Если обратиться к этой странице, передавая в строке запроса параметры (например **`http://localhost:8082/FirstJSP/forward.jspx?name=UserName`**), то, кроме этих параметров, странице **`param.jspx`** будет передан параметр **`addparam`** со значением **`added`**.

Технология взаимодействия JSP и сервлета

В большинстве приложений используются не сервлеты или JSP, а их сочетание. В JSP представляется, как будут выглядеть результаты запроса, а сервлет отвечает за вызов классов бизнес-логики и передачу результатов выполнения бизнес-логики в соответствующие JSP и их вызов. Т.е. сервлеты не генерируют ответа сами, а только выступают в роли контроллера запросов. Такая архитектура построения приложений носит название MVC (Model/View/Controller). Model – классы бизнес-логики и длительного хранения, View – страницы JSP, Controller – сервлет.

Реализацию достаточно простой, но эффективной технологии построения распределенного приложения можно рассмотреть на примере решения задачи проверки логина и пароля пользователя с выводом приветствия в случае положительного результата. Схематично организацию данного приложения можно представить в виде следующей диаграммы:

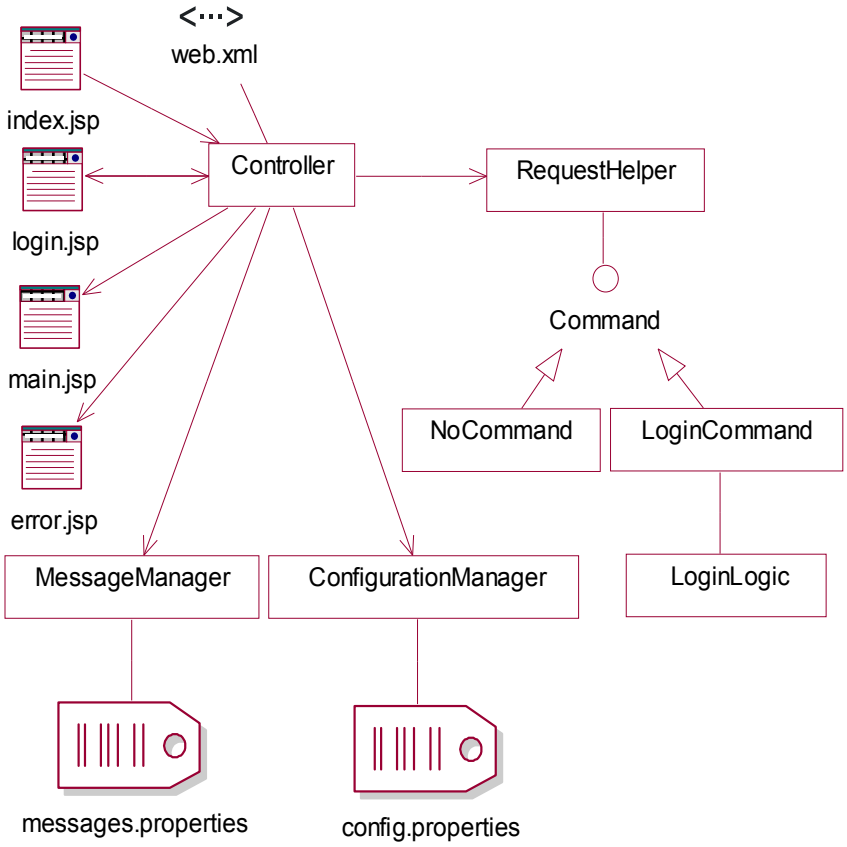

Рис. 19.2. Диаграмма взаимодействия классов и страниц JSP приложения.

```
<!--пример # 24 : прямой вызов контроллера : index.jspx -->
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0">
<jsp:directive.page contentType="text/html; charset=Utf-8"
/>
<html><head><title>Index JSP</title></head>
<body>
<a href="controller">Main Controller</a>
</body></html>
</jsp:root>
```

Следующая страница **login.jsp** содержит форму для ввода логина и пароля для аутентификации в системе:

```
<!--пример # 25 : форма ввода информации и вызов контроллера : login.jsp -->
<%@ page language="java" contentType="text/html;
  charset=ISO-8859-1"  pageEncoding="ISO-8859-1"%>
    <html><head><title>Login</title></head>
    <body><h3>Login</h3>
    <hr/>
    <form name="loginForm" method="POST"
    action="controller">
    <input type="hidden" name="command" value="login" />
        Login:<br/>
        <input type="text" name="login" value=""><br/>
        Password:<br/>
        <input type="password" name="password" value="">
      <br/>
        <input type="submit" value="Enter">
    </form><hr/>
    </body></html>
```

