17장 서블릿 기초

17.1 서블릿 기초

- 서블릿은 JSP 표준이 나오기 전에 만들어진 표준으로 자바로 웹 어플리케이션을 개발할 수 있도록 하기 위해 만들어졌다. 일반적인 서블릿의 개발 과정은 다음과 같다.
  1. 서블릿 규약에 따라 자바 코드를 작성한다.
  2. 자바 코드를 컴파일해서 클래스 파일을 생성한다.
  3. 클래스 파일을 /WEB-INF/classes 폴더에 패키지에 알맞게 위치시킨다.
  4. web.xml 파일에 서블릿 클래스를 설정한다.
  5. 톰캣 등의 컨테이너를 실행한다.
  6. 웹 브라우저에서 확인한다.

17.2 예제 프로젝트 생성

- 이클립스에서 File > Import > Existing Projects into Workspace 메뉴를 이용하여 프로젝트(chap17)를 임포트한다.

17.2.1 서블릿 구현

[chap17/src/example/NowServlet.java]

```
package example;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class NowServlet extends HttpServlet {

        @Override
        protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
                response.setContentType("text/html; charset=utf-8");

                PrintWriter out = response.getWriter();
                out.println("<html>");
                out.println("<head><title>현재시간</title></head>");
                out.println("<body>");
                out.println("현재 시간은");
                out.println(new Date());
                out.println("입니다.");
                out.println("</body></html>");
        }

}
```

17.2.2 web.xml로 매핑하기

- 서블릿 클래스를 생성했다면 그다음으로 할 작업은 WEB-INF 폴더의 web.xml 파일에 서블릿 클래스를 등록하는 것이다.

[chap17/WebContent/WEB-INF/web.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns="http://xmlns.jcp.org/xml/ns/javaee"
          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
          version="3.1">

          <servlet>
                    <servlet-name>now</servlet-name>
                    <servlet-class>example.NowServlet</servlet-class>
          </servlet>

          <servlet-mapping>
                    <servlet-name>now</servlet-name>
                    <url-pattern>/now</url-pattern>
          </servlet-mapping>

          <servlet>
                    <servlet-name>DBCPInit2</servlet-name>
                    <servlet-class>jdbc.DBCPInit2</servlet-class>
                    <init-param>
                              <param-name>jdbcdriver</param-name>
                              <param-value>com.mysql.jdbc.Driver</param-value>
                    </init-param>
                    <init-param>
                              <param-name>jdbcUrl</param-name>
                              <param-value>
                      jdbc:mysql://localhost:3306/chap14?characterEncoding=utf8
                  </param-value>
                    </init-param>
                    <init-param>
                              <param-name>dbUser</param-name>
                              <param-value>jspexam</param-value>
                    </init-param>
                    <init-param>
                              <param-name>dbPass</param-name>
                              <param-value>jsppw</param-value>
                    </init-param>
                    <init-param>
                              <param-name>poolName</param-name>
                              <param-value>chap14</param-value>
                    </init-param>
                    <load-on-startup>1</load-on-startup>
          </servlet>

</web-app>
```

17.2.3 애노테이션으로 매핑하기

- 서블릿 3.0 버전부터는 @WebServlet 애노테이션을 사용하면 web.xml 파일에 따로 등록하지 않아도 서블릿으로 등록된다.

[chap17/src/example/HelloServlet.java]

```
package example;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = "/hello")
public class HelloServlet extends HttpServlet {

        @Override
        protected void doGet(HttpServletRequest request,
                HttpServletResponse response) throws ServletException, IOException {
                request.setCharacterEncoding("utf-8");
                response.setContentType("text/html; charset=utf-8");

                PrintWriter out = response.getWriter();
                out.println("<html>");
                out.println("<head><title>인사</title></head>");
                out.println("<body>");
                out.println("안녕하세요, ");
                out.println(request.getParameter("name"));
                out.println("님");
                out.println("</body></html>");
        }

}
```

17.2.4 HTTP 각 방식별 구현 메서드

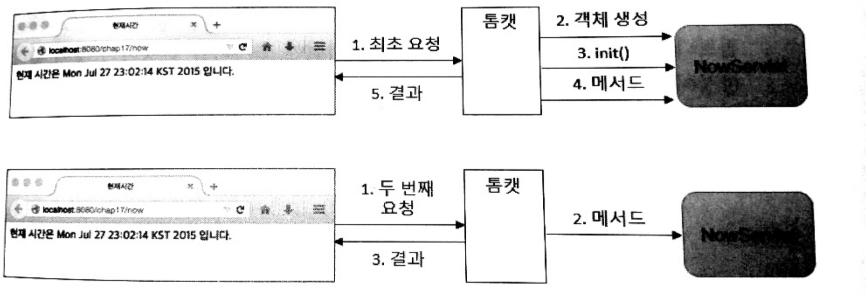
- GET 방식은 doGet() 메서드를 이용해서 처리하고, POST 방식의 경우 doPost() 메서드를 이용해서 처리하도록 정의하고 있다.

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
   ...  // GET 방식에 대한 처리
}

@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
   ...  // Post 방식에 대한 처리
}
```

