Announcements

• Comments on project proposals will go out by email in next couple of days ...
3-Tier Using TP Monitor

TP monitor

client application

interface (API, presentation, authentication)

program flow

router

communication manager

registered programs

resources

TP services

transaction management

wrapper

resource

wrapper

resource

wrapper

resource
• Applications interacting by messages

- SmartQuotation adapter
- database adapter
- SmartForecasting adapter
- e-mail adapter
- XYZ adapter

Integrating application (contains the composition logic)

Message Broker
J2EE Application Server Architecture

- Servlets
- JavaServer Pages (JSP)
- Java API for XML Processing (JAXP)
- JavaMail
- Java Authentication and Authorization Service (JAAS)

- Enterprise Java Beans (EJB)
- Java transaction API (JTA)
- Java Message Service (JMS)
- Java Naming/Directory Interface (JNDI)

- Java DataBase Connectivity (JDBC)
- Java 2 Connector Architecture (J2CA)

Support for communication and presentation
Support for the application integration
Support for access to resource managers
J2EE Application Server Architecture

- Servlets
- JavaServer Pages (JSP)
- Java API for XML Processing (JAXP)
- Java Authentication and Authorization Service (JAAS)
- Enterprise Java Beans (EJB)
- Java Message Service (JMS)
- Java DataBase Connectivity (JDBC)
- Java 2 Connector Architecture (J2CA)
- Java transaction API (JTA)
- Java Naming/Directory Interface (JNDI)
- JavaMail

Supports for communication and presentation
Support for the application integration
Support for access to resource managers
• Client logic in servlet code
• Avoids CGI’s process creation overhead
• JVM can be separate from Web Server
• Still avoids process creation overhead
Trivial HTTP Servlet

- Servlet reads request parameters from `req`
- Servlet writes HTML reply to `resp`

```java
public class MyServlet extends HttpServlet {
    public void doGet(
        HttpServletRequest req, HttpServletResponse resp) ...
    {
        String remoteHost = req.getRemoteHost();
        resp.setContentType( "text/html" );
        PrintWriter out = resp.getWriter();
        out.println( "<html>" );
        out.println( "<body><h1>Hello, " + remoteHost + "!</h1></body></html>" );
    }
}
```
Forms

- HTML form items are made available in req

```java
public class HelloServlet extends HttpServlet {
    public void doGet(
        HttpServletRequest req, HttpServletResponse resp) ... {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<body><h1>Hello, " +
            req.getParameter("username") +
            "!</h1></body></html>" );
    }
}
```

<form method=get action="/servlet/HelloServlet">
<input type=text name=username size=20>
<input type=submit value="introduce yourself">
```
Servlet Life Cycle

- First reference creates servlet object
- Subsequent references just invoke methods
  - request and response parameters
Servlet Life Cycle

- Servlet notified when created and destroyed
- Can save state between calls

```java
public class HitCountServlet extends HttpServlet {
    int timesAccessed;
    public void init(ServletConfig conf) ... {
        timesAccessed = 0;
    }
    public void doGet(
        HttpServletRequest req, HttpServletResponse resp) ... {
        ... timesAccessed++;
    }
}
```
It is more complicated than that. Servlet may be destroyed by container anytime, it must save state to stable storage:

```java
public class HitCountServlet extends HttpServlet {
    ...
    public void destroy() {
        ...
        outfile.writeInt(timesAccessed);
        ...
    }
}
```
Servlet Life Cycle

• Real examples: serially reusable resources that are expensive to create
• Database connection pools
Each call has its own req and resp parameter
But servlet code -- *and all code called by the servlet* -- must be thread-safe!
Session Context

... HTTPSession theSession = req.getSession(true); ...

- Server will create session context
  - assign unique ID
  - track session by some method such as cookies or (yuck) URL encoding
  - more on sessions later
- Servlet can store (anything) in session context, it will persist between calls
• A template engine:

Database or other resource → Template processor → Web server → Browser

Template and instructions
• A JSP template is HTML with snippets of Java embedded in it

• Here is a really simple one ...

  <html>
  <body>
    Hello, visitor. It is now
    <%= new java.util.Date().toString() %>
  </body>
  </html>
Here is one with control flow!

