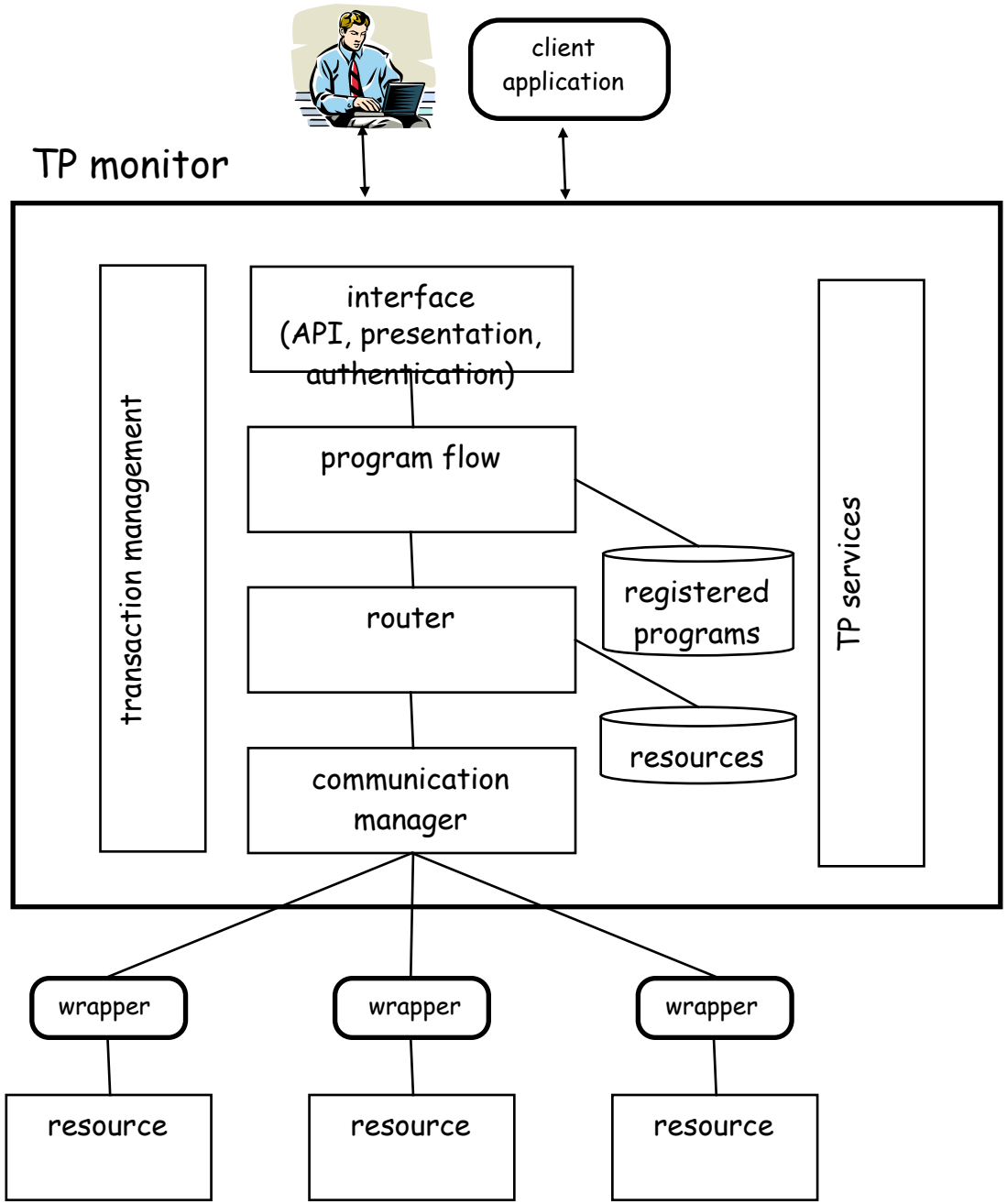
Announcements

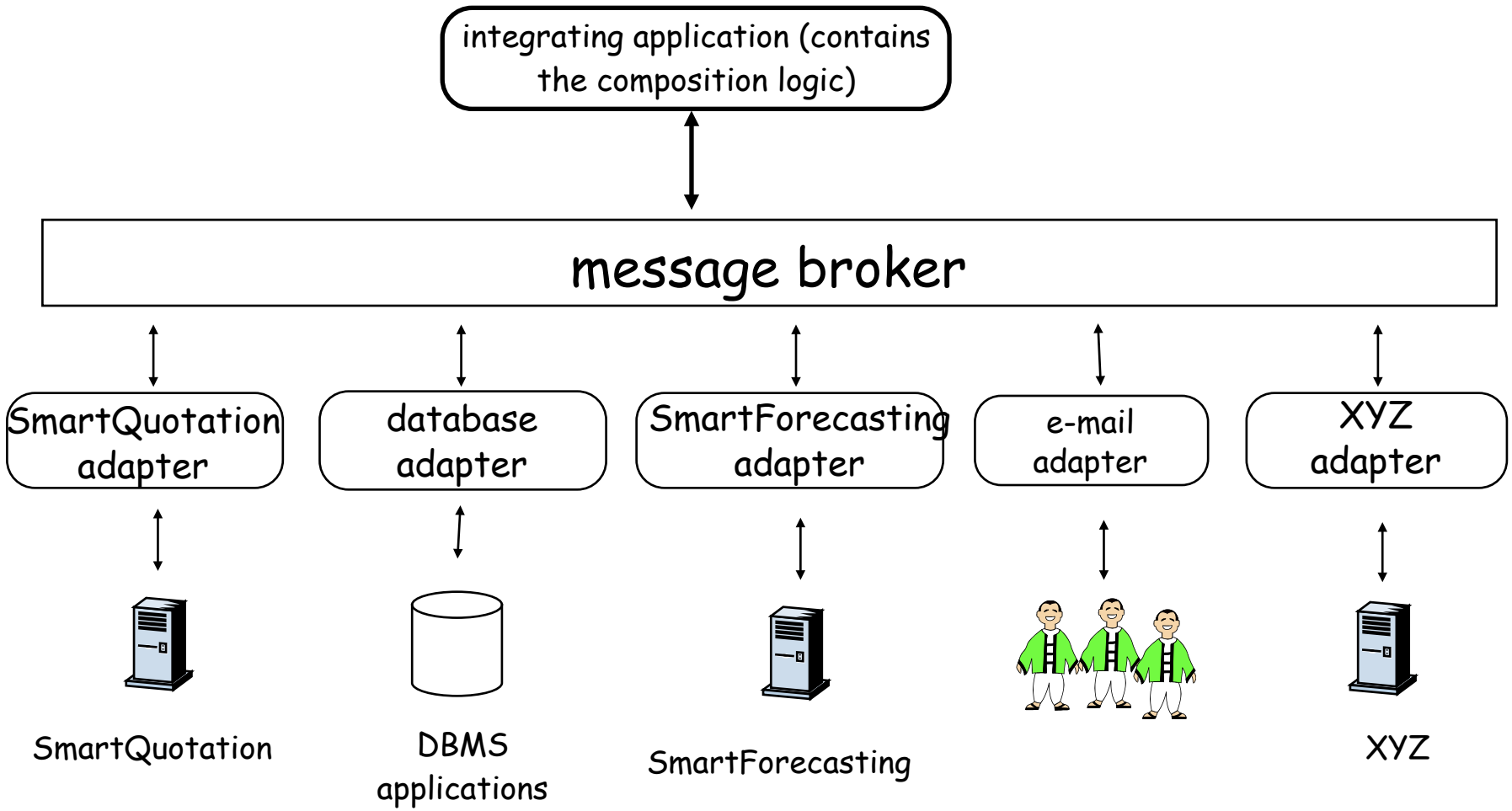
- Comments on project proposals will go out by email in next couple of days ...