Код сервлета-контроллера **Controller**:

```
/* пример # 26 : контроллер запросов : Controller.java */
package by.bsu.famcs.jspservlet;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import by.bsu.famcs.jspservlet.commands.Command;
import by.bsu.famcs.jspservlet.manager.MessageManager;
import
by.bsu.famcs.jspservlet.manager.ConfigurationManager;

public class Controller extends HttpServlet
            implements  javax.servlet.Servlet {
//объект, содержащий список возможных команд
      RequestHelper requestHelper =
            RequestHelper.getInstance();
```

```
      public Controller() {
            super();
      }
      protected void doGet(HttpServletRequest request,
                  HttpServletResponse response)
            throws ServletException, IOException{
            processRequest(request, response);
      }
      protected void doPost(HttpServletRequest request,
                  HttpServletResponse response)
            throws ServletException, IOException{
            processRequest(request, response);
      }
      private void processRequest(HttpServletRequest
            request, HttpServletResponse response)
      throws ServletException, IOException {

            String page = null;
            try {
//определение команды, пришедшей из JSP
                  Command command =
                        requestHelper.getCommand(request);
/*вызов реализованного метода execute() интерфейса Command и передача
параметров классу-обработчику конкретной команды*/
            page = command.execute(request, response);
// метод возвращает страницу ответа
            } catch (ServletException e) {
                  e.printStackTrace();
//генерация сообщения об ошибке
request.setAttribute("errorMessage",
      MessageManager.getInstance().getProperty(
      MessageManager.SERVLET_EXCEPTION_ERROR_MESSAGE));
//вызов JSP-страницы с сообщением об ошибке
page = ConfigurationManager.getInstance()
      .getProperty(ConfigurationManager.ERROR_PAGE_PATH);
            } catch (IOException e) {
                  e.printStackTrace();
                  request.setAttribute("errorMessage",
 MessageManager.getInstance()
.getProperty(MessageManager.IO_EXCEPTION_ERROR_MESSAGE));

            page = ConfigurationManager.getInstance()
.getProperty(ConfigurationManager.ERROR_PAGE_PATH);
            }
//вызов страницы ответа на запрос
            RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(page);
            dispatcher.forward(request, response);
      }
}
```

```
/* пример # 27 : класс контейнер команд : RequestHelper.java */
package by.bsu.famcs.jspservlet;
import java.util.HashMap;
import javax.servlet.http.HttpServletRequest;
import by.bsu.famcs.jspservlet.commands.Command;
import by.bsu.famcs.jspservlet.commands.LoginCommand;
import by.bsu.famcs.jspservlet.commands.NoCommand;

public class RequestHelper {
      private static RequestHelper instance = null;

      HashMap<String, Command> commands =
                  new HashMap<String, Command>();

      private RequestHelper() {
//заполнение таблицы командами
            commands.put("login", new LoginCommand());
      }
   public Command getCommand(HttpServletRequest request) {
//извлечение команды из запроса
      String action = request.getParameter("command");
//получение объекта, соответствующего команде
      Command command = commands.get(action);
            if (command == null) {
//если команды не существует в текущем объекте
                  command = new NoCommand();
            }
            return command;
      }
//создание единственного объекта по шаблону Singleton
   public static RequestHelper getInstance() {
            if (instance == null) {
                  instance = new RequestHelper();
            }
            return instance;
      }
}
/* пример # 28 : интерфейс, определяющий контракт и его реализации :
Command.java: LoginCommand.java : NoCommand.java */
package by.bsu.famcs.jspservlet.commands;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;

public interface Command {
      public String execute(HttpServletRequest request,
                              HttpServletResponse response)
                  throws ServletException, IOException;
}
```

```
package by.bsu.famcs.jspservlet.commands;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import by.bsu.famcs.jspservlet.logic.LoginLogic;
import
by.bsu.famcs.jspservlet.manager.ConfigurationManager;
import by.bsu.famcs.jspservlet.manager.MessageManager;

public class LoginCommand implements Command {

private static final String PARAM_NAME_LOGIN = "login";
private static final String PARAM_NAME_PASSWORD
                                        = "password";

      public String execute(HttpServletRequest request,
                              HttpServletResponse response)
                  throws ServletException, IOException {

            String page = null;
//извлечение из запроса логина и пароля
String login = request.getParameter(PARAM_NAME_LOGIN);
String pass = request.getParameter(PARAM_NAME_PASSWORD);
//проверка логина и пароля
            if (LoginLogic.checkLogin(login, pass)) {
                  request.setAttribute("user", login);
//определение пути к main.jsp
                  page = ConfigurationManager.getInstance()
.getProperty(ConfigurationManager.MAIN_PAGE_PATH);
            } else {
                  request.setAttribute("errorMessage",
 MessageManager.getInstance()
.getProperty(MessageManager.LOGIN_ERROR_MESSAGE));
                  page = ConfigurationManager.getInstance()
.getProperty(ConfigurationManager.ERROR_PAGE_PATH);
            }
            return page;
      }
}
package by.bsu.famcs.jspservlet.commands;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import
by.bsu.famcs.jspservlet.manager.ConfigurationManager;
```

```
public class NoCommand implements Command {

      public String execute(HttpServletRequest request,
             HttpServletResponse response)
             throws ServletException, IOException {
/*в случае прямого обращения к контроллеру переадресация на страницу ввода
логина*/
      String page = ConfigurationManager.getInstance()
.getProperty(ConfigurationManager.LOGIN_PAGE_PATH);
            return page;
      }
}
/* пример # 29 : служебные классы, извлекающие из properties-файлов
необходимую для функционирования приложения информацию :
ConfigurationManager.java: MessageManager.java */
package by.bsu.famcs.jspservlet.manager;
import java.util.ResourceBundle;

public class ConfigurationManager {
      private static ConfigurationManager instance;
      private ResourceBundle resourceBundle;

//класс извлекает информацию из файла config.properties
      private static final String BUNDLE_NAME = "config";

      public static final String DATABASE_DRIVER_NAME =
                                    "DATABASE_DRIVER_NAME";
      public static final String DATABASE_URL =
                                          "DATABASE_URL";
      public static final String ERROR_PAGE_PATH =
                                    "ERROR_PAGE_PATH";
      public static final String LOGIN_PAGE_PATH =
                                    "LOGIN_PAGE_PATH";
      public static final String MAIN_PAGE_PATH =
                                    "MAIN_PAGE_PATH";

      public static ConfigurationManager getInstance() {
            if (instance == null) {
                  instance = new ConfigurationManager();
                  instance.resourceBundle =
                  ResourceBundle.getBundle(BUNDLE_NAME);
            }
            return instance;
      }
      public String getProperty(String key) {
            return (String)resourceBundle.getObject(key);
      }
}
```

```
package by.bsu.famcs.jspservlet.manager;
import java.util.ResourceBundle;
public class MessageManager {
      private static MessageManager instance;
      private ResourceBundle resourceBundle;
//класс извлекает информацию из файла messages. properties
      private static final String BUNDLE_NAME = "messages";
      public static final String LOGIN_ERROR_MESSAGE = "LO-
GIN_ERROR_MESSAGE";
      public static final String SERV-
LET_EXCEPTION_ERROR_MESSAGE =
            "SERVLET_EXCEPTION_ERROR_MESSAGE";
      public static final String IO_EXCEPTION_ERROR_MESSAGE
= "IO_EXCEPTION_ERROR_MESSAGE";

      public static MessageManager getInstance() {
            if (instance == null) {
                  instance = new MessageManager();
                  instance.resourceBundle =
                  ResourceBundle.getBundle(BUNDLE_NAME);
            }
            return instance;
      }

      public String getProperty(String key) {
            return (String)resourceBundle.getObject(key);
      }
}
```

