

## 並列分散システム特論

# Javaによるwebプログラミング 入門

佐藤

## 並列分散システム特論

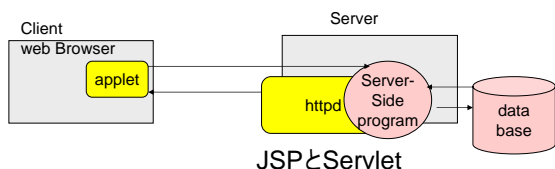
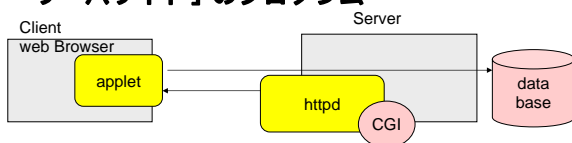
# JavaによるWebプログラミング

- ◆ Applet
- ◆ Servlet – 3層モデル
- ◆ JSP
- ◆ SOAP
- ◆ WebService
- ◆ XML
- ◆ J2EE

## 並列分散システム特論

# 2層モデルから3層モデルへ

### ◆ 「サーバサイド」のプログラム



## 並列分散システム特論

- ◆ システムを変更して、クライアントに影響を及ぼすような変更があった場合にはクライアント側のプログラムを変更しなくてはならない。(アプレットを使えば、改善されるが)

Server-sideのプログラムの変更のみでよい

- ◆ 基幹の大切なデータベースへのアクセスの権限を直接クライアントにももたせなくてはならない。このようなことは特に不特定多数がアクセスする可能性があるインターネットでは問題になる。

Server-sideのプログラムで管理

## 並列分散システム特論

# Servlet

- ◆ webサーバ側にJavaVMを組み込み(組み込みの方法はいろいろある)、サーバ側で実行される
- ◆ 基本的に、Servletで記述されるプログラムはHTMLを生成するプログラム
- ◆ 起動の方法
  - webサーバの起動時にサーバに名前を登録し、その名前で指定する。
  - URLでServletを指定する。
  - <servlet> ...</servlet>のタグで、HTMLファイルの中で指定する。
- ◆ 利点
  - CGIに比べて、プロセスを起動するオーバーヘッドがないため高速となる。
  - Javaを開発言語に使うことができるため、開発効率を向上させることができる。
- ◆ 例: Apache Tomcat

## 並列分散システム特論

# URLパラメータを利用するServlet

```
public class UrlParamterMsg extends GenericServlet {
    public void services(ServletRequest request,
        ServletResponse response)
        throws ServletException, IOException {
        String style = request.getParameter("style");
        response.setContentType("text/html; charset=...");
        PrintWriter pw = response.getWriter();
        pw.println("<html><head>");
        pw.println("<title> ... <title>");
        ....
        pw.println("</html>");
        pw.flush();
        pw.close();
    }
    public String getThisTime(String style) {
        ...
    }
}
```

## URLパラメータを取得するServlet

- ◆ GenericServlet: httpに依存しない場合
- ◆ CGIと同じように、URLのパラメータを取ることができる。

http://host:port/servlet/servlet-name?name=value

- ◆ メソッド javax.servlet.ServletRequest
  - getParameterString(String)
  - getParameterValues(String)

http://host:port/servlet/UrlParameterMsg?style=ja

## HTML FORMの取り扱い

- ◆ HttpServlet: httpに依存したServlet
- ◆ メソッド
  - 最初に呼ばれるメソッド  
doGet(HttpServletRequest request, HttpServletResponse response)
  - POSTで呼ばれるメソッド doPost(HttpServletRequest request, HttpServletResponse response)

## Java Server Pages (JSP)

- ◆ HTMLの中にjavaのコードを埋め込んで、動的にページを作り出す仕組み
- ◆ pageに埋め込まれているjavaのコードから、プログラムを生成し、動的にコンパイルし、servletとして実行することによって実現されている。
  - <% %>に囲まれた間をJavaのプログラムとして実行し、その出力をHTMLの中に埋め込む。
  - <script> </script>で必要なメソッドを定義
- ◆ JavaBeans技術を使うことができる
- ◆ 類似の技術にPHPがある

## JSPの例

```
<script runat="server">
private String getThisTime(String style){ ...}
</script>
<html><head>
<title> ....</title>
...
<%
    out.println("...");
    String style= request.getParamter("style");
    out.println(getThisTime(style));
    out.println("...");
%>
...
</html>
```