3-Tier Using TP Monitor

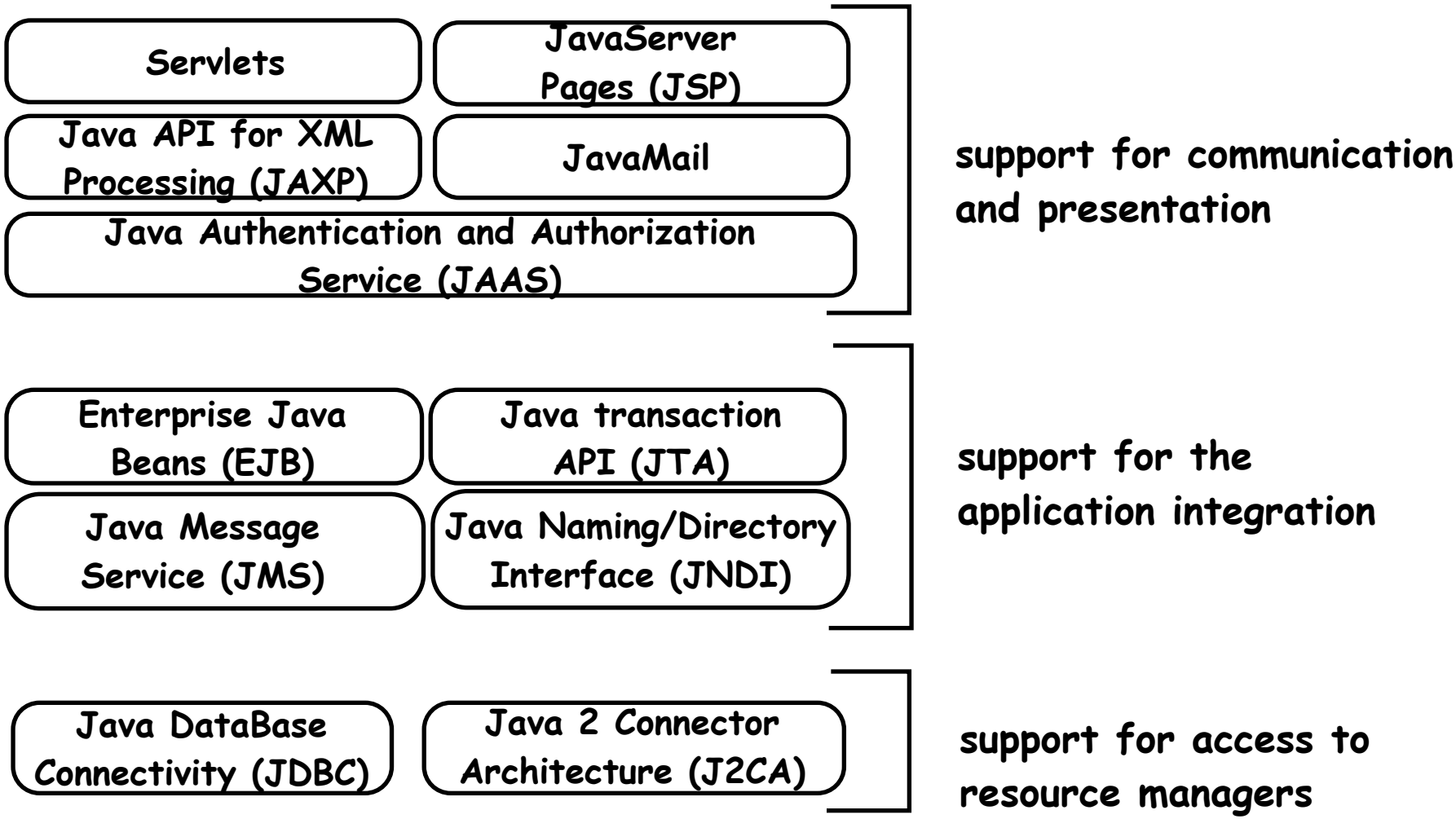


Message Broker

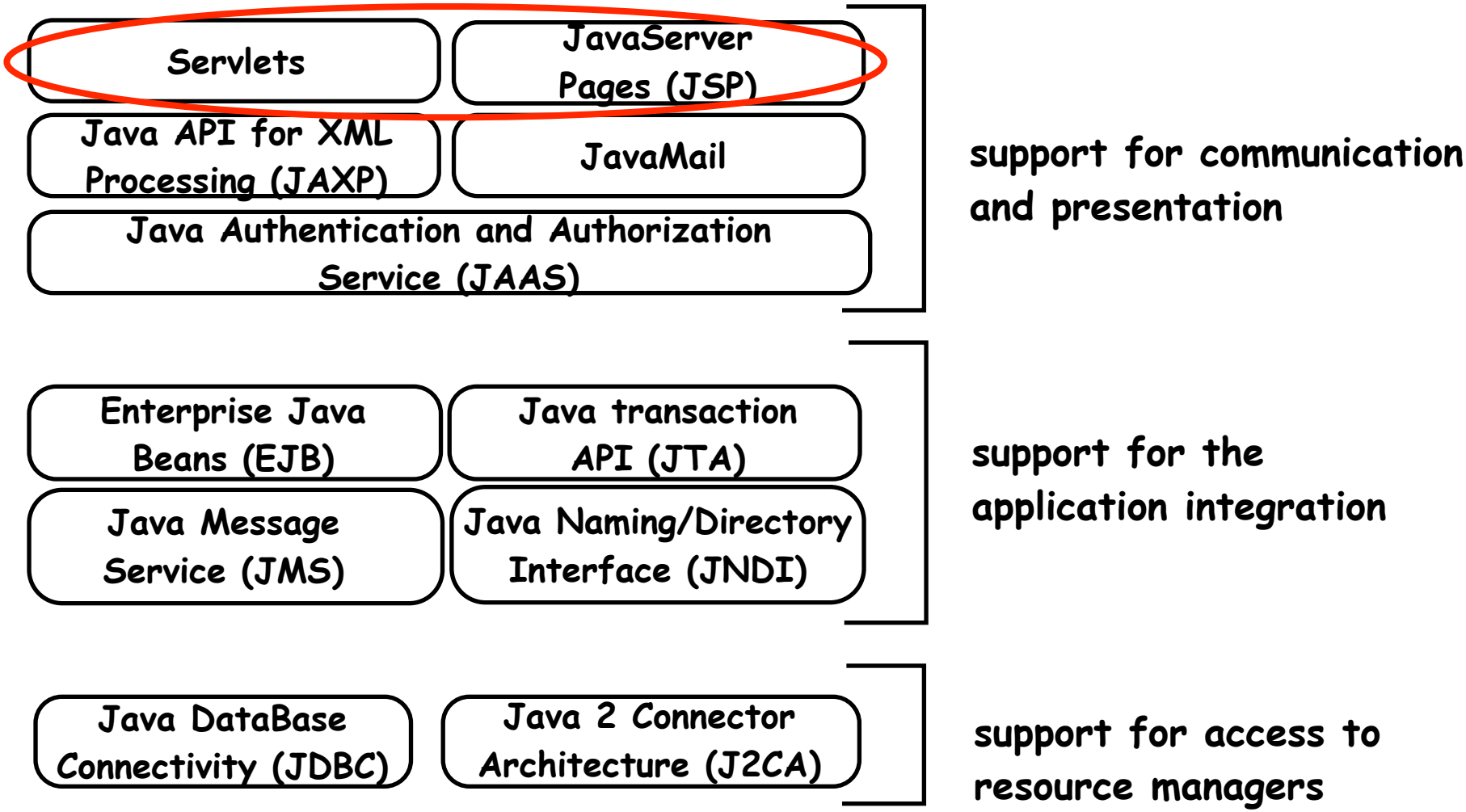
- Applications interacting by messages



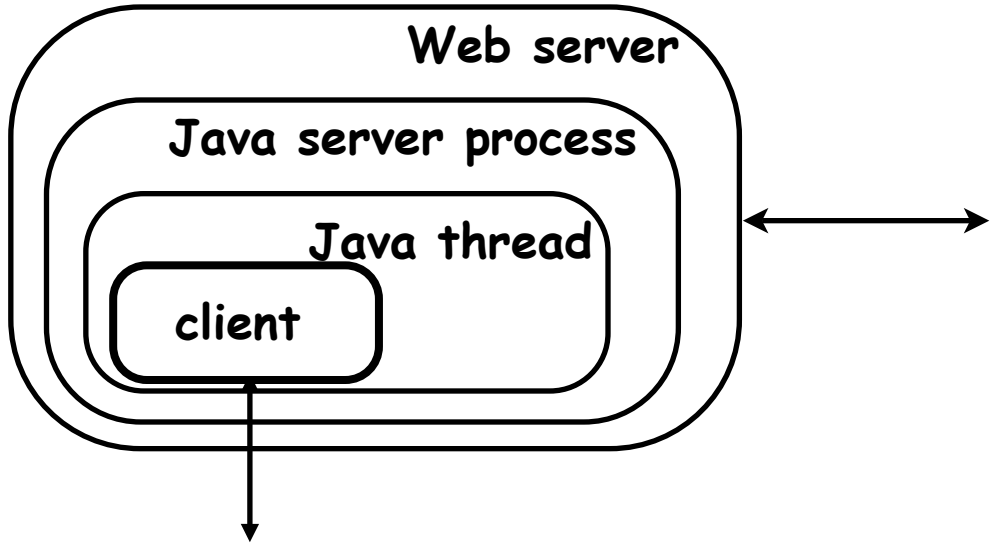
J2EE Application Server Architecture



J2EE Application Server Architecture

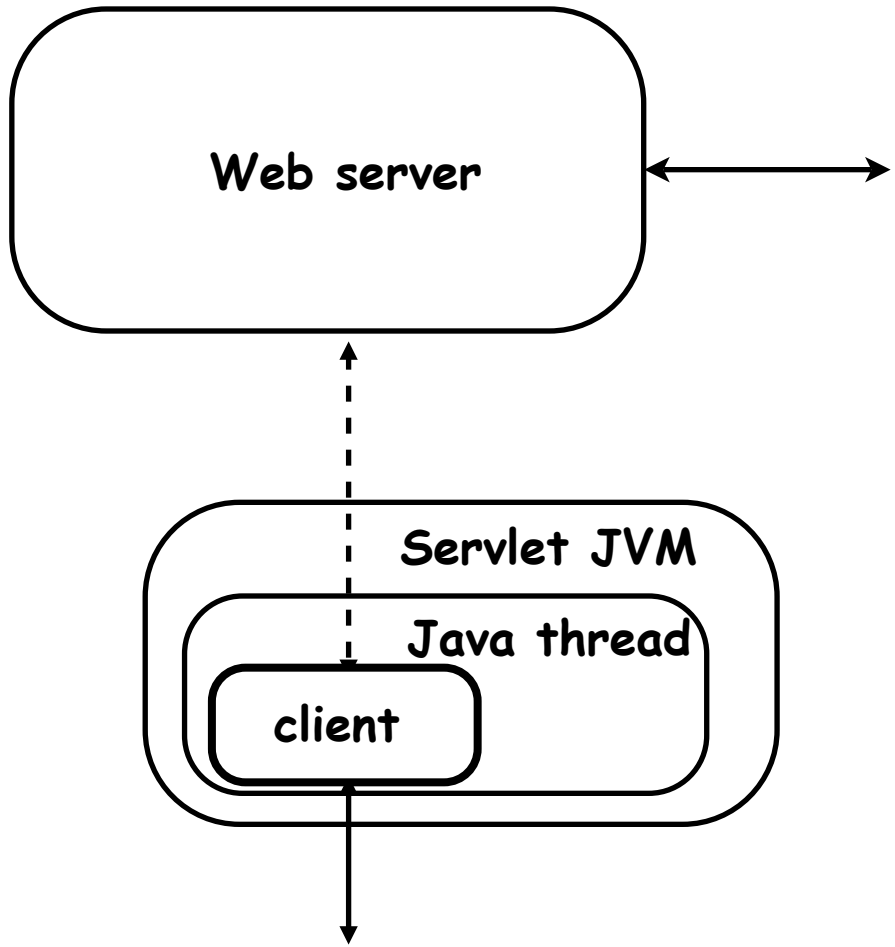


Servlets



- Client logic in servlet code
- Avoids CGI's process creation overhead

Servlets



- 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

```
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>” );
  }
}
```

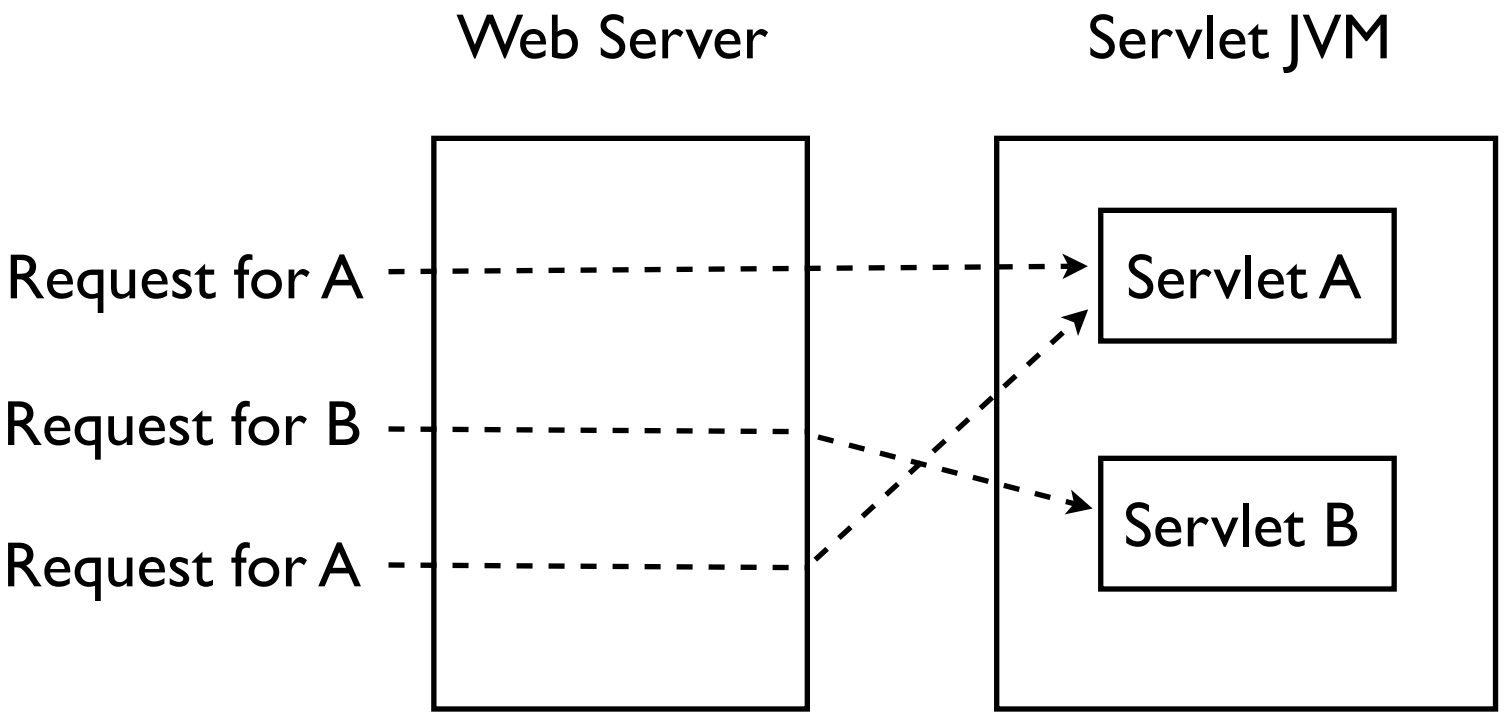
Forms

- HTML form items are made available in req

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

```
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>" );
}}
```

