БИНГО! Ты только что нашел решение своей проблемы! Только давай договоримся – ты прочтёшь текст до конца, окей? :)

Давай начистоту: тут один шлак, лучше закажи работу на **[author-24.pro](https://author-24.pro/)** и не парься – мы всё сделаем за тебя! Даже если остался **один** **день** до сдачи работы – мы справимся, и ты получишь «Отлично» по своему предмету! Только представь: ты занимаешься своим любимым делом, пока твои одногруппники теряют свои нервные клетки…

Проникнись… Это бесценное ощущение :)

Курсовая, диплом, реферат, статья, эссе, чертежи, задачи по матану, контрольная или творческая работа – всё это ты можешь передать нам, наслаждаться своей молодостью, гулять с друзьями и радовать родителей отличными оценками. А если преподу что-то не понравится, то мы **бесплатно** переделаем так, что он пустит слезу от счастья и поставит твою работу в рамочку как образец качества.

Ещё сомневаешься? Мы готовы подарить тебе **сотни часов свободного времени** за смешную цену – что тут думать-то?! Жизнь одна – не трать её на всякую фигню!

Перейди на наш сайт **[author-24.pro](https://author-24.pro/)** - обещаю, тебе понравится! :)

А работа, которую ты искал, находится ниже :)

Факультет «Информатика и системы управления»

Методические указания к лабораторной работе

по курсу «Распределенные системы обработки информации»

*"Разработка Web‑приложений с применением Java Servlet Технологии"*

Москва 2004 г.

***Цель работы***

Получить знания о Java Servlet технологии, сервере Jakarta Tomcat, его настройке для работы с сервлетами.

# 

# *Задание для домашней подготовки*

Ознакомиться с материалом, предоставленным в приложении к данным методическим указаниям. Ознакомиться с текстом задания к лабораторной работе. Написать Web – приложение.

# 

# *Задания к лабораторной работе*

Написать пользовательский интерфейс online форума.

# 

# *Содержание отчета*

Отчет должен содержать:

1. Постановку задачи, решаемой отлаженных программ.
2. Руководство пользователя отлаженных программ, содержащее описание интерфейсов всех функций программ.
3. Руководство системного программиста в объеме, необходимом для последующего сопровождения (модификации) отлаженных программ другим программистом (само документируемый листинг программы, необходимые пояснения алгоритма и его программной реализации).

# 

# *Контрольные вопросы*

1. Преимущества Java Servlet?
2. Из каких этапов состоит жизненный цикл сервлета?
3. Способы вызова сервлетов с браузера и страницы?
4. Способы передачи параметров сервлету и их обработка?
5. Как происходит запись и чтение атрибутов сессии?

# *Литература*

Java in a Nutshell Дэвид Флэнэген Киев 1998

http://faq.realcoding.net/index.php

http://java.sun.com/j2ee/learning/tutorial/index.html

http://egor.spb.ru/doc/db/mysql/14.html#\_ftnref1

http://www.coreservlets.com/Apache-Tomcat-Tutorial/

http://gcc.gnu.org/ml/java-patches/2000‑q4/index.html#00245

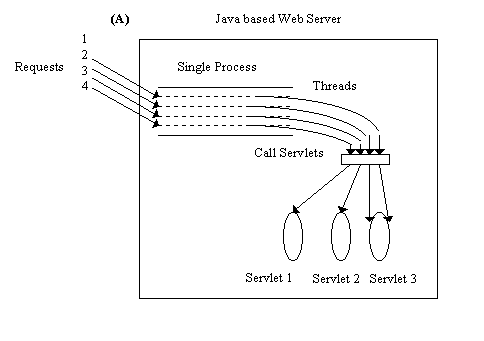
http://www-ucs.usc.edu/~toms/jakartafaq.html

http://jakarta.apache.org/faq

***Теоретическая часть***

### Сервлеты Java

Сервлеты расширяют функциональность Web‑серверов и являются мощным средством программирования. Сервлеты – это модули, расширяющие возможности серверов типа запрос-ответ, таких как Java‑совместимые web‑сервера. Сервлеты для серверов то же самое, что и апплеты для браузеров, но в отличие от апплетов, у сервлетов нет графического интерфейса пользователя. Сервлеты могут быть встроены в различные серверы, так как API сервлетов, который используется для их написания, не предполагает никакого знания ни об окружении сервера, ни о его протоколе. Сервлеты наиболее широко используются с HTTP‑серверами; многие web‑сервера поддерживают API сервлетов. Они обеспечивают такой способ генерирования динамических документов, который позволяет облегчить их написание и ускорить выполнение. Сервлеты также решают проблему программирования на стороне сервера с платформо-зависимыми API, которые разрабатываются с использованием Java Servlet API, стандартного расширения Java. Характерной особенностью сервлетов является то, что они не требуют создания новых процессов при каждом новом запросе. Множество сервлетов выполняются параллельно в рамках одного процесса на сервере.



