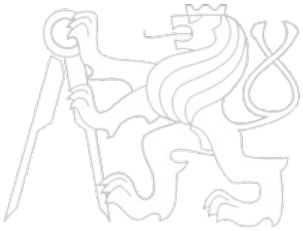
Middleware and Web Services

Lecture 3: Introduction to Application Server

doc. Ing. Tomáš Vitvar, Ph.D.
tomas@vitvar.com • @TomasVitvar • http://vitvar.com



Czech Technical University in Prague
Faculty of Information Technologies • Software and Web Engineering • http://vitvar.com/courses/mdw

Modified: Sun Oct 29 2017, 22:57:53
Humla v0.3

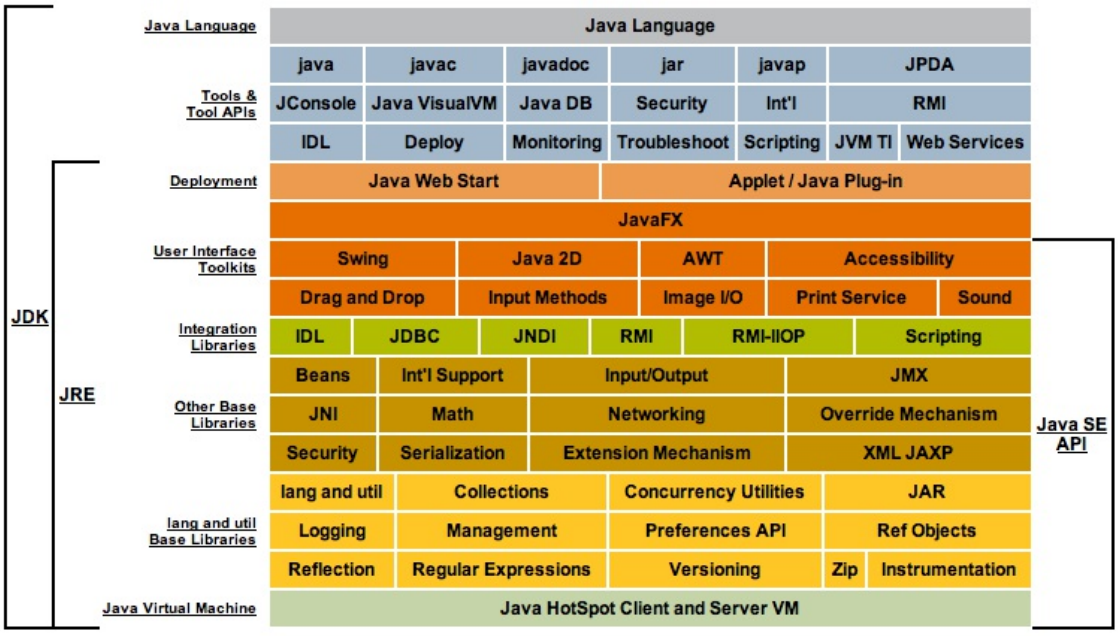
Overview

- Architecture
- I/O Communication
- Servlet Technology

Application Server Overview

- An environment that runs an application logic
  - *A client communicates with the server using an application protocol*
- Application Server
  - *A modular environment*
    - *→ provides technology to realize enterprise systems*
    - *→ JEE containers – Java technology for AS components*
    - *→ Supports a variety of objects such as Servlets, JPSs, JMS*
  - *Provides services such as naming and directory, performance, failover*
  - *Provides Web server capabilities*
  - *Can be a single server or multiple servers*
- Web Tier – HTTP Server
  - *Web Server supports HTTP only*
  - *HTTP request/response, security, proxy, caching*

Standard Java Technology Stack



Application Server Layers

Drawing not available

- Features
  - *AS instance appears as a single process in the OS*
    - *→ you can use standard OS commands to investigate its operation*
    - *→ AS listens on a single or multipe IPs (VIPs) and a tcp port*
  - *AS is a Java process*
    - *→ you can use Java tools to investigate its operation*
    - *→ Garbage collector stats, thread dumps, memory allocations, etc.*