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

```
public class HitCountServlet extends HttpServlet {
  int timesAccessed;
  public void init(ServletConfig conf) ... {
    timesAccessed = 0;
  }
  public void doGet(
       HttpServletRequest req, HttpServletResponse resp) ... {
    ... timesAccessed++; ...
  }
}
```

Servlet Life Cycle

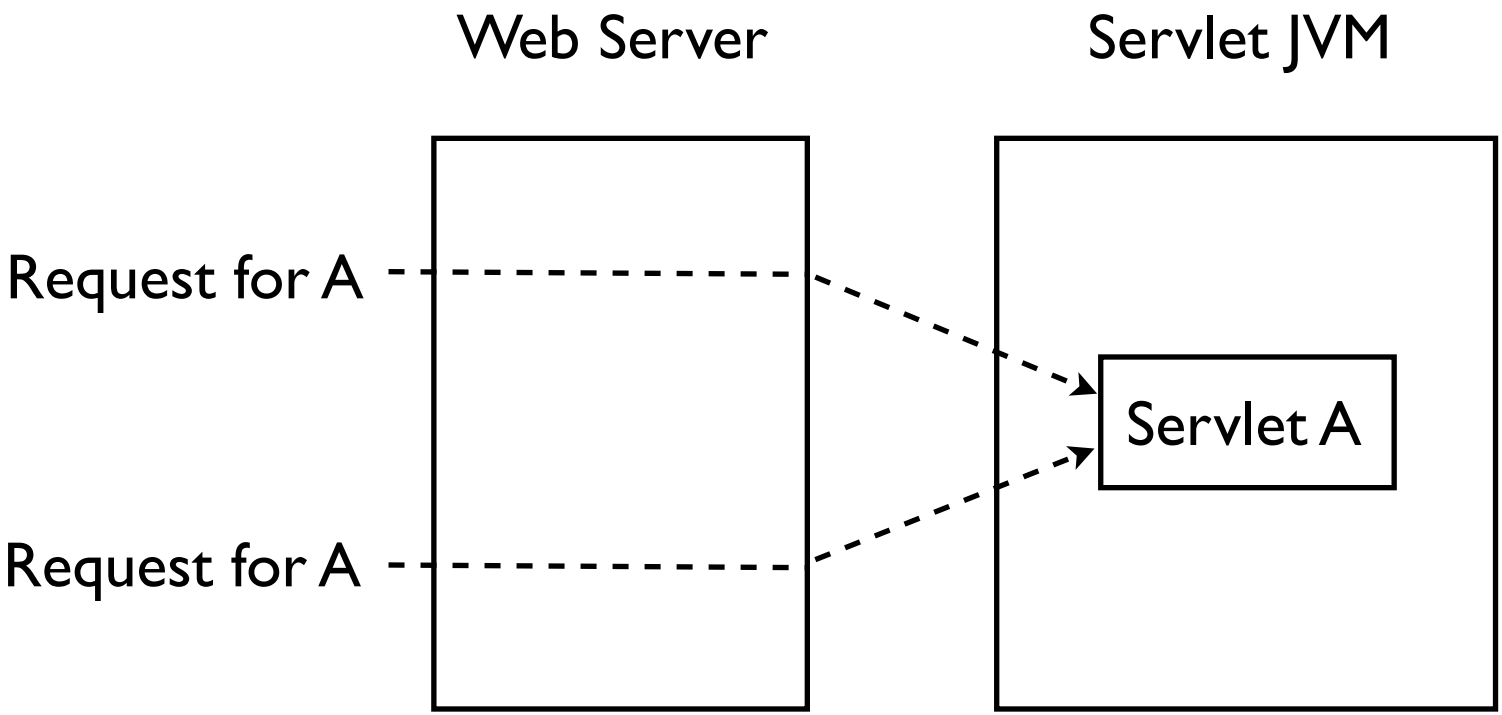
It is more complicated than that.
Servlet may be destroyed by container anytime,
it must save state to stable storage:

```
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

Concurrency - Thread Safety



- Each call has its own req and resp parameter
- But servlet code -- *and all code called by the servlet* -- must be thread-safe!

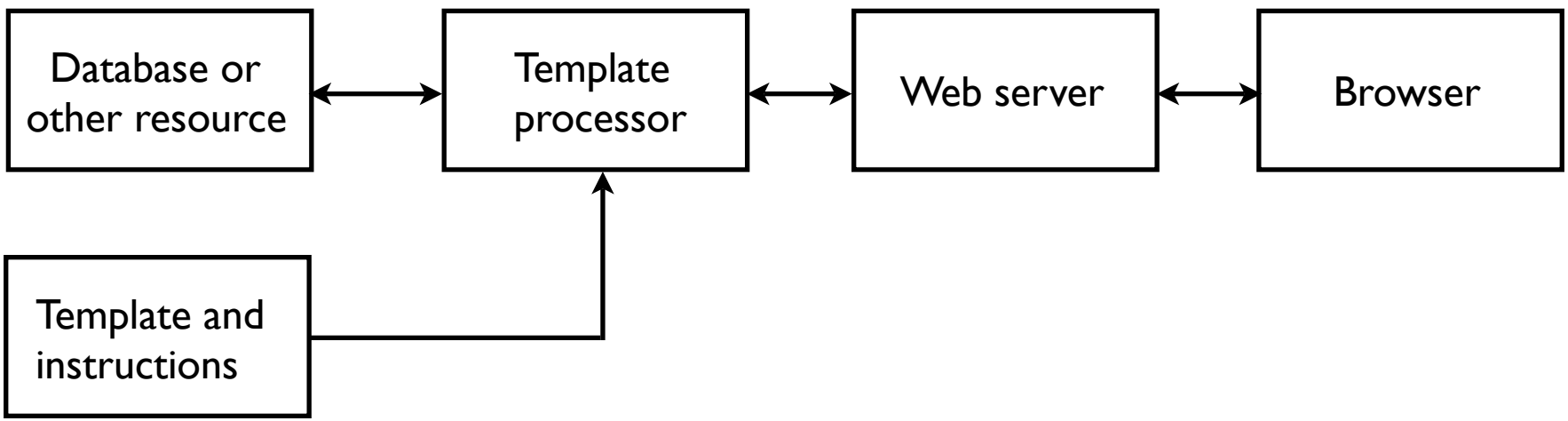
Session Context

