

# Структура web.xml

JSP & Servlets  
11 уровень, 6 лекция

ОТКРЫТА

## 7.1 Общая схема web.xml

**Файл web.xml** хранит информацию о конфигурации приложения. Он не является обязательной его частью, однако очень широко используется для настройки конфигурации веб-приложения.

Этот файл должен располагаться в папке **WEB-INF**. При запуске Tomcat считывает его содержимое и использует записанную в нем конфигурацию. Если же файл содержит ошибки, то и Tomcat отображает ошибку.

Пример web.xml:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
4      http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" version="4.0">
5
6      <servlet>
7          <servlet-name>HelloWorld</servlet-name>
8          <servlet-class>HelloServlet</servlet-class>
9      </servlet>
10
11     <servlet-mapping>
12         <servlet-name>HelloWorld</servlet-name>
13         <url-pattern>/welcome</url-pattern>
14     </servlet-mapping>
15
16     <welcome-file-list>
17         <welcome-file>index.html</welcome-file>
18     </welcome-file-list>
19
20 </web-app>
```

Зеленым цветом тут записан маппинг имя сервлета "HelloWorld" и класса сервлета "HelloServlet". Синим цветом записан маппинг имя сервлета "HelloWorld" и куска URL "http://localhost/welcome". Таким образом тут написано, что при обращении к пути /welcome нужно вызвать сервлет HelloServlet.class.

Красным цветом указан файл, который нужно отдать по запросу http://localhost/ — это так называемая welcome page. Если пользователь просто вобьет в браузере имя, соответствующее корню нашего веб-приложения, то ему отдается содержимое файла index.html.

## 7.2 servlet, servlet-mapping

Один сервлет может обслуживать запросы по разным урлам, поэтому в web.xml сервлет и его маппинг на урлы записываются отдельно. Сначала описываем сервлеты, давая каждому уникальное строковое имя, а затем уже указываем, как каждый сервлет мапится на какой url.

Пример web.xml:

```
1  <web-app>
2
3  <servlet>
4      <servlet-name>remoting</servlet-name>
5      <servlet-class>com.javarush.RemotingServlet</servlet-class>
6      <load-on-startup>1</load-on-startup>
7  </servlet>
8
9  <servlet-mapping>
10     <servlet-name>remoting</servlet-name>
11     <url-pattern>/remoting/*</url-pattern>
12 </servlet-mapping>
13
14 <servlet>
15     <servlet-name>restapi</servlet-name>
16     <servlet-class>com.javarush.RestApiServlet</servlet-class>
17     <load-on-startup>2</load-on-startup>
18 </servlet>
19
20 <servlet-mapping>
21     <servlet-name>restapi</servlet-name>
22     <url-pattern>/api/*</url-pattern>
23 </servlet-mapping>
24
25 </web-app>
```

В этом примере объявлено два сервлета, и каждый замаплен на свой шаблон url. Сервлет `RemotingServlet` обслуживает все запросы, которые идут на `/remoting/*`. Сервлет `RestApiServlet` обслуживает все запросы, которые идут на `/api/*`. Так же у сервлетов прописан порядок из загрузки — параметр `load-on-startup`.

### 7.3 Параметры сервлета

С помощью `web.xml` сервлету при его инициализации можно передать параметры, они будут доступны через интерфейс `ServletConfig`. Также можно задать параметры всему веб-приложению, они будут доступны через интерфейс `ServletContext`.

Пример `web.xml`:

```
1  <web-app>
2      <context-param>
3          <description>Server production mode</description>
4          <param-name>productionMode</param-name>
5          <param-value>false</param-value>
6      </context-param>
7
8      <context-param>
9          <param-name>appPropertiesConfig</param-name>
10         <param-value>
11             classpath:local-app.properties
12             classpath:web-app.properties
13         </param-value>
14     </context-param>
15
16     <servlet>
17         <servlet-name>mainervlet</servlet-name>
18         <servlet-class>com.javarush.ApplicationServlet</servlet-class>
19         <init-param>
```

```
20         <param-name>application</param-name>
21         <param-value>com.javarush.App</param-value>
22     </init-param>
23     <init-param>
24         <param-name>widgetset</param-name>
25         <param-value>com.javarush.WidgetSet</param-value>
26     </init-param>
27     <init-param>
28         <param-name>ui</param-name>
29         <param-value>com.javarush.AppUI</param-value>
30     </init-param>
31 </servlet>
32 </web-app>
```

Зеленым цветом выделен код, где мы задаем параметры для `ServletContext`. Их там два:

- `productionMode` со значением `false`
- `appPropertiesConfig` с массивом из двух строк:
  - `classpath:local-app.properties`
  - `classpath:web-app.properties`

Синим цветом указаны параметры для сервлета `ApplicationServlet`, они будут доступны ему через `ServletConfig`:

- `application` со значением `com.javarush.App`
- `widgetset` со значением `com.javarush.WidgetSet`
- `ui` со значением `com.javarush.AppUI`

## 7.4 filter, filter-mapping

Веб-приложение может также содержать специальные `служебные сервлеты – фильтры`. Они выполняют различные служебные задачи: перенаправляют вызовы, проверяют авторизацию и т. д.

Пример web.xml:

```
1  <web-app>
2
3      <servlet>
4          <servlet-name>remoting</servlet-name>
5          <servlet-class>RemotingServlet</servlet-class>
6          <load-on-startup>1</load-on-startup>
7      </servlet>
8
9      <servlet-mapping>
10         <servlet-name>remoting </servlet-name>
11         <url-pattern>/remoting/*</url-pattern>
12     </servlet-mapping>
13
14     <filter>
15         <filter-name>total_filter</filter-name>
16         <filter-class>com.javrush.TotalFilter</filter-class>
17     </filter>
18
19     <filter-mapping>
20         <filter-name>total_filter</filter-name>
21         <url-pattern>/*</url-pattern>
22     </filter-mapping>
23
24 </web-app>
```

Прежде чем запрос попадет в сервлет `RemotingServlet`, он будет обработан фильтром `TotalFiler`. Этот фильтр настроен так, чтобы **перехватывать все запросы**, которые идут к нашему веб-приложению. Об это однозначно намекает шаблон урлов, на которые он замаплен: `/*`.

Больше про сервлеты и фильтры ты прочитаешь в следующих лекциях.

−

+9

+

Комментарии (3)

популярные

новые

старые

JavaCoder

Введите текст комментария

**Bek**

Уровень 5

1 декабря 2022, 08:39

⋮

Добрый день! Я создал war file с методами. Подскажите пожалуйста, как написать программу так что бы вызвать метод? Уже неделя прошла, не могу разобраться. Для примера есть "pos.war", внутри pos.war есть метод "call". Когда в браузере набираю localhost/8080/pos/call ошибка входит.

Ответить

−

0

+

⬤

**Никита**

Уровень 89

EXPERT

18 августа 2022, 13:27

⋮

Хоть бы объяснили заче эти параметры передавать и что они означют..

Ответить

−

+12

+

**Александр**

Уровень 50

11 июля 2022, 13:41

⋮

Также можно задать параметры всему веб-приложению, они будут доступны через класс ContextConfig(неверно), д/б указан ServletContext. Исправьте.

Ответить

−

+2

+



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 