**Java Servlet Development Kit (JSDK)**

Набор инструментов разработки Java – сервлетов(JSDK) содержит библиотеки классов, которые нужны для создания сервлетов. Он включает также утилиту, известную как servletrunner, которая дает возможность проверить некоторые из создаваемых сервлетов. Начиная с JSDK Version 2 пакет Jsdk2.0 встроен в JDK.

### Servlet API.

Коды, требующиеся для построения сервлетов, содержат два пакета: javax.servlet и javax.servlet.http. Вместе они представляют Servlet API.

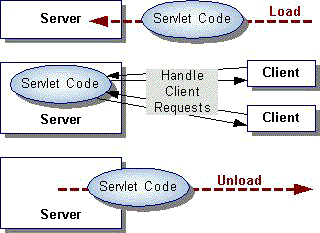
### Интерфейс сервлета

Центральной абстракцией в API сервлета является Servlet API‑интерфейс. Все сервлеты реализуют этот интерфейс либо прямо, либо, чаще, путем расширения класса, который реализует этот интерфейс, например класса HttpServlet API. Servlet‑интерфейс объявляет, но не реализует методы, управляющие сервлетом и его связью с клиентами. Разработчики сервлетов обеспечивают реализацию отдельных (или всех) методов в процессе разработки сервлета.

### Жизненный цикл сервлета

Все сервлеты имеют одинаковый жизненный цикл:

* Сервер загружает и инициализирует сервлет
* Сервлет обрабатывает ноль или более запросов клиентов
* Сервер выгружает сервлет



**Вызов сервлетов с браузеров**

Для вызова сервлета с браузера достаточно набрать запрос – URL сервлета. URL сервлета имеет следующую форму:

http://machine-name:port/servlet/servlet-name

где servlet-name относится к имени, данному этому сервлету. Например

http://localhost:8080/servlet/bookstore

URL сервлетов может содержать запросы, такие как HTTP GET‑запросы. Например

http://localhost:8080/servlet/bookdetails? bookId=203

### Вызов сервлетов с HTML‑страницы

Для вызова сервлета с HTML‑страницы достаточно использовать URL сервлета с соответствующей HTML‑меткой. Метки, поддерживающие URL, включают те, с которых начинаются якоря и формы, а также meta‑метки. Например

<a href=«http://localhost:8080/servlet/cashier>Calls the servlet</a>

Также для вызова сервлета можно использовать атрибут action в теге form в HTML‑странице, указав URL сервлета. Например

<form action= «http://localhost:8080/servlet/receipt» method= «post»>

Для вызова сервлета также можно использовать тег meta в HTML‑странице, указав URL сервлета. Например

<meta http-equiv= «refresh» content= «4; url=http://localhost:8080/servlet/bookstore;»>

### Пакет javax.servlet

Пакет javax.servlet содержит ряд интерфейсов и классов, устанавливающих обрамление, в котором работают сервлеты.

|  |  |
| --- | --- |
| **Интерфейс** | **Описание** |
| Servlet | Объявляет методы цикла жизни для сервлета. |
| ServletConfig | Позволяет сервлетам получать параметры инициализации. |
| ServletContext | Активизирует возможности сервлетов для регистрации событий и доступа к информации об их среде. |
| ServletRequest | Используется для чтения данных из запроса клиента. |
| ServletResponse | Используется для записи данных в ответ клиенту. |
| SingleThreadModel | Указывает, что сервлет защищен от многопоточности. |

|  |  |
| --- | --- |
| **Класс** | **Описание** |
| GenericServlet | Реализует интерфейс Servlet и ServletConfig |
| ServletInputStream | Обеспечивает входной поток для чтения запросов от клиента. |
| ServletOutputStream | Обеспечивает выходной поток для записи ответов клиенту. |
| ServletException | Указывает, что произошла ошибка сервлета. |
| UnavailableException | Указывает, что сервлет постоянно или временно недоступен. |

#### Инициализация сервлета

Когда сервер загружает сервлет, он запускает init‑метод сервлета.

public void init (ServletConfig config) throws ServletException

Инициализация завершается до начала обработки запрос клиента и до удаления сервлета. Сервер вызывает метод init только один раз при загрузке сервлета. В дальнейшем, метод init больше не вызывается, если только сервер не перезагружает сервлет. Сервер не может перезагрузить сервлет до тех пор, пока он не удалит сервлет, вызвав метода destroy.