## J2EEとEnterprise Java Beans

- ◆ J2EE: Java 2 Platform, Enterprise Edition
- ◆ MI/IOP: 従来のRMIに加えて、RPCのプロトコルとして、CORBA(Common Object Request Broker)の標準プロトコルであるIOP(Internet Inter-ORB Protocol)を使えるようにしたもの。これにより、Javaだけでなく、C/C++などから呼び出したり、その逆も可能になる。
- ◆ JBDC: データベースにアクセスするための標準のJava APIを提供するもの。SQL文(Structured Query Languageの略、リレーショナルデータベースの標準的な言語)を簡単に各種のデータベースに送信できる。メジャーなデータベースである、Sybase, Oracle, IBM DB2などと同じAPIでアクセスすることができる。JBDCのAPIとデータベースを結びつける部分をドライバーといい、ドライバーを使ってデータベースと接続することをコネクションという。

- ◆ JNDI(Java Naming and Directory Interface): ネットワーク上にあるいろいろなネーミングとディレクトリサービスをまとめたもの。RMIのregistryもこの中の1つになっている。そのほかに、標準的なディレクトリサービスであるLDAP(Lightweight Directory Access Protocol)やCORBAのNamingサービスであるCOS (Common Object Service) もSPI(Service Provider Interface)として統合されている。
- ◆ Servlet: httpサーバ側で、Javaプログラムを動かす仕組み。
- ◆ JavaServer Page(JSP): HTML中にJavaのプログラムを記述し、動的なWebページを動かすための仕組み。
- ◆ Enterprise JavaBeans: ネットワーク上で使えるようにしたBeans。
- ◆ Java Transaction: JavaのデータベースなどのtransactionをサポートするAPI
- ◆ XMLライブラリ: JavaでXML文書を扱うためのパッケージ。直接は関係ないが、J2EEの中のいろいろなところで使われている。

## RMI/IIOP

- ◆ COBRAの protocols であるIIOPをつかった RMI の implementation
- ◆ 今ではこちらのほうが主流
- ◆ CORBAのオブジェクトでも呼び出せる
- ◆ プログラミングでの違い
  - UnicastRemoteObjectの代わりに、PortableRemoteObjectを使う。
  - RMI registryでなく、JNDIRegistryをつかう。
  - rmicを使うときに、-iiopのオプションを使う

## SOAP

- ◆ SOAP: Simple Object Access Protocol
- ◆ XMLを使ったRPCのための protocols
  - HTTPで、カプセル化できる。
  - 送信側は、URLに対するPOSTメッセージ
  - 受信側は、HTTPからのメッセージ
  - URLを指定して、RPCを行う
- ◆ クライアントのライブラリ
  - org.apache.soap.\*

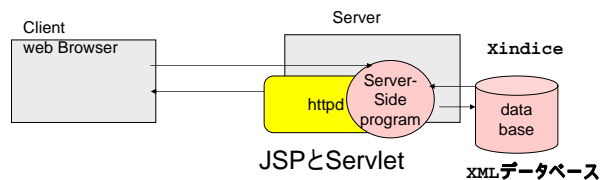
## SOAPを使ったRPCの例

```
public class GetPassword {
    public static main(String args[]){
        String urlString =
            "http://localhost/soap/servlet/rpcrouter";
        Call c = new Call();
        c.setTargetObjectURI("urn:userinfoservice");
        c.setMethodName("getPassword");
        c.setEncodingStyleURI(...);
        Vector v = new Vector();
        v.addElement(...);
        c.setParams(v);

        ...
        r = c.invoke(new URL(urlString), "");
        Parameter result = r.getReturnValue();
    }
}
```

## webプログラミングの実際

- ◆ Xindiceの付録についていたAddressbookを例に
  - Xindice: native XML データベース
  - servletからアクセス
  - User interfaceは、JSPで書いてある。



## tomcatとは

- ◆ Tomcat は、Java Servlet 2.2 および JavaServer Page 1.1 技術の正式なリファレンス実装
- ◆ Tomcatは、Apacheライセンスに基づいたオープンで誰でも参加できる環境下で、多くの優秀な開発者が協力して開発
  - <http://www.jajakarta.org/>
- ◆ All javaの実装
- ◆ apacheにも組み込める

