Знакомство с сервлетами

JSP & Servlets 12 уровень, 0 лекция

ОТКРЫТА

1.1 interface Servlet

Сегодня начинаем новую и интересную тему — **сервлеты**. Именно добавление сервлетов в Java привело к тому, что Java сейчас — де-факто стандарт для больших серверных приложений. 80% всего корпоративного ПО в мире пишется на Java. А в Китае так и все 100%. Так что же такое сервлеты?

Сервлет — это именно то, что превращает Java-программу в веб-сервис и позволяет ей обрабатывать запросы от клиентов. А дело было так...

В 90-е годы, сразу после появления всемирной паутины, появились веб-клиенты (браузеры) и веб-сервера. **Веб-сервера** обычно просто раздавали через интернет файл, которые у них хранились: html-страницы, скрипты, картинки и т. п.

В какой-то момент все пришли к выводу, что нужно бы сделать обе стороны поумнее. В HTML-страницы добавили JavaScript, а серверам добавили плагины — специальные скрипты, которые вызывались в ответ на определенные запросы и позволяли сделать поведение сервера гибче и умнее.

Так вот **сервлет** — это такой **Java-плагин**, который встраивался в [Java веб-сервер] и позволял ему выполнять Java-код при запросе на определенные страницы. А уже этот Java-код, представленный классом, унаследованным от класса Servlet, делал то, что задумали его разработчики.

И как ты уже знаешь, самый популярный Java веб-сервер — это <u>Tomcat</u>. Названный, кстати, в честь кота Тома из мультфильма "Том и Джери".

Как же Tomcat взаимодействует с сервлетами? На самом деле этот процесс стандартизирован и называется жизненным циклом сервлета. В нем сервлет — это загружаемый объект, а веб-сервер — это контейнер сервлетов.

Если сервлет еще не загружен, то:

- 1. Класс сервлета загружается контейнером.
- 2. Контейнер создает экземпляр класса (объект) сервлета.
- 3. Контейнер вызывает метод init() у объекта сервлета. Метод вызывается только один раз.

Стандартный цикл работы — обслуживание клиентского запроса:

- Каждый запрос обрабатывается в отдельном потоке.
- Контейнер вызывает метод service() у сервлета и передает туда объекты ServletRequest и ServletResponse.
- Для завершения работы сервлета вызывается метод destroy() у объекта сервлета. Вызывается он всего один раз.

Причин завершения работы сервлета может быть очень много:

- Программист перезапускает веб-сервер, нужно корректно завершить работу всех сервлетов.
- Программист загружает новую версию сервлета, старую нужно правильно выгрузить.
- И так далее.

Запомни главное: веб-сервер и его сервлеты должны без сбоев и перезагрузки работать месяцами, обслуживая тысячи запросов в минуту. Поэтому код и загрузки, и работы, и выгрузки сервлета всегда нужно писать очень качественно.

1.2 Класс HttpServlet

Класс Servlet существует для стандартизации работы сервлета и контейнера. Непосредственно с этим классом программисты не работают. Ну или работают очень редко. Чаще всего используется класс [HttpServlet], унаследованный от Servlet'a.

У этого класса есть несколько методов, которые нам пригодятся. Ты будешь часто их использовать:

	Метод	Описание
1	<pre>init()</pre>	Вызывается один раз при загрузке сервлета
2	destroy()	Вызывается один раз при выгрузке сервлета
3	service(HttpRequest, HttpResponse)	Вызывается для каждого нового запроса к сервлету
4	<pre>doGet(HttpRequest, HttpResponse)</pre>	Вызывается для каждого нового GET-запроса к сервлету
5	<pre>doPost(HttpRequest, HttpResponse)</pre>	Вызывается для каждого нового POST-запроса к сервлету
6	doHead(HttpRequest, HttpResponse)	Вызывается для каждого нового HEAD-запроса к сервлету
7	doDelete(HttpRequest, HttpResponse)	Вызывается для каждого нового DELETE-запроса к сервлету
8	doPut(HttpRequest, HttpResponse)	Вызывается для каждого нового PUT-запроса к сервлету

Meтоды init() и destroy() унаследованы от класса Servlet. Поэтому если ты решишь переопределить их в своем сервлете, тебе нужно будет так же вызвать их реализацию из базового класса. Для этого используется команда super.имяМетода().