#### Удаление сервлета

Сервлеты работают до тех пор, пока сервер не удалит их. Когда сервер удаляет сервлет, он запускает метод destroy сервлета.

public void destroy()

Этот метод запускается один раз; сервер не будет запускать сервлет снова до тех пор, пока сервер не перезагрузит и переинициализирует сервлет.

#### Взаимодействие с клиентами

После инициализации сервлет способен обрабатывать запросы клиентов. Cервлет работает с запросами клиентов с помощью метода service.

public void service (ServletRequest req, ServletResponse res)

throws ServletException, IOException

Метод service поддерживает стандартные HTTP‑запросы клиентов, отсылая каждый полученный запрос тому методу, который создан для работы с запросом данного вида.

Когда сервлет принимает вызов клиента, он получает два объекта:

* ServletRequest, который инкапсулирует связь клиента с сервером.
* ServletResponse, инкапсулирующий обратную связь сервера с клиентом.

ServletRequest и ServletResponse являются интерфейсами, определенными пакетом javax.servlet.

### ServletRequest интерфейс

ServletRequest интерфейс позволяет сервлету получить доступ:

* к следующей информации: имена параметров, введенных клиентом; протокол (схема), используемый клиентом; имя удаленного хоста, пославшего запрос; имя сервера, получившего этот запрос.
* к входному потоку ServletInputStream для получения данных от клиентов, использующих такие входные протоколы, как HTTP POST- и PUT‑методы.

Интерфейсы, расширяющие ServletRequest интерфейс, позволяют сервлету получить и другие данные конкретного протокола. Например, HttpServletRequest интерфейс содержит методы, позволяющие получить доступ к информации HTTP‑заголовка.

### ServletResponse интерфейс

ServletResponse интерфейс обеспечивает сервлет методами, необходимыми для ответа клиенту. Этот интерфейс:

* позволяет сервлету установить длину содержимого и MIME‑тип ответа.
* обеспечивает выходной поток ServletOutputStream и Writer, посредством которых сервлет может послать данные ответа.

Интерфейсы, расширяющие ServletResponse интерфейс, предоставляют сервлету возможности конкретного протокола.

### Чтение параметров сервлета и формирование ответа клиенту

Класс ServletRequest включает методы, позволяющие читать имена и значения параметров, которые включены в запрос клиента.

String getParameter (String pname) – возвращает значение параметра с именем pname.

Enumeration getParameterNames() – возвращает перечисление имен параметров этого запроса.

String[] getParameterValues() – возвращает перечисление значений параметров этого запроса.

### Чтение параметров инициализации

Параметры инициализации могут использоваться для открытия файлов, создания подключений базы данных или выполнения других действий. К этой информации можно обращаться двумя способами:

* Метод init(), объявленный интерфейсом Servlet, принимает объект ServletConfig в качестве его параметра;
* Метод getServletConfig(), объявленный интерфейсом Servlet, возвращает объект ServletConfig.

Манера, в которой параметры инициализации предоставляются сервлету, зависит от сервера.

### Пакет javax.servlet.http

Пакет javax.servlet содержит ряд интерфейсов и классов, облегчающих построение сервлетов, которые работают с HTTP – запросами и ответами.

|  |  |
| --- | --- |
| **Интерфейс** | **Описание** |
| HttpServletRequest | Используется для чтения данных из HTTP – запроса клиента. |
| HttpServletResponse | Используется для записи данных в HTTP – ответ клиенту. |
| HttpSession | Позволяет читать и записывать данные сеанса |
| HttpSessionBindingListener | Информирует объект о том, что он связан или не связан с сеансом. |
| HttpSessionContext | Обеспечивает управляемость сеансов |

|  |  |
| --- | --- |
| **Класс** | **Описание** |
| Cookie | Позволяет сохранять информацию состояния на машине клиента |
| HttpServlet | Обеспечивает методы для обработки запросов и ответов HTTP |
| HttpSessionBindingEvent | Указывает на наличие или отсутствие связи блока прослушивания с сеансовым значением |
| HttpUtils | Обеспечивает методы утилит для сервлетов. |

### Класс HttpServlet. Работа с HTTP – запросами

Методы, которым метод service передает HTTP‑запросы:

* doGet, для работы с GET-, условными GET-, и HEAD‑запросами
* doPost, для работы с POST‑запросами
* doPut, для работы с PUT‑запросами
* doDelete, для работы с DELETE‑запросами

По умолчанию, эти методы возвращают ошибку BAD\_REQUEST (400). Сервлет должен переопределить либо метод service, либо методы для работы с теми HTTP‑запросами, которые он поддерживает. Метод service, принадлежащий HttpServlet, также вызывает метод doOptions, когда сервлет получает OPTIONS‑запрос и метод doTrace, когда сервлет получает TRACE‑запрос. По умолчанию, реализация метода doOptions автоматически определяет какие HTTP‑опции поддерживаются и возвращает эту информацию. По умолчанию, реализация метода doTrace выдает ответ с сообщением, содержащим все заголовки, посланные в TRACE‑запросе. Эти методы, как правило, не переопределяются.

