JavaOne℠

Sun's Worldwide Java Developer Conference

Servers and Server Extensions

Java™ Server and Servlets

Pavani Diwanji
Dave Connelly
Prasad Wagle

Outline

- Jeeves - The Java™ Web Server
- Technical Overview
- Implementation and Examples
- Administration and Performance
- Demos
- Questions?

Why?

- Why do yet another web server?
- Why do a server toolkit in Java?
- Why develop the web server in Java?

Why?

- Java Ubiquity: client and server side
- Cannot do toolkit without a reference Application Server
- Java: A better language for developing Applications
- Multiplatform servers and servlets

What Are Http Servlets?

- Servlets are Java objects that extend the web server functionality: server-side applets
- Dynamically loadable at runtime
- Loaded from the local disk or the net

Http Servlets...

- Servlets are identified using a class name or a URL (i.e http://host/<myservlet.class>)
- Servlets are instantiated at server startup time or on demand
- Servlets live on until server decides to remove them
- Arguments can be passed to a servlet at initialization time and per request

Http Servlet Interface

```
interface HttpServlet {
  initialize(ServletContext, ServerProperties);
  service(HttpRequest, HttpResponse);
  destroy();
  …
}
```

- Handles to the per request, response information like input and output streams etc.
- ServletContext: Context information accessible to the servlet

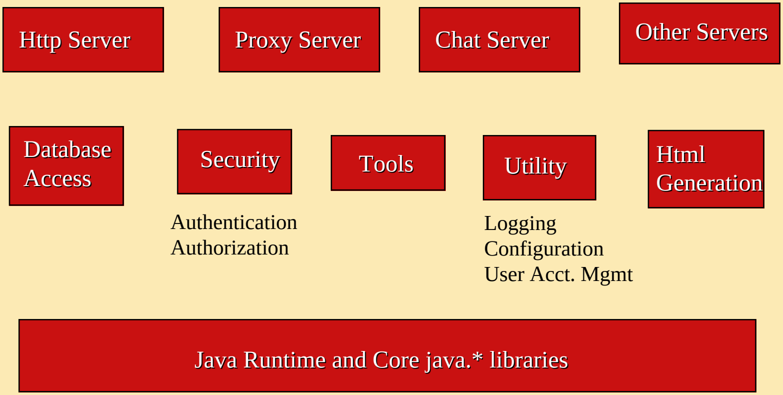
Http Servlet Invocation

- Conventional invocation (CGI style)
  - http://<server host>/date.html?<args>
- Explicit invocation:
  - http://<server host>/<servlet url>?<args>
- Server side includes
  - Servlet tag
  - Parsing is expensive

Example Servlets

- Web servlets: All the existing cgi-scripts can be done better as Java servlets
- Online publishing helper servlets
- Financial servlets: online banking etc.
- Other Cool Servlets: Chat, Calendar, Notifier
- Proxy servlets: Filtering, Traffic characterizing etc.

Web Server Toolkit



Http Server

Proxy Server

Chat Server

Other Servers

Database Access

Security

Authentication
Authorization

Tools

Utility

Logging
Configuration
User Acct. Mgmt

Html Generation

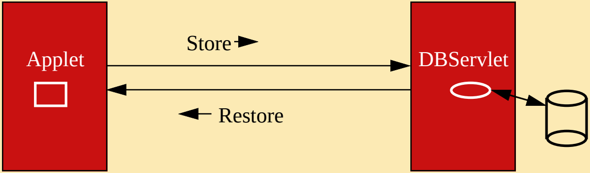
Java Runtime and Core java.* libraries

Database Connectivity

- Example servlets using JDBC APIs: Connectivity to Sybase and Oracle databases
- Capability to execute SQL queries on the database
- Working with vendors to provide higher level database connectivity classes as well

Simple, But Useful

- Persistant Applets using DBServlet
- Serialization API
- Security



Security: Servlets

- Security needed for network servlets
- Runtime server security manager gives policy control to the server administrator
- Servlets loaded from the network runs in a separate thread in a distinct group
- Use signed class loader for loading signed servlets
- First release: Allow only local servlets and signed network servlets

Security: Access Control

- ACL classes can guard multiple entities: Servers, servlets, files, directories
- Access Control Manager is consulted by the server, for protected entities
- Default Server ACL
- Servlets can do custom authorization

Security: Authentication

- Authentication: Standard security protocol support i.e. SSL
- Make use of the base encryption capabilities (java.security)
- Current state: Basic Http Authentication, ACL based Authorization

Dynamic Html Generation

- Really simple but useful: sun.server.html
- Classes: HtmlPage, HtmlTag, HtmlText..
- Code:

```
HttpOutputStream out = resp.getOutputStream();
HtmlPage = new HtmlPage(“Welcome Page”);
page.addElement(new HtmlTagPair(“H1”, “Hello”));
page.write(out);
out.flush();
```