```html
<html>
<body>
<% java.util.Date theDate = new java.util.Date(); %>
<% if (theDate.getHours() < 12) { %>
Good morning,
<% } else { %>
Good afternoon,
<% } %>
visitor. It is now <%= theDate.toString() %>
</body>
</html>
```
• The template is compiled (once) into a servlet
• Later references re-use existing servlet
J2EE Application Server Architecture

- Servlets
- JavaServer Pages (JSP)
- Java API for XML Processing (JAXP)
- Java Authentication and Authorization Service (JAAS)
- Enterprise Java Beans (EJB)
- Java transaction API (JTA)
- Java Message Service (JMS)
- Java Naming/Directory Interface (JNDI)
- Java Database Connectivity (JDBC)
- Java 2 Connector Architecture (J2CA)

support for communication and presentation

support for the application integration

support for access to resource managers
JavaMail

- Send and receive mail objects
- Straightforward
J2EE Application Server Architecture

- **Servlets**
- **JavaServer Pages (JSP)**
- **Java API for XML Processing (JAXP)**
- **JavaMail**
- **Java Authentication and Authorization Service (JAAS)**
- **Enterprise Java Beans (EJB)**
- **Java transaction API (JTA)**
- **Java Message Service (JMS)**
- **Java Naming/Directory Interface (JNDI)**
- **Java DataBase Connectivity (JDBC)**
- **Java 2 Connector Architecture (J2CA)**

Support for communication and presentation

Support for the application integration

Support for access to resource managers
JNDI Architecture

Naming

- Binding
- Context

Directory

- Attributes
- DirContext

Initial context

Application

Service Provider

Service provider interface
Naming and Directory

- Naming service
  - context = bindings of names to objects or contexts
  - enumerate subcontexts
- Directory service
  - dircontext = bindings of names to objects with attributes or dircontexts
  - enumerate attributes
- Initial Context = where you start looking for objects / services
  - (like X.500 or LDAP)
<table>
<thead>
<tr>
<th>Servlets</th>
<th>JavaServer Pages (JSP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java API for XML Processing (JAXP)</td>
<td>JavaMail</td>
</tr>
<tr>
<td>Java Authentication and Authorization Service (JAAS)</td>
<td>support for communication and presentation</td>
</tr>
<tr>
<td>Enterprise Java Beans (EJB)</td>
<td>Java transaction API (JTA)</td>
</tr>
<tr>
<td>Java Message Service (JMS)</td>
<td>Java Naming/Directory Interface (JNDI)</td>
</tr>
<tr>
<td>Java DataBase Connectivity (JDBC)</td>
<td>Java 2 Connector Architecture (J2CA)</td>
</tr>
</tbody>
</table>

support for the application integration

support for access to resource managers
JDBC Architecture

Connection

ResultSet

Statement

PreparedStatement

Callable Statement

Application

Service Provider

Service provider
JDBC Architecture II

Application

Service Provider

Connection

Oracle driver

JDBC/ODBC bridge

Sybase driver

Oracle driver

ODBC DB

Sybase
Capabilities

• Connections
  • pooling support
• SQL queries
  • precompiled queries
  • database stored queries
• Result sets
  • enumeration
  • scrollable
Capabilities

- Transaction control
  - auto-commit
  - explicit commit/rollback
  - savepoints
- Isolation levels
  - READ_(UN)COMMITTED
  - REPEATABLE_READ
  - SERIALIZABLE
Capabilities

• Distributed transactions
  • if supported by the database and driver
J2EE Application Server Architecture

- Servlets
- JavaServer Pages (JSP)
- Java API for XML Processing (JAXP)
- JavaMail
- Java Authentication and Authorization Service (JAAS)
- Enterprise Java Beans (EJB)
- Java transaction API (JTA)
- Java Message Service (JMS)
- Java Naming/Directory Interface (JNDI)
- Java Database Connectivity (JDBC)
- Java 2 Connector Architecture (J2CA)

**Support for communication and presentation**

**Support for the application integration**

**Support for access to resource managers**
Point-to-Point

sender  →  receiver
Topic Based (pub/sub)
Capabilities

- Message filtering (Boolean predicates) at receiver
- Durable subscriptions if desired
- Transactional support
  - JMS queues only
  - or other resource managers as well
J2EE Application Server Architecture

- Servlets
- JavaServer Pages (JSP)
- Java API for XML Processing (JAXP)
- JavaMail
- Java Authentication and Authorization Service (JAAS)
- Enterprise Java Beans (EJB)
- Java transaction API (JTA)
- Java Message Service (JMS)
- Java Naming/Directory Interface (JNDI)
- Java 2 Connector Architecture (J2CA)
- Java DataBase Connectivity (JDBC)

Supports for:

- Communication and presentation
- Application integration
- Access to resource managers
What is a bean?

- Java Object
- Container provides many standard services
  - location
  - life cycle
  - persistence
  - transactions
  - etc
Classes of Beans

- **Session**
  - embodies business logic associated with a session’s business process
  - always given a session context
  - not persistent
  - stateful or stateless
Classes of Beans

- **Entity**
  - represents a real-world entity
  - may be shared by multiple sessions
  - persistent beyond session or EJB container lifetime
  - persistence may be *bean-managed* (JDBC/JTA) or *container-managed*
Classes of Beans

- Message-Driven
  - communication by queueing (JMI) rather than RPC
EJB Classes and Stubs

- **client**
  - create/find/remove
  - business methods/remove

- **container**
  - notify
  - bean impl

- **home interface**
  - create/find/remove

- **home stub**
  - home impl

- **client interface**
  - business methods/remove

- **client stub**
  - client i/f impl

- **client**
  - server
Deployment

- XML deployment descriptors
  - identify programmer-provided code
  - dependencies
  - transactional behavior
  - security properties
- Container generates stubs