***Практическая часть***

### Сервер Jakarta Tomcat

Перейдем к практической части лабораторной работы. Для ее выполнения вам понадобится сервер Jakarta Tomcat Server, который можно скачать на web‑узле http://java.sun.com. Установите дистрибутив. Объявите переменные среды JAVA\_HOME = «<path>/jdk…» и CATALINA\_HOME =»<path>\jakarta-tomcat…» (Control Panel / System / Advanced / Environment Variables). Теперь Вы имеет установленный tomcat. Если вам надо чтобы какие-то библиотеки подключались в CLASSPATH – положите их в каталог lib. По умолчанию сервер использует порт 8080. Для запуска Tomcat используйте bin\startup.bat, для остановки – bin\shutdown.bat. Корневым каталогом для Ваших докуметов будет webapps/ROOT/

Для классов – webapps/ROOT/WEB-INF/classes/. Например, класс myServlet расположенный в этом каталоге вызывается: http://localhost:8080/servlet/myServlet. Класс myPackage.myServlet (расположенный в пакете myPackage и соответственно в каталге webapps/ROOT/WEB-INF/classes/myPackage/) вызывается: http://localhost:8080/servlet/myPackage.myServlet

***Примеры***

Приведем несколько примеров сервлетов.

 // HelloWorld.java выводит на браузер надпись HelloWorld

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class HelloWorld extends HttpServlet {

public void doGet (HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException

{

response.setContentType («text/html»);

PrintWriter out = response.getWriter();

out.println («<html>»);

out.println («<body>»);

out.println («<head>»);

out.println («<title>Hello World!</title>»);

out.println («</head>»);

out.println («<body>»);

out.println («<h1>Hello World!</h1>»);

out.println («</body>»);

out.println («</html>»);

}

}

Как вы видите, сервлет обрабатывает Get – запрос браузера.

response.setContentType («text/html»); – устанавливаем тип ответа

PrintWriter out = response.getWriter(); – открываем поток, при помощи которого происходит запись выходных данных на браузер.

Приведем пример сервлета, обрабатывающего входные параметры.

 // код HTML‑страницы, вызывающей сервлет

<form action= «RequestParamExample» method=POST>

First Name:

<input type=text size=20 name=firstname>

<br>

Last Name:

<input type=text size=20 name=lastname>

<br>

<input type=submit>

</form>

 // RequestParamExample.java получает входные параметры и выводит на браузер их значения

import java.io.\*;

import java.util.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class RequestParamExample extends HttpServlet {

public void doPost (HttpServletRequest request, HttpServletResponse res)

throws IOException, ServletException

{

Enumeration e = request.getParameterNames();

PrintWriter out = res.getWriter ();

while (e.hasMoreElements()) {

String name = (String) e.nextElement();

String value = request.getParameter(name);

out.println (name + «=» + value);

}

}

}

Здесь мы из вызывающей страницы получаем параметры firstname и lastname. В сервлете:

Enumeration e = request.getParameterNames(); – получаем имена всех параметров запроса, затем открываем поток записи н браузер, и до тех пор, пока множество имен не исчерпано, получаем имя параметра

String name = (String) e.nextElement();

Получаем, значение параметра, зная имя

String value = request.getParameter(name);

Выводим его на брузер

out.println (name + «=» + value);

Приведем пример работы с сессиями:

import java.io.\*;

import java.util.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class SessionExample extends HttpServlet {

public void doGet (HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException

{

response.setContentType («text/html»);

PrintWriter out = response.getWriter();

HttpSession session = request.getSession(true);

 // добавить атрибут в сессию

String dataName = request.getParameter («dataName»);

String dataValue = request.getParameter («dataValue»);

session.setAttribute (dataName, dataValue);

 // вывести значения атрибутов сессии

Enumeration e = session.getAttributeNames();

while (e.hasMoreElements()) {

String name = (String) e.nextElement();

String value = session.getAttribute(name).toString();

out.println (name + «=» + value);

}

}

}

Сервлет обрабатывает Get‑запрос, извлекает значение параметра

String dataName = request.getParameter («dataName»);

String dataValue = request.getParameter («dataValue»);

И добавляет его в сессию

session.setAttribute (dataName, dataValue);

Затем получаем имена всех атрибутов сессии:

Enumeration e = session.getAttributeNames();

И пишем на браузер их значения

String value = session.getAttribute(name).toString();

out.println (name + «=» + value);