```
...
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

JSP

- A *template engine*:



JSP

- A JSP template is HTML with snippets of Java embedded in it
- Here is a really simple one ...

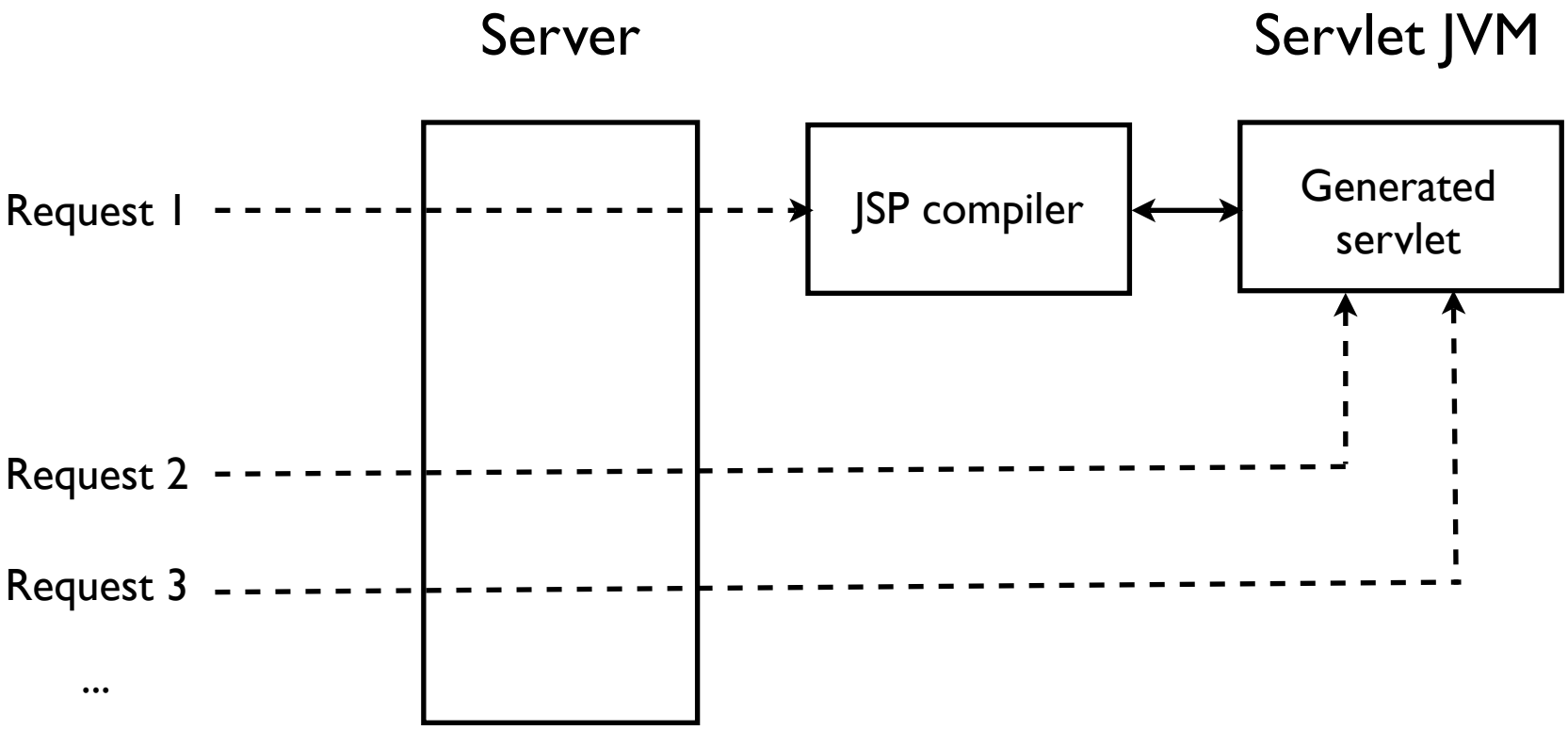
```
<html>
<body>
 Hello, visitor.  It is now
    <%= new java.util.Date().toString() %>
</body>
</html>
```

JSP

- Here is one with control flow!

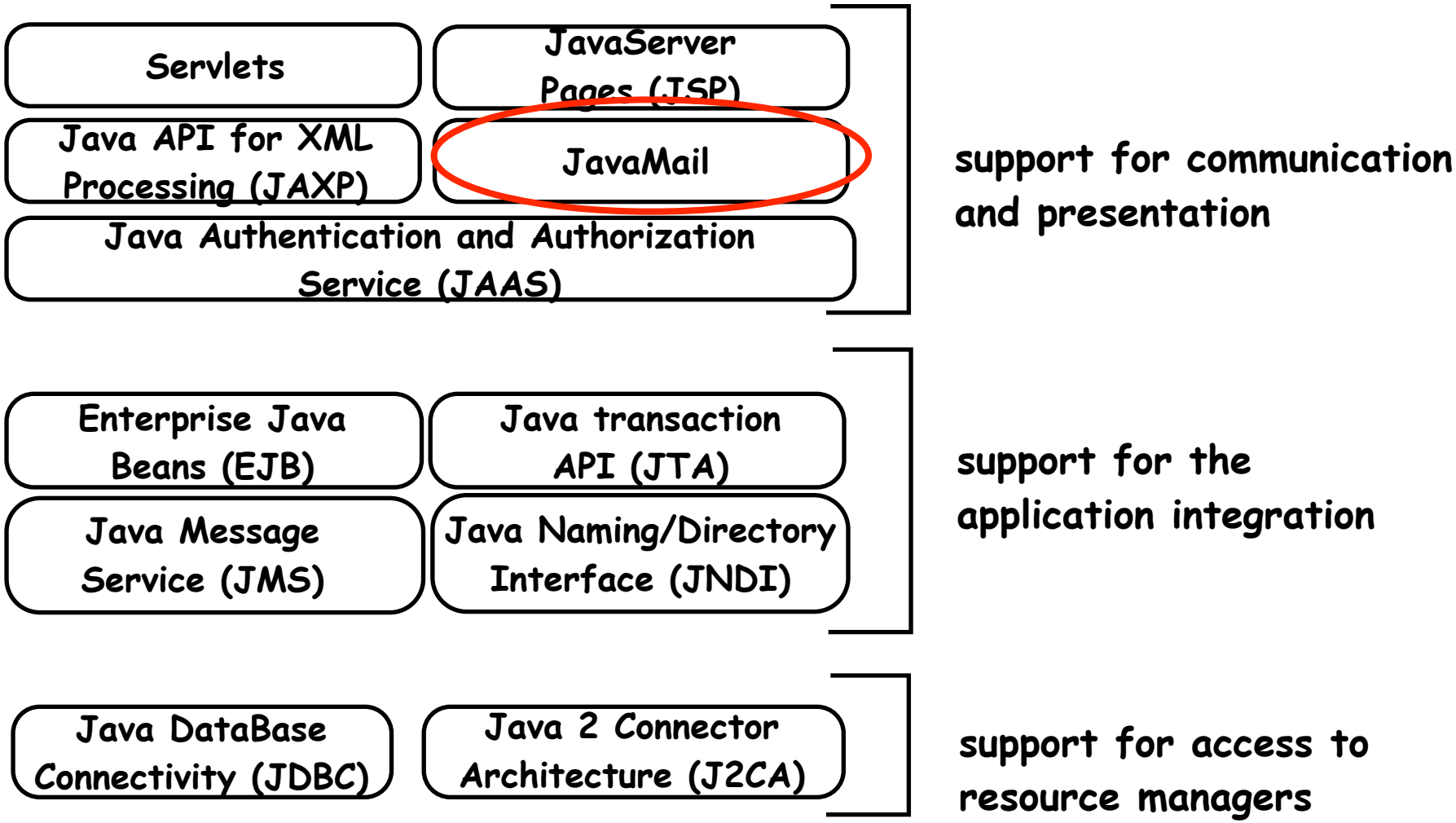
```
<html>
<body>
<% java.util.Date theDate = new java.util.Date(); %>
<% if (theDate.getHours() < 12) { %>
Good morning,
<% } else { %>
Good afternoon,
<% } %>
visitor.  It is now <%= theDate.toString() %>
</body>
</html>
```