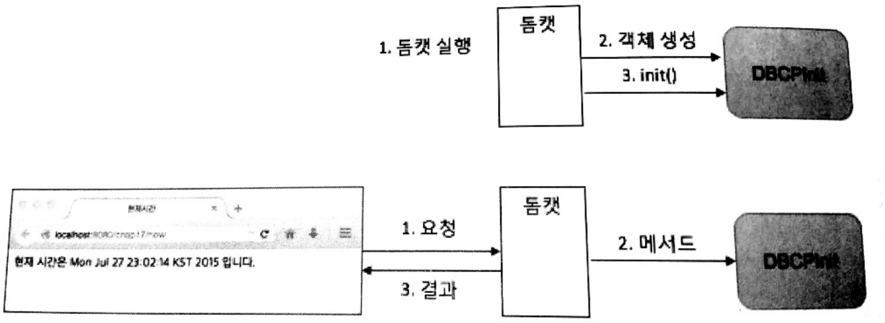
17.2.5 서블릿 로딩과 초기화

- 최초 요청이 발생하면 서블릿 객체를 생성



현재시간
localhost:8080/chap17/now
현재 시간은 Mon Jul 27 23:02:14 KST 2015 입니다.
1. 최초 요청
5. 결과
톰캣
2. 객체 생성
3. init()
4. 메서드
NowServlet
현재시간
localhost:8080/chap17/now
현재 시간은 Mon Jul 27 23:02:14 KST 2015 입니다.
1. 두 번째 요청
3. 결과
톰캣
2. 메서드
NowServlet

■ <load-on-startup> 태그를 사용하면 서블릿을 미리 초기화할 수 있다. 태그의 값은 로딩 순서를 의미한다.



■ @WebServlet 태그를 사용하는 경우에는 loadOnStartup 속성을 이용해서 로딩 값을 지정한다.

```
// 서블릿
public class DBCPInit2 extends HttpServlet {
          @Override
          public void init() throws ServletException {
                    loadJDBCDriver();
                    initConnectionPool();
          }
          ...
}

// web.xml
<servlet>
          <servlet-name>DBCPInit2</servlet-name>
          <servlet-class>jdbc.DBCPInit2</servlet-class>
          <load-on-startup>1</load-on-startup>
</servlet>

// @WebServlet 태그
@WebServlet(urlPatterns="/hello", loadOnStartup=1)
public class InitServlet extends HttpServlet {
   ...
}
```

17.2.6 초기화 파라미터

■ 서블릿은 web.xml의 <init-param> 태그를 이용해서 서블릿을 초기화할 때 필요한 값을 전달하는 방법을 제공하고 있다.

(1) DBCPInt2.java

[chap17/src/jdbc/DBCPInit2.java]

```
package jdbc;

import java.sql.DriverManager;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;

import org.apache.commons.dbcp2.ConnectionFactory;
import org.apache.commons.dbcp2.DriverManagerConnectionFactory;
import org.apache.commons.dbcp2.PoolableConnection;
import org.apache.commons.dbcp2.PoolableConnectionFactory;
import org.apache.commons.dbcp2.PoolingDriver;
import org.apache.commons.pool2.impl.GenericObjectPool;
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;

public class DBCPInit2 extends HttpServlet {

    @Override
    public void init() throws ServletException {
        loadJDBCDriver();
        initConnectionPool();
    }


    private void loadJDBCDriver() {
        String driverClass = getInitParameter("jdbcdriver");
        try {
            Class.forName(driverClass);
        } catch (ClassNotFoundException ex) {
            throw new RuntimeException("fail to load JDBC Driver", ex);
        }
    }

    private void initConnectionPool() {
        try {
            String jdbcUrl = getInitParameter("jdbcUrl");
            String username = getInitParameter("dbUser");
            String pw = getInitParameter("dbPass");

            ConnectionFactory connFactory =
                    new  DriverManagerConnectionFactory(jdbcUrl,  username,
pw);

            PoolableConnectionFactory poolableConnFactory =
                    new PoolableConnectionFactory(connFactory, null);
            poolableConnFactory.setValidationQuery("select 1");

            GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
            poolConfig.setTimeBetweenEvictionRunsMillis(1000L * 60L * 5L);
            poolConfig.setTestWhileIdle(true);
            poolConfig.setMinIdle(4);
            poolConfig.setMaxTotal(50);

            GenericObjectPool<PoolableConnection> connectionPool =
                    new           GenericObjectPool<>(poolableConnFactory,
poolConfig);
            poolableConnFactory.setPool(connectionPool);

            Class.forName("org.apache.commons.dbcp2.PoolingDriver");
            PoolingDriver driver =
                    (PoolingDriver)
DriverManager.getDriver("jdbc:apache:commons:dbcp:");
            String poolName = getInitParameter("poolName");
            driver.registerPool(poolName, connectionPool);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

(2) web.xml

[chap17/WebContent/WEB-INF/web.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">

    <servlet>
        <servlet-name>now</servlet-name>
        <servlet-class>example.NowServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>now</servlet-name>
        <url-pattern>/now</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>DBCPInit2</servlet-name>
        <servlet-class>jdbc.DBCPInit2</servlet-class>
        <init-param>
            <param-name>jdbcdriver</param-name>
            <param-value>com.mysql.jdbc.Driver</param-value>
        </init-param>
        <init-param>
            <param-name>jdbcUrl</param-name>
            <param-value>
         jdbc:mysql://localhost:3306/chap14?characterEncoding=utf8
       </param-value>
        </init-param>
        <init-param>
            <param-name>dbUser</param-name>
            <param-value>jspexam</param-value>
        </init-param>
        <init-param>
            <param-name>dbPass</param-name>
            <param-value>jsppw</param-value>
        </init-param>
        <init-param>
            <param-name>poolName</param-name>
            <param-value>chap14</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

</web-app>
```