Пример сервлета:

```
1
     public class FirstHttpServlet extends HttpServlet {
 2
         @Override
 3
          protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletExcepti
 4
 5
              // Получаем параметр "secret" из запроса
 6
 7
             String secret = request.getParameter("secret");
 8
9
              // Кладем параметр "secret" в Http-сессию
             HttpSession session = request.getSession(true);
10
11
              session.setAttribute("secret", secret);
12
              // Печатаем HTML в качестве ответа для браузера
13
              response.setContentType("text/html;charset=UTF-8");
14
15
              PrintWriter out = response.getWriter();
16
             try {
17
                 out.println("<html>");
18
                 out.println("<head>");
                 out.println("<title>Заголовок</title>");
19
                 out.println("</head>");
20
                 out.println("<body>");
21
22
                 out.println("<h1>Пример сервлета "+ secret +"</h1>");
                 out.println("</body>");
23
                 out.println("</html>");
24
25
              } finally {
                 out.close();
26
27
             }
         }
28
29
     }
```

1.3 Метод service(HttpServletRequest, HttpServletResponse)

Если смотреть на обработку клиентского запроса с точки зрения сервлета, то дела обстоят примерно так.

Для каждого клиентского запроса контейнер (веб-сервер) создает объекты [HttpServletRequest] и [HttpServletResponse], а затем вызывает метод [service(HttpServletRequest request, HttpServletResponse response)] у соответствующего сервлета. В него передаются эти объекты, чтобы метод мог взять нужные данные из [request' а] и положить результат работы в [response].

У метода service() есть реализация по умолчанию. Если ее не переопределить, то выполняться будет именно она. Вот что он делает.

Meтод service() определяет из request'а тип HTTP-метода (GET, POST, ...) и вызывает метод соответствующий запросу.

	Метод	Описание
1	service(HttpRequest, HttpResponse)	Вызывается для каждого нового запроса к сервлету
2	<pre>doGet(HttpRequest, HttpResponse)</pre>	Вызывается для каждого нового GET-запроса к сервлету
3	<pre>doPost(HttpRequest, HttpResponse)</pre>	Вызывается для каждого нового POST-запроса к сервлету
4	<pre>doHead(HttpRequest, HttpResponse)</pre>	Вызывается для каждого нового HEAD-запроса к сервлету
5	<pre>doDelete(HttpRequest, HttpResponse)</pre>	Вызывается для каждого нового DELETE-запроса к сервлету
6	doPut(HttpRequest, HttpResponse)	Вызывается для каждого нового PUT-запроса к сервлету

В своем классе ты можешь или переопределить один метод service(), или не трогать его, но тогда переопределять методы doGet(), doPost(), ... по необходимости.

< Предыдущая лекция

Следующая лекция >



Комментарии (4) популярные новые старые

JavaCoder

Введите текст комментария

Ріхта Уровень 90 ехрегт

12 августа 2022, 04:39

По факту названия Тотсат в честь персонажа мультсериала не нашел никакой информации.

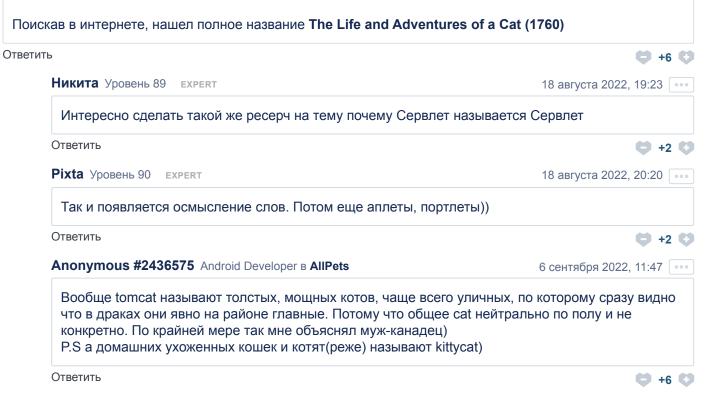
Однако, исходя официального вики tomcat

Как Tomcat получил свое название?

Джеймс Дункан Дэвидсон придумал **"Tomcat"**, поскольку животное представляло собой нечто, что могло позаботиться о себе и постоять за себя. Вот так он придумал это имя.

В других источниках говорится, что было оказано влияние трудов выпуска O'Reilly, имеющие на своих обложках животных. Он был этим так воодушевлен, что хотел тоже выпустить похожее.

<u>wikipedia</u> трактует, что термин **"Tomcat"** происходит от литературного персонажа XVIII века, кота по имени Том.



ОБУЧЕНИЕ СООБЩЕСТВО КОМПАНИЯ Курсы программирования Пользователи Онас Статьи Kypc Java Контакты Помощь по задачам Отзывы Форум Подписки Чат **FAQ** Задачи-игры Истории успеха Поддержка Активности



RUSH

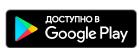
JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ







"Программистами не рождаются" © 2023 JavaRush