JSP Life Cycle



- The template is compiled (once) into a servlet
- Later references re-use existing servlet

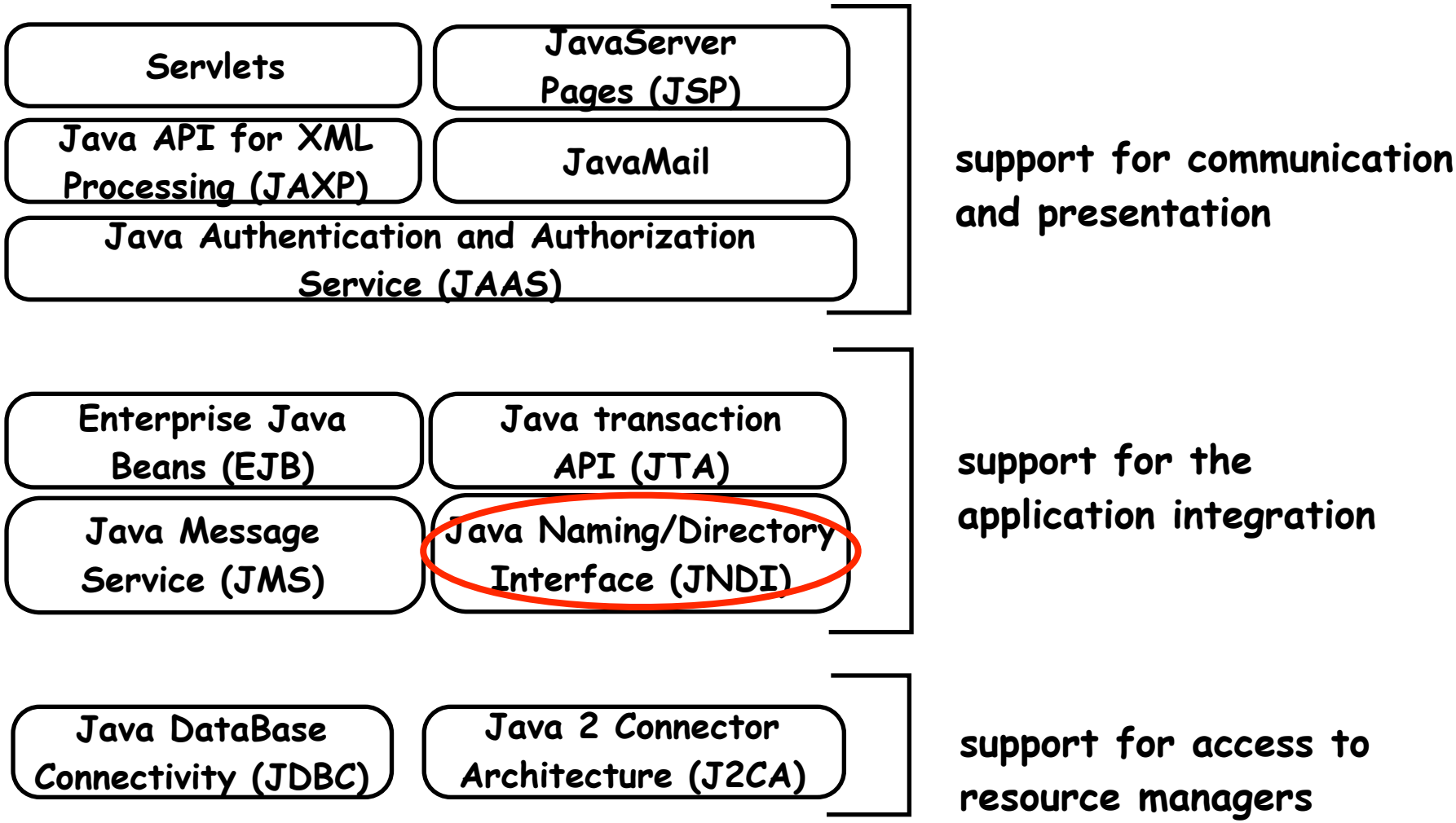
J2EE Application Server Architecture



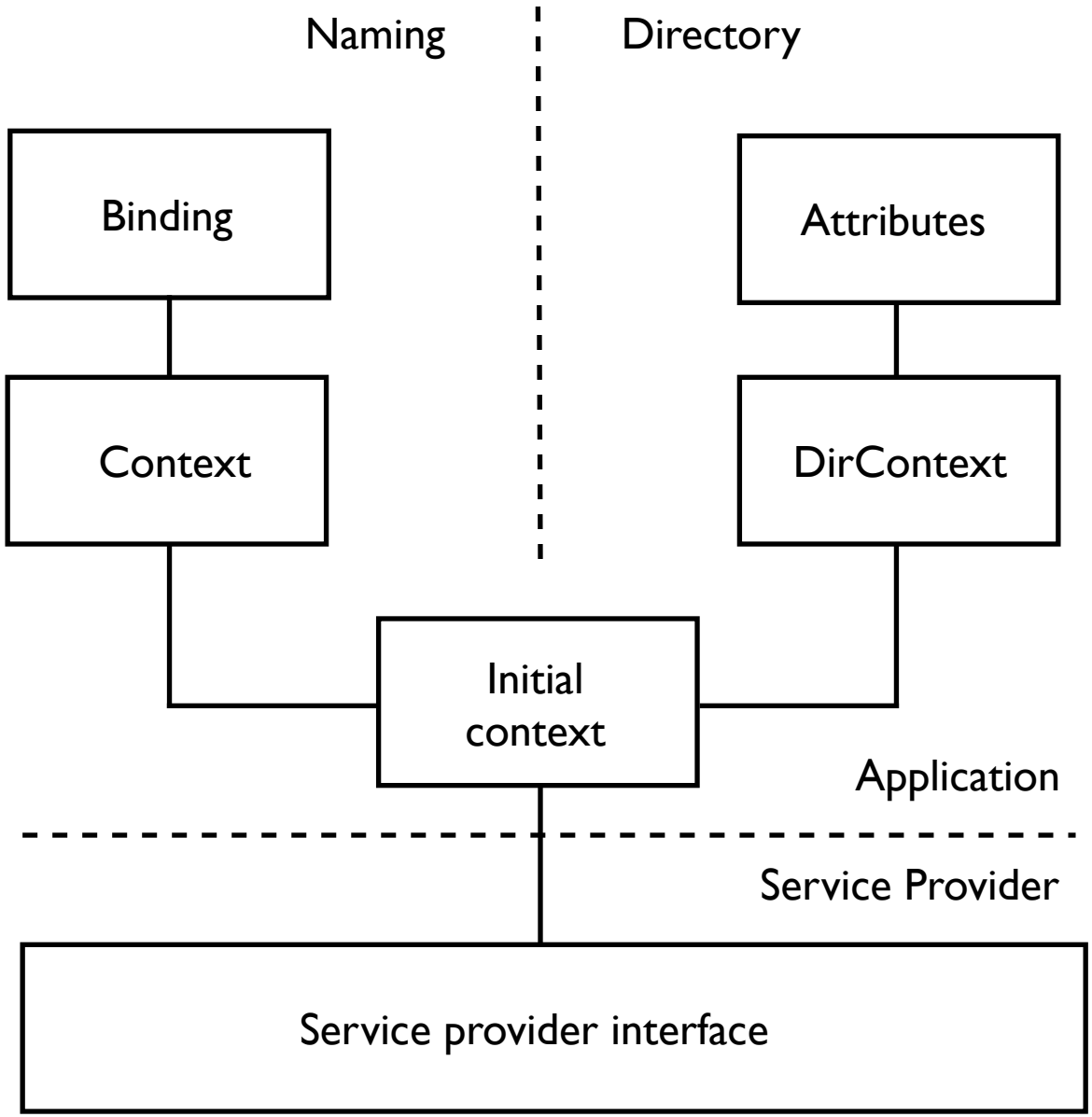
JavaMail

- Send and receive mail objects
- Straightforward

J2EE Application Server Architecture



JNDI Architecture

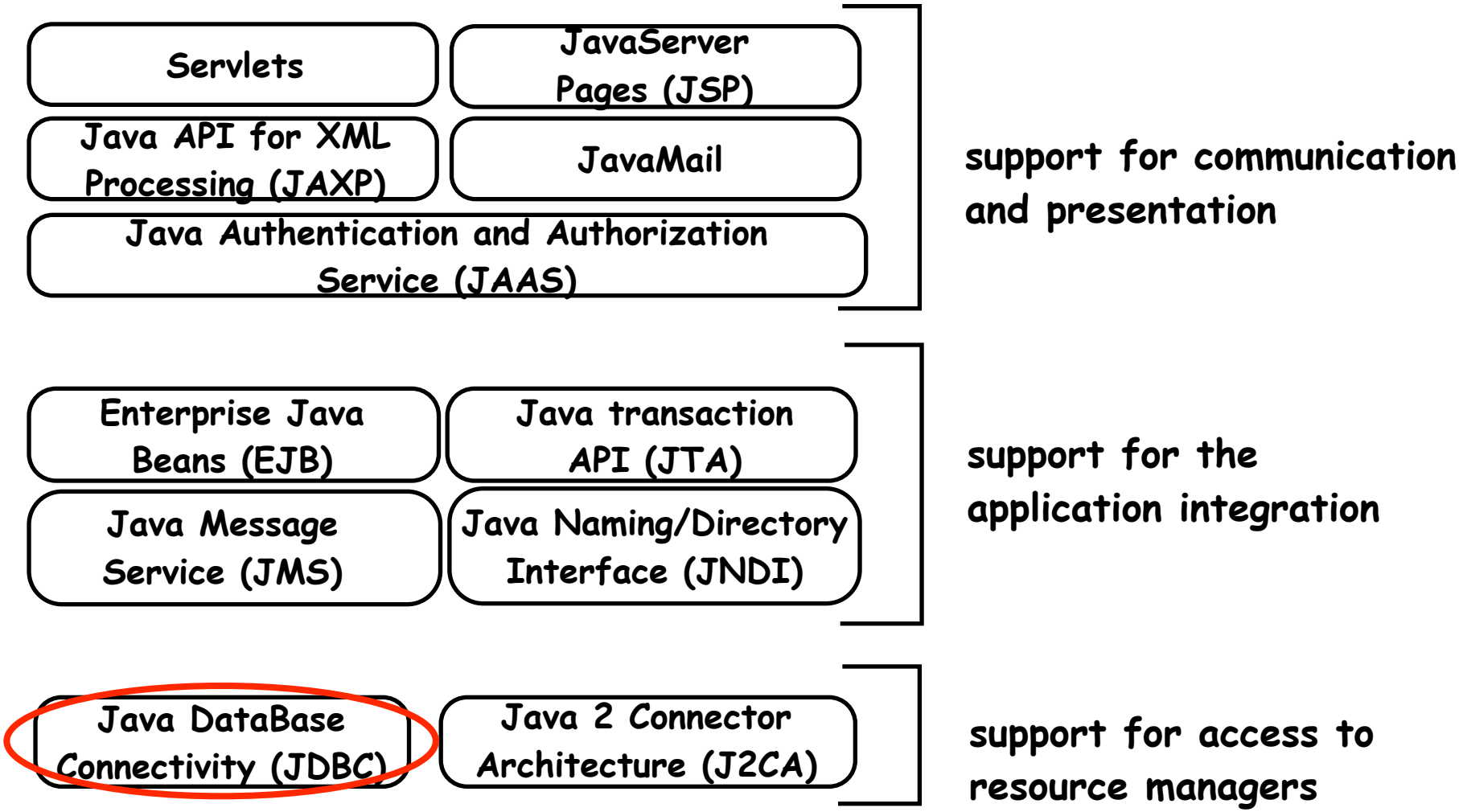


