Mini Servlets

Abschlussprojekt Internetprogrammierung

Stefan Wehr

17. Juli 2006

1 Einleitung

Zum Abschluss der Vorlesung *Internetprogrammierung* im Sommersemester 2006 sollen die erworbenen Kenntnisse in einem Abschlussprojekt vertieft werden. Ziel des Projekts ist die Erstellung eines einfachen Servletcontainers. Als Ausgangspunkt wird auf der Vorlesungsseite [2] ein einfacher Webserver bereitgestellt, der um das Parsen von Konfigurationsdateien sowie um das dynamische Laden und Ausführen von Servlets erweitert werden muss. Außerdem müssen aus dem HTTP Request die Parameter extrahiert und aufbereitet werden. Ein Sessionmechanismus ist ebenfalls Bestandteil des Containers.

Zunächst wird in Abschnitt 2 kurz erklärt, wie der Servletcontainer allgemein funktioniert. In Abschnitt 3 wird dann erklärt, welche Aspekte des HTTP Protokolls unterstützt werden sollen. Abschnitt 4 stellt eine einfache API für Servlets vor, die an die originale Servlet API [4] angelehnt ist. Anschließend beschreibt Abschnitt 5 die Konfiguration des Servletcontainers. In Abschnitt 6 werden schließlich die Modalitäten zur Abgabe des Projekts erklärt.

2 Allgemeine Funktionsweise

Der Servletcontainer ist prinzipiell ein Webserver mit erweiterter Funktionalität. Beim Eintreffen eines Request wird zunächst entschieden, ob der Request sich auf eine statische Resource bezieht, oder ob er von einem Servlet behandelt werden soll (siehe auch Abschnitt 5). Handelt es sich um eine statische Resource, dann wird im sogenannten `document-root` Verzeichnis nach der entsprechenden Datei gesucht und der Inhalt an den Client ausgeliefert.

Falls der Request allerdings von einem Servlet bedient werden soll, dann muss der Servletcontainer die entsprechende Klasse zunächst dynamisch laden. Der Container sucht solche Klassen im `class-root` Verzeichnis. Ist die Klasse geladen, ruft der Container eine auf die Request-Art passende Methode auf, welche den für den Client bestimmten Inhalt generiert.

Zur Aufgabe des Servletcontainers gehört auch die Bereitstellung eines Sitzungsmechanismus. Damit können über mehrere Interaktionsschritte hinweg Daten für bestimmte

Clients persistent gehalten werden. Die Identifikations eines Clients geschieht über eine Session-ID. Diese ID wird in der URL mitgeliefert. Der Container kümmert sich dabei um die korrekte Einbettung der Session-ID in die URL. So wird etwa aus der URL `http://localhost:8080/HelloWorld?name=stefan` nach Einbettung der Session-ID `http://localhost:8080/HelloWorld;msessionid=746sfsdfa0943lksdfa23?name=stefan`.

3 Unterstützung des HTTP Protokolls

Der Servletcontainer muss die HTTP Methoden `GET` und `POST` unterstützen [5]. Dabei müssen die Parameter des Requests an das Servlet weitergereicht werden. Bei der `GET` Methoden werden Parameter URL-kodiert (`application/x-www-form-urlencoded`). Bei `POST` muss neben diesem Encoding auch `multipart/form-data` unterstützt werden, damit Fileuploads möglich sind.

Dem Benutzer der Servlet-API stehen keine Methoden zur Verfügung, um HTTP Header zu lesen oder zu schreiben. Welche HTTP Header vom Servletcontainer benutzt werden, ist also implementierungsspezifisch. Cookies müssen nicht implementiert werden, da die Session-ID in der URL kodiert wird (siehe Abschnitt 2).

4 Servlet API

Auf der Vorlesungseite [2] stehen im Package `de.uni_freiburg.informatik.proglang.mservlets.interfaces` eine Reihe von Interfaces zur Verfügung, welche die Schnittstelle zwischen dem Servletcontainer und einem Servlet definieren. Die Interfaces sind dabei an entsprechende Interfaces oder Klassen der originalen Servlet API [4] angelehnt. An dieser Stelle seien nur die Signaturen der Interfaces erwähnt. Detaillierte Dokumentation findet sich im Quellcode oder in der original Servlet API.