Server Administration

- Most existing servers allow local administration
- Some servers allow administration through the Web but they use a fixed forms based interface
- Dynamic, user-customizable server tools through the use of applets (client side) and corresponding server objects

Roadmap

- When?
  - First Internal release – April
  - Demo in Developer conference – May
  - Alpha soon, FCS later this year
- Provide the base extensible framework and a completely Java-based web server along with some sample demo servlets to start with
- Use the toolkit to feature other servers later

Java Server and Servlets

David Connelly
JavaSoft

Overview

- Server foundation class
- HttpServer implementation
- Servlet API and example
- Network servlets
- Performance issues

Class sun.server.Server

- Abstract class for implementing connection-oriented servers
- Manages pool of handler threads
- Many tuneable parameters
- Parameters can be changed on-the-fly

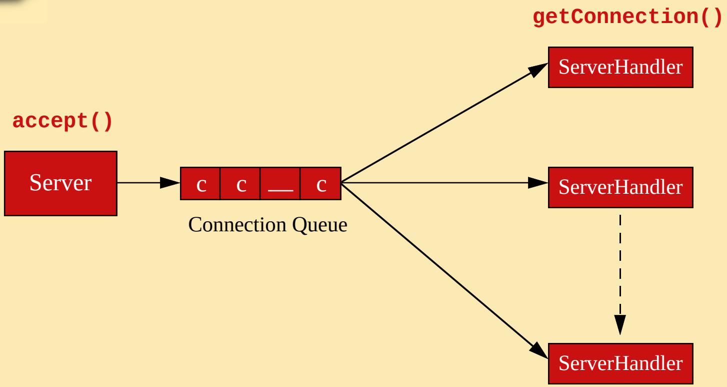
sun.server.Server

```
public abstract
class Server implements Runnable {
  public Server(ServerProperties props);
  public Socket getConnection();
  public void run();
  public abstract ServerHandler createHandler();
  …
}
```

sun.server.ServerHandler

```
public abstract
class ServerHandler implements Runnable {
  public void run();
  public abstract
  void handleConnection(Socket s)
    throws IOException;
  …
}
```

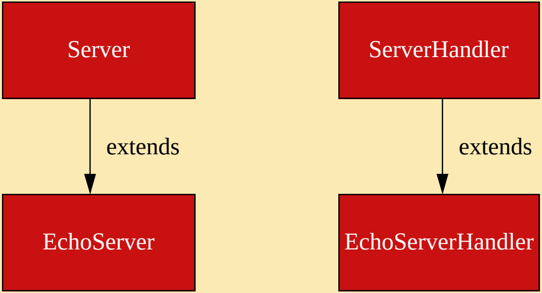
How it Works...


getConnection()
accept()
Server
c
c
c
Connection Queue
ServerHandler
ServerHandler
ServerHandler

Implementing a Server

- Extend sun.server.Server
  - implement createHandler()
- Extend sun.server.ServerHandler
  - implement handleConnection()

Example: EchoServer



`createHandler()`

`handleConnection()`

Example: EchoServer

```
class EchoServer extends Server {
   public EchoServer() {
        init(System.getProperties());
   }
   public ServerHandler createHandler() {
        return new EchoServerHandler(this);
   }
   public static void main(String args[]) {
        new EchoServer().run();
   }
}
```

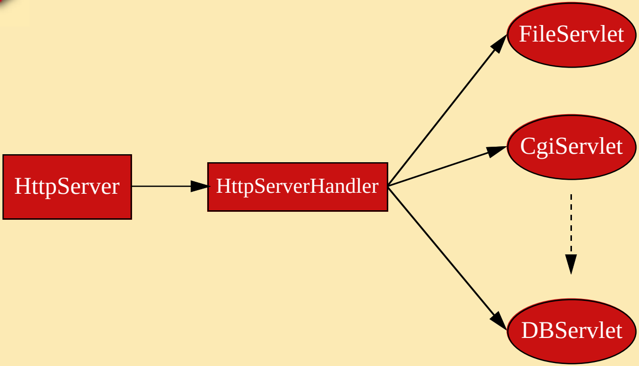
EchoServerHandler

```
class EchoServerHandler extends ServerHandler {
  void handleConnection(Socket s) throws IOException {
    InputStream in = s.getInputStream();
    OutputStream out = s.getOutputStream();
    int len; byte[] buf = new byte[512];
    while ((len = in.read(buf)) != -1) {
      out.write(buf, 0, len);
    }
    in.close(); out.close();
  }
  …
}
```

HttpServer

- Extends sun.server.Server
- Inherits from sun.server.Server:
  - Threads management
  - Observable properties
- Supports local and network servlets
- Core extensions through servlets

How it Works...