Naming and Directory

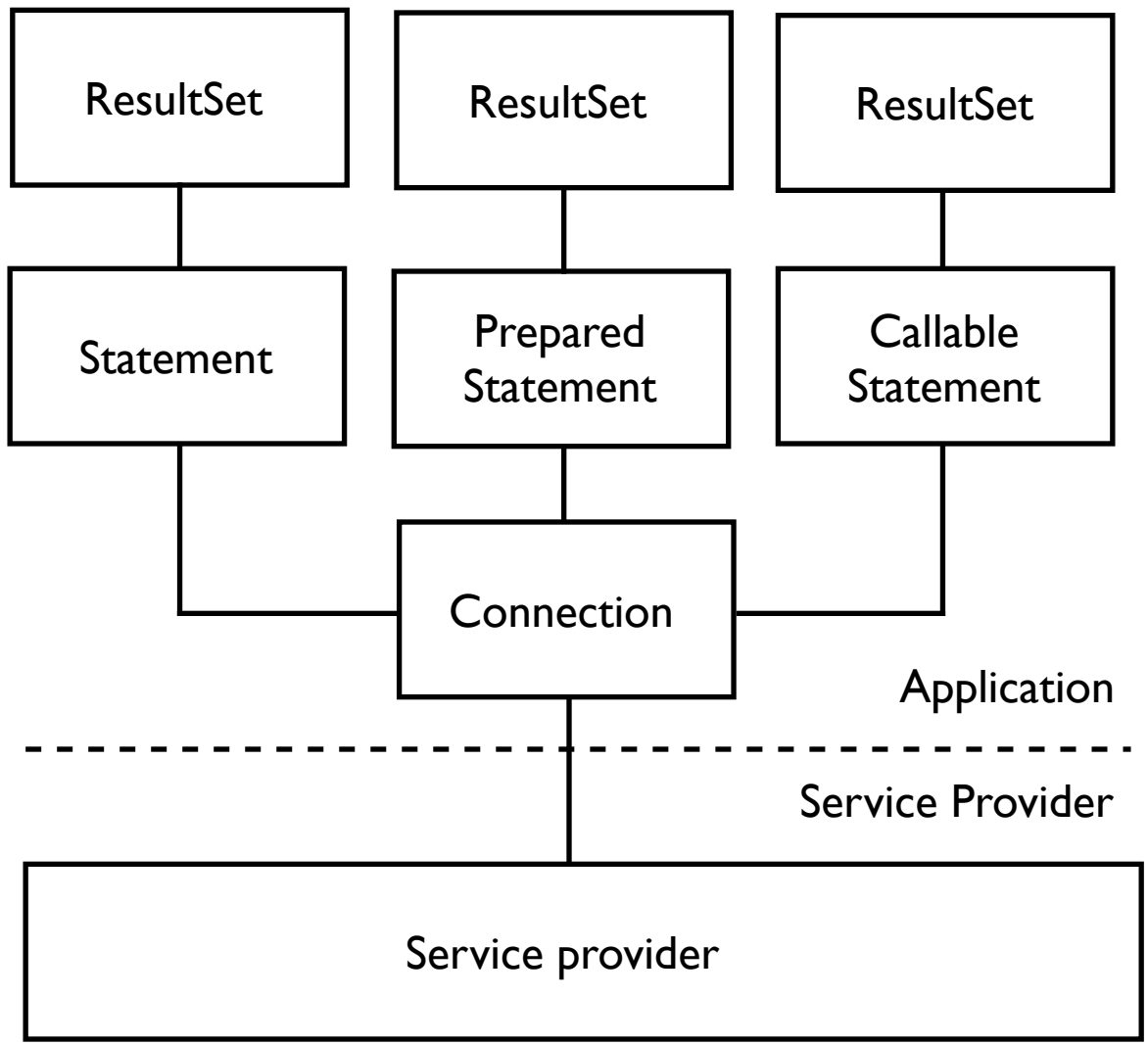
- Naming service
  - context = bindings of names to objects or contexts
  - enumerate subcontexts
- Directory service
  - dircontext = bindings of names to objects with attributes or dircontexts
  - enumerate attributes

- Initial Context = where you start looking for objects / services
- (like X.500 or LDAP)

