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

---

Overview

- **Architecture**
  - *I/O Communication Models*
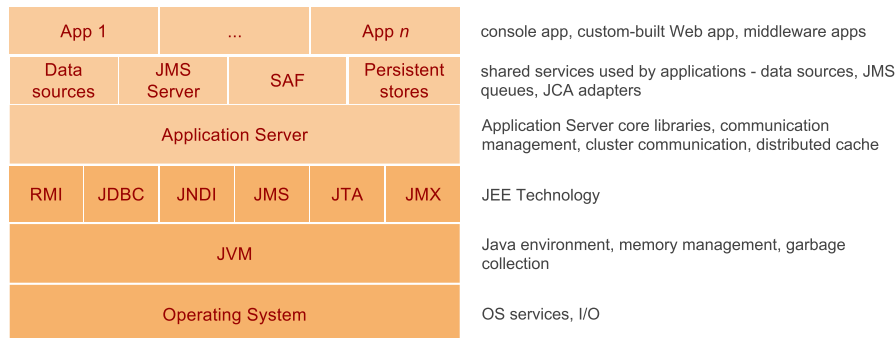- Servlet Technology

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*

- Communication models
  - *Blocking I/O (also called synchronous I/O)*
  - *Non-blocking I/O (also called asynchronous I/O)*

Java Enterprise Edition – JEE

- Java Enterprise Edition (JEE)
  - *Collection of technologies for server-side programming*
  - *Programming of application components*
  - *Main technologies*
    - → *Servlet technology and Java Server Pages (JSP)*
    - → *Remote Method Invocation (RMI)*
    - → *Java Database Connectivity Services (JDBC)*
    - → *Java Messaging System (JMS)*

- Basis for many application servers such as
  - *Oracle WebLogic*
  - *Google AppEngine (Java)*
  - *JBoss*
  - *GlassFish*
  - *IBM WebSphere*

Application Server Layers

App 1	...	App *n*	console app, custom-built Web app, middleware apps

Data sources	JMS Server	SAF	Persistent stores	shared services used by applications - data sources, JMS queues, JCA adapters

Application Server	Application Server core libraries, communication management, cluster communication, distributed cache

RMI	JDBC	JNDI	JMS	JTA	JMX	JEE Technology

JVM	Java environment, memory management, garbage collection

| Operating System	OS services, I/O

- Features
  - *AS 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.*

Overview

- Architecture
 - *I/O Communication Models*
- Servlet Technology

Blocking I/O Model

- ## The server creates a thread for every connection
  - *For example, 1K connections = 1K threads, big overhead*
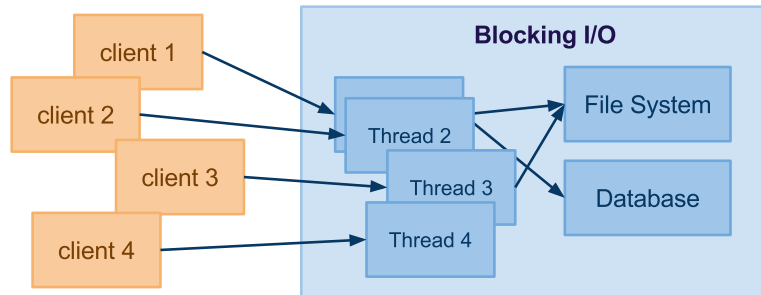
- ## Characteristics
  - *the thread is reserved for the connection*
  - *When processing of the request requires other interactions with DB/FS or network communication is slow*
    → scales very bad as the thread's execution is "blocked"

Non-Blocking I/O Model

- ## Connections maintained by the OS, not the Web app
  - *The Web app registers events, OS triggers events when occur*
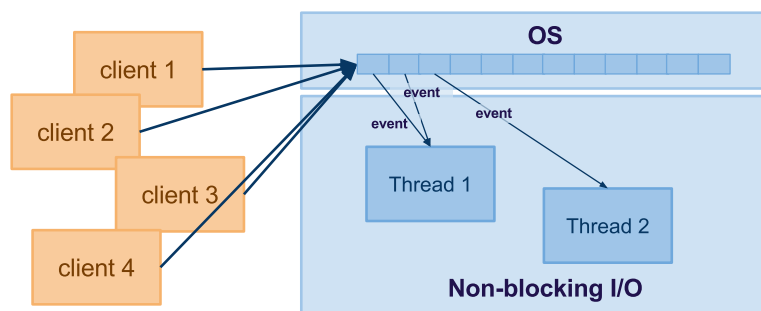
- ## Characteristics
  - *Event examples: new connection, read, write, closed*
  - *The app may create working threads, but controls the number!*
    → much less number of working threads as opposed to blocking I/O

Overview

- Architecture
- 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:*

```xml
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:*

```java
package com.vitvar.mdw;

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

public class Main extends HttpServlet {
    public doGet(HttpServletRequest request, HttpServletResponse response) {
        // GET method implementation here
    }

    public doPost(HttpServletRequest request, HttpServletResponse response) {
        // POST method implementation here
    }

    // other methods such as doPost, doDelete, doOptions
}
```

Support for Sessions

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

```java
// method doGet in a servlet
public doGet(HttpServletRequest request, HttpServletResponse response) {
    // access the session object through the request
    HttpSession session = request.getSession();

    // unique identification of the session, the value used for the cookie
    String id = session.getId();

    // get the value of the attribute
    Object value = session.getAttribute("data");

    // set the value of the attribute
    session.setAttribute("data", new String("some data"));

    // this will set a max-age of the session cookie
    session.setMaxInactiveInterval(3600);
}
```