HttpServer
HttpServerHandler
FileServlet
CgiServlet
DBServlet

Core Servlets

- FileServlet
  - Handles file GET requests
  - Ram cache
- CgiServlet
  - For backwards compatibility
- StubServlet
  - For downloading network servlets

HttpServlet Interface

```
interface HttpServlet {
  initialize(ServletContext, ServerProperties);
  service(HttpRequest, HttpResponse);
  destroy();
  …
}
```

BasicHttpServlet

- Implements HttpServlet
- Base class for writing most servlets
- Provides default implementation
  - initialize(), destroy(), etc…

HttpRequest

- Encapsulates Http request message
- Input stream for reading entity data
- Contains all header field values
- Headers parsed on demand

HttpResponse

- Encapsulates Http response message
- Used to set response headers
- Output stream for writing entity data

Example: HelloServlet

```
class HelloServlet extends BasicHttpServlet {
 static String hello = “Hello, world\r\n”;
 public HelloServlet() {};
 public void service(HttpRequest req,
                     HttpResponse res) {
  res.setContentLength(hello.length());
  res.writeHeaders();
  res.getOutputStream().print(hello);
 }
}
```

FormServlet

- Automates parsing of form input
- Passes form input in Hashtable
- Servlets can extend FormServlet
- Override sendResponse()
- Easier than writing cgi-bin

Example: FormServlet

```
class SimpleForm extends FormServlet {
  public void sendResponse(HttpResponse res,
                           Hashtable table) {
    HttpOutputStream out = res.getOutputStream();
    res.putHeaders();
    Enumeration e = table.keys();
    while (e.hasMoreElements()) {
      String key = (String)e.getNextElement();
      out.print(key + “ = “ + table.get(key));
    }
  }
}
```

Network Servlets

- StubServlet
  - manages downloaded servlets
- Separate thread group, class loader
- Working on signed servlets support

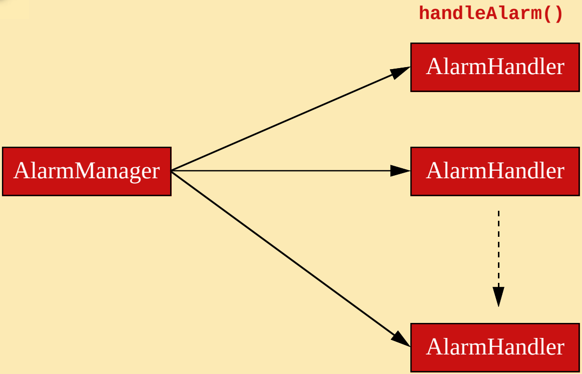
Performance Enhancements

- Reuse per-request data
- Minimize object allocation
- Reduce GC overhead
- Minimize synchronized method calls
- HttpInputStream, HttpOutputStream

Keep-alive

- Support keep-alive count and timeout
- Content length maintains keep-alive
- Limitations with read()
- Alarm timer closes connection
- Thread.interrupt() is better solution

AlarmManager Class



Where Are We Now?

- HTTP 1.0 support, CGI
- Keep-alive
- HttpServlets: API and examples
- Prototype: signed network servlets
- ACL support
- Win32, Solaris releases

Where Are We Going?

- Proxy
- Filters
- SSL
- Session support
- Administration tools

Jeeves Administration and Performance

Prasad Wagle
JavaSoft

Jeeves Installation

- Simple
- Single binary release
- Disk space
- Memory
- Documentation - http://<server>/doc/

Jeeves Administration

- Dynamic
- Remote
- Secure
- Front end is Java applets and HTML
- Back end is composed of servlets

Jeeves Administration

- Changing server properties
- Monitoring the server
- Administering servlets

Administration Examples

- Server properties
  - MinThreads, maxThreads
  - Keep-alive
  - Memory cache size
  - Document hierarchy mappings

Administration Examples

- Logging
  - Files or databases
  - Format (Common Log, Free format)

Administration Examples

- Controlling access to documents
  - Based on user, host, groups
  - Create users, groups
- Security

Jeeves Monitoring

- Dynamic
- Number and rate of requests
- Type of requests
- Callbacks when certain events happen

Servlet Administration

- Servlet startup
- View loaded servlets
- Upload a servlet
- Servlet security restrictions
- Stop a servlet

Performance Measurement

- Confusion
- No standard metric
- No standard realistic workload
- Webperf
- SPECweb

Jeeves Performance

- 150 HTTPops/sec on an Ultra
- 1 HTTPop = 1 request = 1 hit
- 150 HTTPops/sec = 15 million/day
- Very respectable
- Will release SPECweb results

Jeeves Performance

- Java runtime:
  - Garbage collection
  - Threads
  - Monitors
  - JIT compiler

Credits and Contact Info.

JavaSoft

Engineering Credits

- James Gosling (Architect)
- Dave Connelly (Server, Protocols)
- Satish Dharmaraj (Security)
- Pavani Diwanji (Lead Engineer)
- Rachel Gollub (Release)
- Marianne Mueller (Security)
- Freeman Murray (Html Gen., Demos)
- Prasad Wagle (Server, Performance)
- Dave Brownell (Security)

Contact Information

- http://www.javasoft.com/products/jeeves/
- jeeves@goa.eng.sun.com

Questions

?