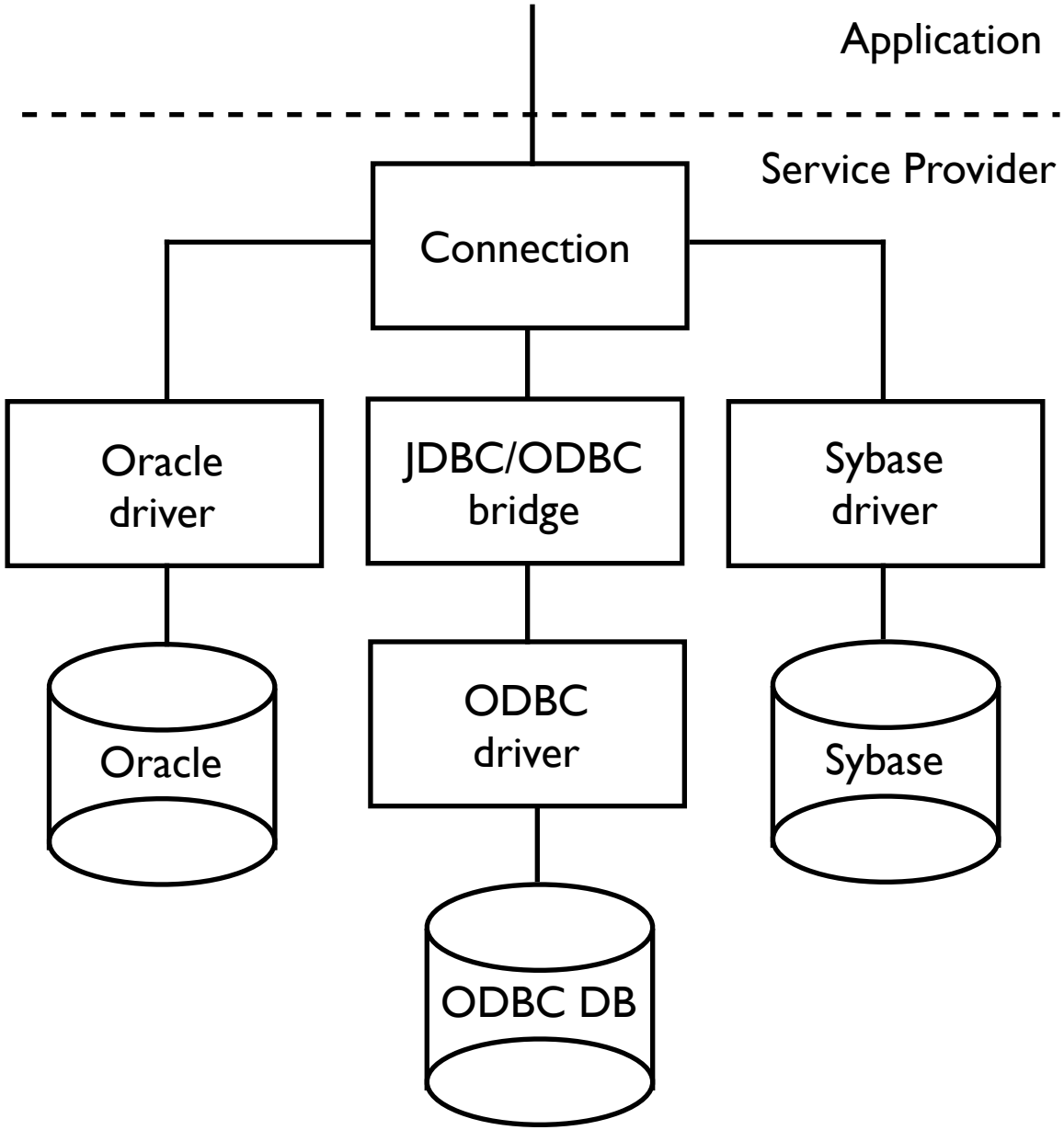
J2EE Application Server Architecture



JDBC Architecture



JDBC Architecture II



Capabilities

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

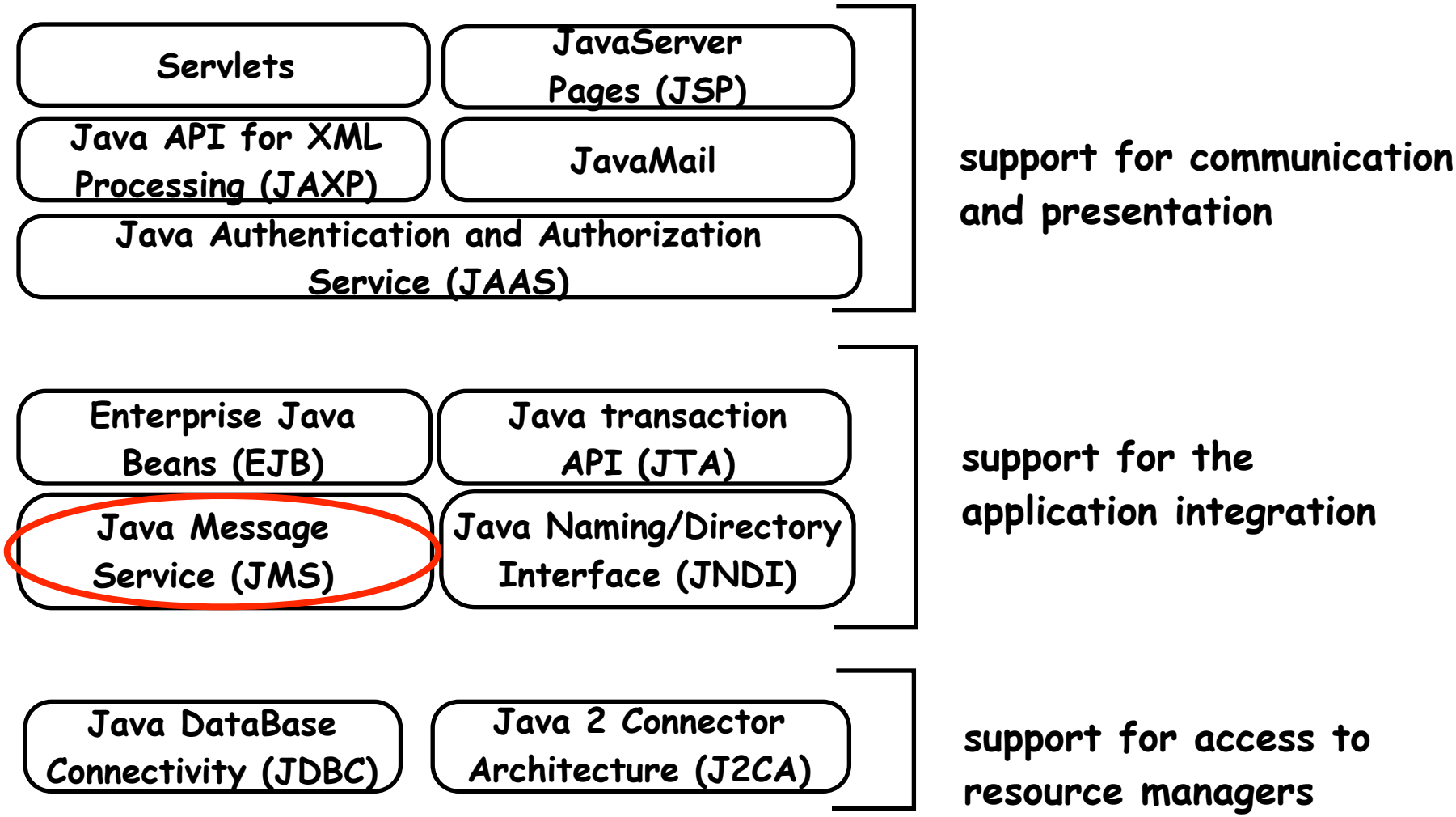
Capabilities

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

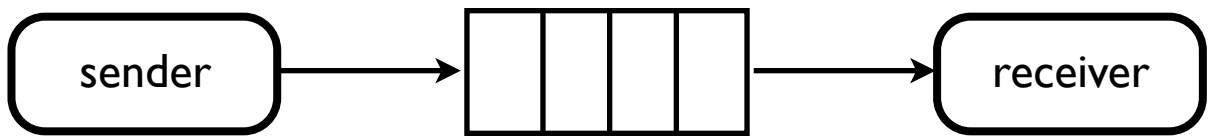
Capabilities

- Distributed transactions
  - if supported by the database and driver

