лицам участвовать в формировании будущих версий спецификаций платформ языка *Java*.

Основа *JCP* – так называемые *JSR (Java Specification Request*/Запрос на Спецификацию *Java*), формальные документы, описывающие спецификации и технологии, которые предлагается добавить к Java платформе.

Разработка *JSR* включает следующие шаги:

* Предложение (*Initiation*). Спецификация предлагается членами сообщества, и, путём голосования, принимается Исполнительным Комитетом (*JCP Executive Committee*). В это время *JSR*, присваивается номер.
* Черновик для сообщества (*Community Draft*). После принятия *JSR,* формируется экспертная группа, которая разрабатывает первый черновик спецификации. Этот черновик выносится на рассмотрение Исполнительного Комитета и членов сообщества. По итогам обсуждения выносится решение о том, переходить к следующей стадии, или необходим новый доработанный *Community Draft*.
* Открытый черновик (*Public Draft*). Черновик, дошедший до этой стадии, выкладывается в свободный доступ, и любой человек, имеющий доступ к сети Интернет, может рассмотреть его и высказать своё мнение. Экспертная группа использует эти отзывы для дальнейшего совершенствования спецификации. После этого доработанная спецификация выносится на голосование Исполнительного Комитета для окончательного утверждения. К этому времени становятся доступными Эталонная реализация (*reference implementation*) и набор автоматических тестов *Technology Compatibility Kit* (*TCK*).
* Поддержка (*Maintenance*). Окончательная версия спецификации, эталонной реализации и *TCK*, обновляется по мере поступления запросов на доработку, уточнение и расширение. Некоторые запросы могут потребовать пересмотра спецификации экспертной группой и выпуска новой версии, или даже формирования нового *JSR*.

**Выводы:**

В ходе выполнения лабораторной работы была изучена платформа *Java EE*, позволяющая создавать приложения на языке *Java*. Были рассмотрена технологии, которые включает в себя платформа. К примеру, разработан пример использования *Servlet* технологии – создан файл, обрабатывающий запросы веб-клиента. Данный функционал можно реализовать и на других платформах, однако *Java EE* предоставляет готовые технологии, используя который желаемый результат был получен менее трудозатратно.

*Java EE* также имеет технологию, позволяющую динамически генерировать страницы на стороне сервера, что очень полезно для решения определенных задач. Сгенерированная в процессе выполнения лабораторной работы страница отображается в браузере желаемым образом, ее генерация на стороне сервера дает ряд преимуществ, такие как, к примеру, вставка динамически вычисляемого значения текущей даты. Можно сделать вывод, что *Java EE* хороша мобильностью, сетью готовых разработок (*API*), а также набором готовых продуктов для обеспечения безопасности и надежности, что позволяет создавать приложения быстрее.