Example Weblogic Infrastructure

Drawing not available

Terminology

- Domain
  - *A group of servers with specific configuration of applications and objects*
- Administration Server
  - *An instance of application server that manages the domain*
- Managed Server
  - *An instance of application server running instances of applications and objects*
- Cluster
  - *A group of managed servers; they contain the same copy of applications and objects*
- Machine
  - *A physical machine and OS running one or more servers (Admin or Managed)*
- Node Manager
  - *A process that provides an access to admin and managed servers on the machine*
- Load Balancer
  - *A network element that distributes client requests to managed servers based on a specific algorithm*

Console Example – Weblogic Server

Drawing not available

Application Server from the OS View

- Process ID, command line arguments

```
$ ps ax | grep WLS_SOA
 1820 ?        Sl   289:15 /opt/oracle/jrockit/bin/java -jrockit
 -Xms768m -Xmx1536m -Dweblogic.Name=WLS_SOA1 -Djava.security.policy=
 /opt/oracle/11g/fmw/wlserver_10.3/server/lib/weblogic.policy
 -Dweblogic.ProductionModeEnabled=true
 ...
```

- Open files by the process

```
$ ll -l /proc/1820/fd
 lr-x------ 1 oracle oinstall 64 Oct 12 16:53 0 -> /dev/null
 l-wx------ 1 oracle oinstall 64 Oct 12 16:53 1 -> /opt/oracle/11g/domains/soa_domain/se
 lr-x------ 1 oracle oinstall 64 Oct 12 16:53 10 -> /opt/oracle/11g/fmw/oracle_common/mo
 lr-x------ 1 oracle oinstall 64 Oct 12 16:53 100 -> /opt/oracle/11g/fmw/modules/com.bea
 ...
```

- Open sockets by the process

```
$ netstat -anp | grep 1820
tcp  0 0 192.168.94.52:8001      0.0.0.0:*              LISTEN      1820/java
tcp  0 0 192.168.94.10:8088      0.0.0.0:*              LISTEN      1820/java
tcp  0 0 192.168.94.10:39763     192.168.94.20:33001    ESTABLISHED 1820/java
tcp  0 0 192.168.94.52:8001      192.168.94.20:59589    ESTABLISHED 1820/java
tcp  0 0 192.168.94.10:33498     192.168.94.20:33001    ESTABLISHED 1820/java
tcp  0 0 192.168.94.10:33504     192.168.94.20:33001    ESTABLISHED 1820/java
 ...
```


Application Server from the JVM View

- Thread dumps
  - *All threads that the application server uses, a snapshot on all the threads*
  - *Prints stack trace of currently run threads*

```
$ jrockit 1820 print_threads
```

- Command line arguments
  - *Prints all command line arguments of the JVM process*
    - → *Memory settings, log file locations, etc.*

```
$ jrockit 1820 command_line
```

- Java flight recordings
  - *Recordings of the JVM process in time (usually 5 minutes)*
  - *Shows memory usages, garbage collections phases, threads statuses, etc.*

Overview

- Architecture
- I/O Communication
- Servlet Technology

Blocking I/O Model

- The server creates a thread for every connection
  - *For example, 1K connections = 1K threads, big overhead*

Drawing not available

- Characteristics
  - *the thread is reserved for the connection*
  - *When processing of the request requires other interactions with DB/FS or network communication is slow*
    → scales very bad as the thread's execution is "blocked"

Non-Blocking I/O Model

- Connections maintained by the OS, not the Web app
  - *The Web app registers events, OS triggers events when occur*

Drawing not available

- Characteristics
  - *Event examples: new connection, read, write, closed*
  - *The app may create working threads, but controls the number!*
    → much less number of working threads as opposed to blocking I/O

Handling Requests in Weblogic

Drawing not available

- **Muxer** – component that handles communication via network sockets.
- **Request queue** – queue of requests to be processed.
- **Self-tunning thread pool** – a pool of threads in various states.
- **Work manager** – a configuration of **maximum threads** and a **capacity** that can be used to handle requests for a specific application/service.