## tomcatでのアプリケーションのdeploy

- ◆ アプリケーションは、\$(TOMCAT\_HOME)/webappsにおく。
  - \$(TOMCAT\_HOME)/webapps/xyzはhttp:<address>/xyzで参照される
- ◆ webapps/Addressbookには
  - 直下には、html, jspなど
    - 最初はindex.{html,jsp}が起動される
  - WEB-INFの下がservletのプログラム
    - web.xml servletの環境設定
    - classes servletのプログラムをおく
    - lib servletを動かすためのライブラリ

## index.jsp、header.jsp

- ◆ はじめに起動されるページ
- ◆ <%@ include ...%>で、header.jspとfooter.jspをinclude
  - これは、枠を表示
- ◆ 実際は、header.jspにあるメニューから飛ぶ
  - メニュー " List Contact"は、  
<A HREF=  
"/Addressbook/servlet/Task?action=listcontacts">  
List Contacts</A>
- ◆ classesの中にあるservletにアクセスするには  
/Addressbook/servlet/<クラス名>?パラメータ  
でアクセスする

## Task.java

- ◆ servletのメインプログラム
  - 通常、このURIが参照された場合にはGETメッセージになるため、doGetは呼ばれる
  - doPostは、HTMLのフォームから呼ばれた場合
    - addContact.jsp
  - doPostとdoGetは同じにして、パラメータとしてわたされるactionで区別している
    - request.getParameter("action");
- ◆ 最初に、beanを作る
  - このアプリケーションでは、groupとpersonというbeanをつけている (header.jsp)
  - JSPとservletで共有できる " 変数 "

## task.java

- ◆ "action"パラメータで、分岐して
  - gotoPage("JSPのページ",request, response);
  - 例えば、addformの場合、/addContact.jsp
- ◆ エラーの場合は、error.jsp ?error=... で、response.sendRedirectする。

## データベースのアクセス

```

package org.apache.xindice.examples;

import org.xmldb.api.base.*;
import org.xmldb.api.modules.*;
import org.xmldb.api.*;

public class Example1 {
    public static void main(String[] args) throws Exception {
        Collection col = null;
        try {
            String driver =
                "org.apache.xindice.client.xmldb.DatabaseImpl";
            Class c = Class.forName(driver);
            Database database = (Database) c.newInstance();
            DatabaseManager.registerDatabase(database);
            col =
                DatabaseManager.getCollection("xmldb:xindice:///db/addressl
            String xpath = "//person[fname='John']";

```

## データベースのアクセス

```

col = DatabaseManager.getCollection(
    "xmldb:xindice:///db/addressbook");
String xpath = "//person[fname='John']";
XPathQueryService service
(XPathQueryService) col.getService("XPathQueryService",
ResourceSet resultSet = service.query(xpath);
ResourceIterator results = resultSet.getIterator();
while (results.hasMoreResources()) {
    Resource res = results.nextResource();
    System.out.println((String) res.getContent());
}
}
catch (XMLDBException e) {
    System.err.println("XML:DB Exception occured " + e.
}
finally {
    if (col != null) {
        col.close();

```

## データベースについて

- ◆ データベース Database
- ◆ コレクション Collection
- ◆ サービス XPathQueryService
- ◆ リソースセット ResourceSet
- ◆ リソースIterator

並列分散システム特論

## addContact.jsp

---

- ◆ 名前の入力のためのFORM
  - POSTで、送る
  - そのときに、ACTION=ADDCONTACT
  - それを/Addressbook/servlet/Taskに送る。

並列分散システム特論

## addContact.java

---

- ◆ HttpServletRequest, HttpServletResponseを引数として、requestからgetParameterで、入力を作る
- ◆ Actionをextendして作られている

並列分散システム特論

## Action.java

---

- ◆ 他のactionのベースクラス
  - AddContact.java
  - DeleteContact.java
  - ...
- ◆ getCollection
  - 初期化時に作られるDBconnectionから、collectionを返す
  - DBConnectionは、もしもDBとつながっていなかったらつなげる

並列分散システム特論

## Group, Person

---

- ◆ Person.java Servlet上の一人分のデータ
- ◆ Group.java Servlet上のPersonデータの列

並列分散システム特論

## ListContact.java、listContact.jsp

---

- ◆ /personでXPathQueryをして、groupを作る。
- ◆ listContact.jsp
  - ListContact.javaが呼ばれてから、表示
  - groupというbeansを共有しながら、表示している。