Jede Klasse, die im Servletcontainer als Servlet läuft, muss das Interface `MServlet` implementieren.

```
public interface MServlet {
  public void doGet(MServletRequest req, MServletResponse res);
  public void doPost(MServletRequest req, MServletResponse res);
}
```

Das Interface `MServletRequest` stellt Informationen über den Client der Anfrage zur Verfügung.

```
public interface MServletRequest {
    public String getServletPath();
    public MSession getSession();
    public RequestParameter getParameter(String name);
    public RequestParameter[] getParameterValues(String name);
    public Iterator<RequestParameter> getParameterNames();
```

```
    public String encodeURL(String url);
}
```

Die beiden Interfaces `RequestParameter` und `RequestFileParameter` repräsentieren Formulareingabefelder. Letzteres Interface stellt zusätzliche Methoden bereit, die für Fileuploads verwendet werden.

```
public interface RequestParameter {
        public String getName();
        public String getValue();
}

public interface RequestFileParameter extends RequestParameter {
        public String getContentType();
        public InputStream getInputStream();
}
```

Über das Interface `MServletResponse` kann ein Servlet eine Antwort an den Client schicken.

```
public interface MServletResponse {
        public void sendError(int status, String msg);
        public void setContentType(String type);
        public PrintWriter getWriter() throws IOException;
        public PrintStream getOutputStream() throws IOException;
}
```

Das Interface `MSession` stellt die Schnittstelle zum Sitzungsmechanismus zur Verfügung.

```
public interface MSession {
        public Object getAttribute(String name);
        public void setAttribute(String name, Object value);
        public Iterator<String> getAttributeNames();
        public boolean isNew();
}
```

Das Interface `HttpConstants` definiert allgemeine HTTP Konstanten.

```
public interface HttpConstants {
    public static final int HTTP_OK = 200;
    public static final int HTTP_BAD_REQUEST = 400;
    public static final int HTTP_FORBIDDEN = 403;
    public static final int HTTP_NOT_FOUND = 404;
    /* ... */
}
```

5 Konfiguration

Die Konfiguration des Servletcontainers wird in einer XML Datei gespeichert. Der Dateiname wird dem Server als Kommandozeilenargument übergeben. Im einzelnen gibt es die folgenden Konfigurationsvariablen:

timout Der Timeout in Millisekunden (zwischen 1000 und 10000).

workers Die Anzahl der Worker-Threads (zwischen 5 und 25).

logfile Der Name der Logdatei.

document-root Das Verzeichnis unter dem statische Inhalte wie Bilder etc. abgelegt werden.

class-root Das Verzeichnis unter dem Servletklassen abgelegt werden.

Neben Konfigurationsvariablen gibt es noch sogenannte `servlet-mapping` Elemente in der Konfigurationsdatei. Ein solches Element enthält ein Element `url`, welches ein URL Pattern angibt. Wenn das Pattern passt, dann wird die im Element `class` spezifizierte Servletklasse verwendet, um den Request zu bedienen.

Beispielkonfigurationsdatei:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<mini-servlets-config>
  <timeout>2000</timeout>
  <workers>10</workers>
  <logfile>/home/wehr/mini-servlets/log</logfile>
  <document-root>/home/wehr/mini-servlets/docroot</document-root>
  <class-root>/home/wehr/mini-servlets/classroot</class-root>
  <servlet-mappings>
    <servlet-mapping>
      <url>*.do</url>
      <class>de.uni_freiburg.informatik.proglang.ControllerServlet</class>
    </servlet-mapping>
    <servlet-mapping>
      <url>/HelloWorld</url>
      <class>de.uni_freiburg.informatik.proglang.HelloWorldServlet</class>
    </servlet-mapping>
  </servlet-mappings>
</mini-servlets-config>
```

6 Abgabemodalitäten

Die Frist zur Abgabe des Projekts ist Freitag, der 11. August 2006. Die Abgabe soll als .zip oder .tar.gz Archiv erfolgen. Innerhalb des Archivs muss sich ein Shell-Skript

`start-server.sh` befinden, welches den Server startet. Insbesonder soll kein manuelles Setzen des `CLASSPATH` o.ä. nötig sein. Bewertet wird das Projekt durch Deployen und Verwenden eines vom Betreuer implementierten Test-Servlets.

Literatur

[1] Commons FileUpload, 2006. `http://jakarta.apache.org/commons/fileupload/`.

[2] Homepage zur Vorlesung Internetprogrammierung, 2006. `http://proglang.informatik.uni-freiburg.de/teaching/inetprog/2006/`.

[3] Jakarta ORO, 2006. `http://jakarta.apache.org/oro/`.

[4] Servlet API, 2006. `http://java.sun.com/products/servlet/2.5/docs/servlet-2_5-mr2/index.html`.

[5] Network Working Group. RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1, 1999. `http://www.faqs.org/rfcs/rfc2616.html`.