Overview

- Architecture
- I/O Communication
- Servlet Technology

Overview

- Technology to extend application server functionalities
  - *A Java class that can respond to any type of requests*
    - → *A servlet defines an interface for a specific protocol*
    - → *Your application implements the servlet's interface*
- Commonly used to respond to HTTP requests
  - *A basis for an application running on an application server*
  - *HTTP Servlet Java classes*
    - → `HttpServlet` – *provides HTTP protocol interface*
    - → `HttpServletRequest` – *represents HTTP request*
    - → `HttpServletResponse` – *represents HTTP response*

Directory Structure

- Your application
  - *collection of documents and libraries your application requires*
  - *packaged in* `war` *or* `ear` *archive*
    - → *JAR that includes not only java classes but also additional resources such as* `.xml`, `.html`, `.js`, `.css`, `.jpg` *files.*
- Content of `war` package

```
# web archive root
war
|   # directories and documents accessible through the app root /
|   # such as img, css, js, ...
|-- (public-directory | public-document)*
|   # directories and documents internal to your application
|-- WEB-INF
        |-- (private-directory | private-document)*
        |   # compiled java classes of your application
        |-- classes
        |   # all java libraries your application requires
        |-- lib
        |   # configuration of your application
        |-- web.xml
        |-- # other platform-specific configurations
            # such as app-engineweb.xml for GAE
```


Configuration in web.xml

- `web.xml` defines configuration for
  - *list of servlets, mapping of servlets to URL paths, welcome files, filters, EJB references, authentication mechanism, etc.*
  - *basic configuration example:*

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <web-app
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns="http://java.sun.com/xml/ns/javaee">
5
6      <servlet>
7          <servlet-name>main</servlet-name>
8          <servlet-class>com.vitvar.mdw.main</servlet-class>
9      </servlet>
10
11     <servlet-mapping>
12         <servlet-name>main</servlet-name>
13         <url-pattern>/</url-pattern>
14     </servlet-mapping>
15
16     <welcome-file-list>
17         <welcome-file>index.jsp</welcome-file>
18     </welcome-file-list>
19 </web-app>
```

Handling HTTP Requests

- HTTP Servlets
  - *Servlet is a class that extends capabilities of application servers via a request-response programming model*
  - *HTTP servlets are classes that extend* `HTTPServlet` *abstract class*
  - *Example:*

```
1  package com.vitvar.mdw;
2
3  import javax.servlet.http.HttpServlet;
4  import javax.servlet.http.HttpServletRequest;
5  import javax.servlet.http.HttpServletResponse;
6
7  public class Main extends HttpServlet {
8      public doGet(HttpServletRequest request, HttpServletResponse response) {
9          // GET method implementation here
10     }
11
12     public doPost(HttpServletRequest request, HttpServletResponse response) {
13         // POST method implementation here
14     }
15
16     // other methods such as doPost, doDelete, doOptions
17 }
```


Support for Sessions

- `HttpSession` interface
  - *Allows to store session data in the memory*
  - *Java API for HTTP State Management*
    - → *Hides details from developers*

```
1  // method doGet in a servlet
2  public doGet(HttpServletRequest request, HttpServletResponse response) {
3      // access the session object through the request
4      HttpSession session = request.getSession();
5
6      // unique identification of the session, the value used for the cookie
7      String id = session.getId();
8
9      // get the value of the attribute
10     Object value = session.getAttribute("data");
11
12     // set the value of the attribute
13     session.setAttribute("data", new String("some data"));
14
15     // this will set a max-age of the session cookie
16     session.setMaxInactiveInterval(3600);
17 }
```