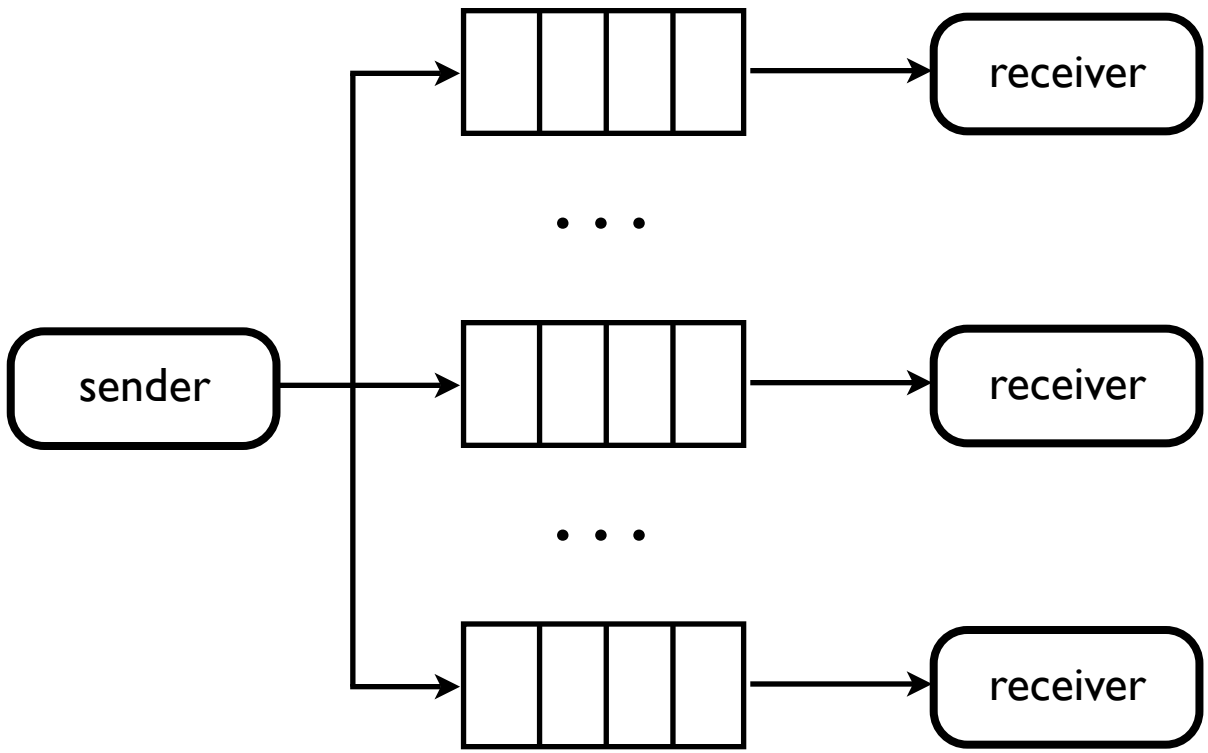
J2EE Application Server Architecture



Point-to-Point



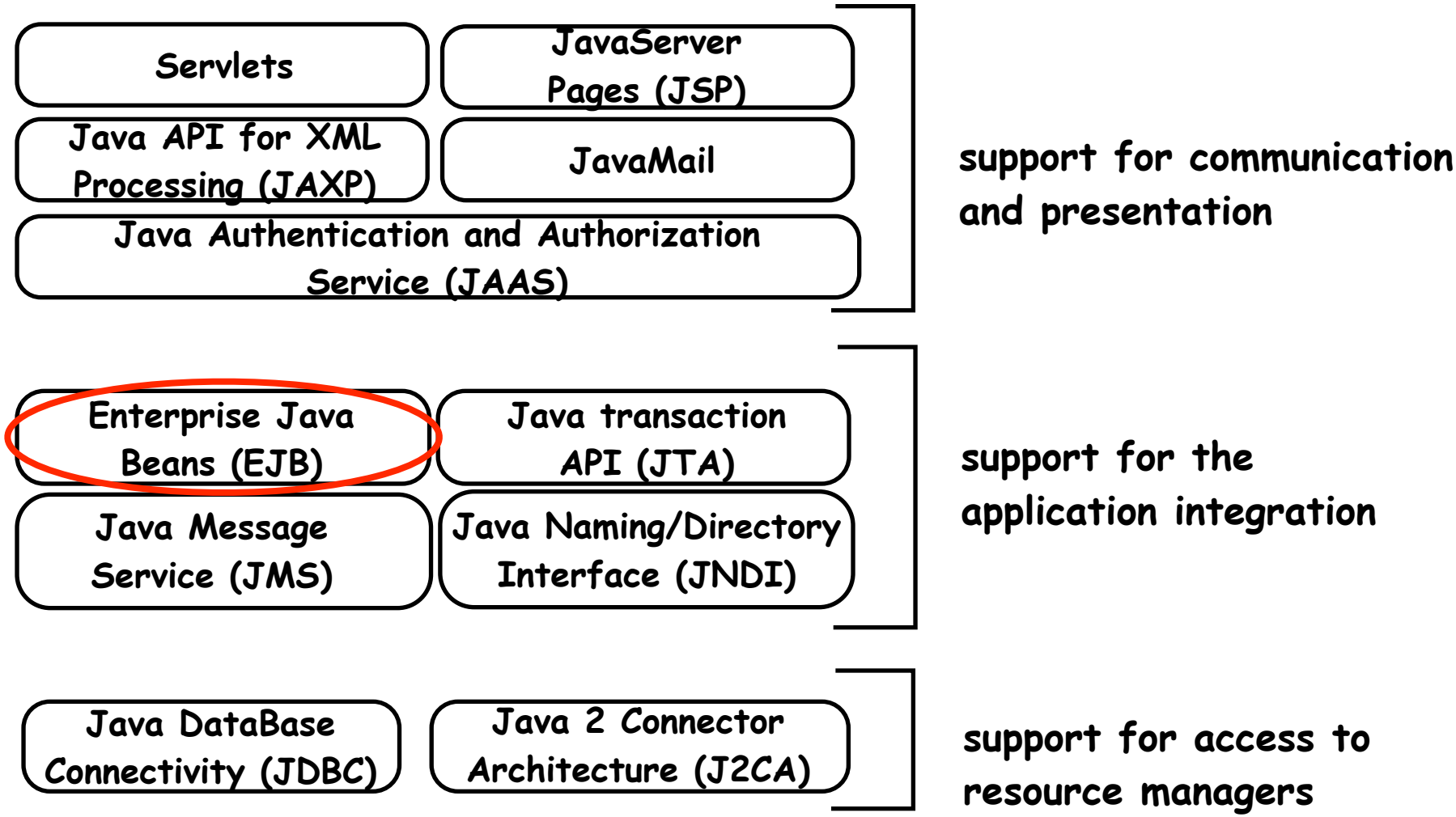
Topic Based (pub/sub)



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



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

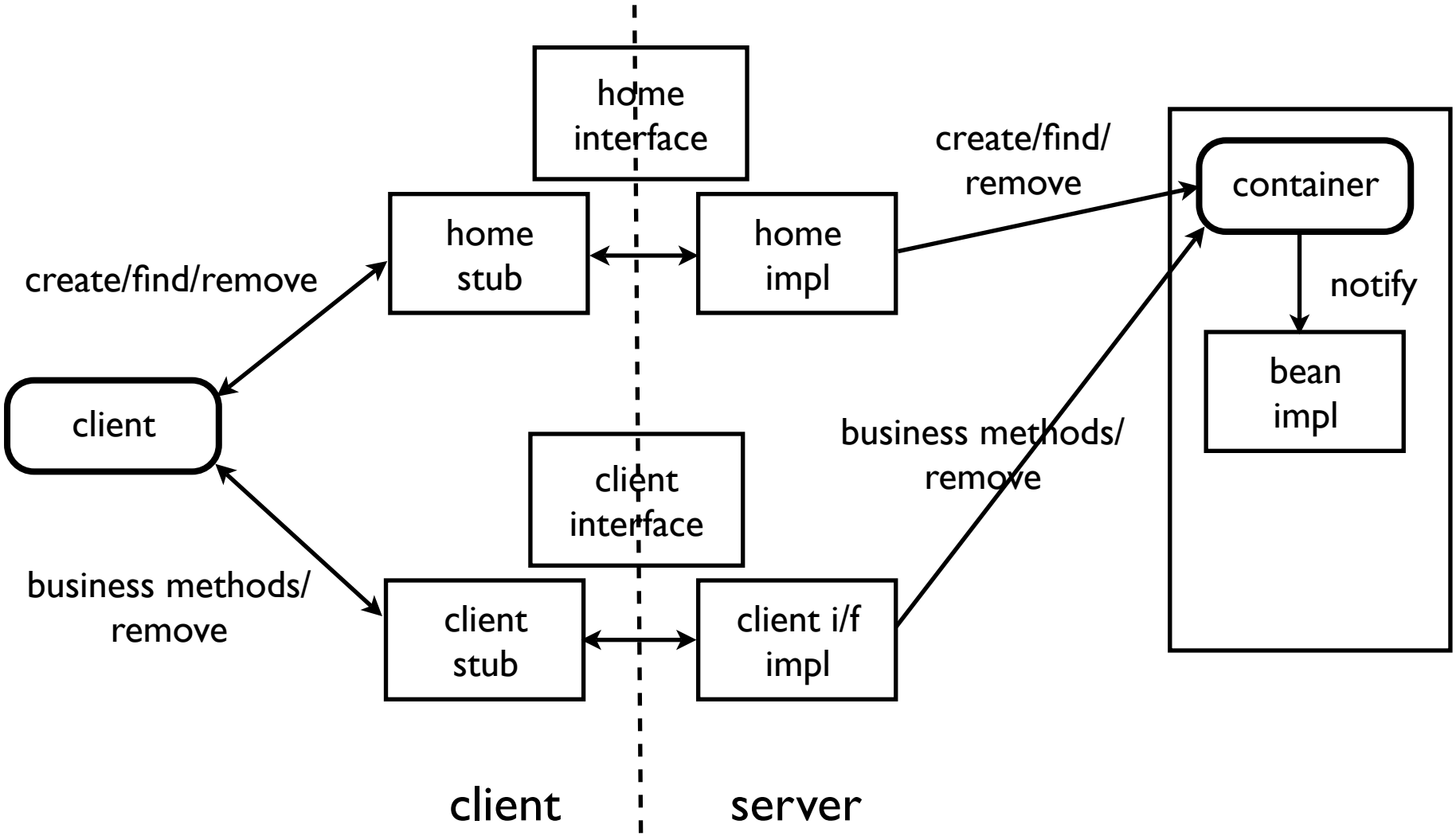
Classes of Beans

- 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



Deployment

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