Далее приведено содержимое файла *config.properties:*

```
################################
## Application configuration ##
################################
DATABASE_DRIVER_NAME=com.mysql.jdbc.Driver
DATABASE_URL=jdbc:mysql://localhost:3306/db1?user=
root&password=root
ERROR_PAGE_PATH=/jsp/error.jspx
LOGIN_PAGE_PATH=/jsp/login.jspx
MAIN_PAGE_PATH=/jsp/main.jspx
```

Далее приведено содержимое файла *messages.properties:*

```
################################
##         Messages      ##
################################
LOGIN_ERROR_MESSAGE=Incorrect login or password
SERVLET_EXCEPTION_ERROR_MESSAGE=ServletException: Servlet
encounters difficulty
IO_EXCEPTION_ERROR_MESSAGE=IOException: input or output er-
ror while handling the request
```

Ниже приведен код класса бизнес-логики **LoginLogic**, выполняющий проверку правильности введенных логина и пароля с помощью запроса в БД:

```
/* пример # 30 : бизнес-класс проверки данных пользователя : LoginLogic.java */
package by.bsu.famcs.jspservlet.logic;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import
by.bsu.famcs.jspservlet.manager.ConfigurationManager;

public class LoginLogic {
      public static boolean checkLogin(
            String login, String password) {
            // проверка логина и пароля
            try {
//организация простейшего соединения с базой данных
String driver = ConfigurationManager.getInstance()
.getProperty(ConfigurationManager.DATABASE_DRIVER_NAME);

                  Class.forName(driver);
                  Connection cn = null;
                  try {
String url = ConfigurationManager.getInstance()
.getProperty(ConfigurationManager.DATABASE_URL);
            cn = DriverManager.getConnection(url);
                        PreparedStatement st = null;
                        try {
                  st = cn.prepareStatement(
"SELECT * FROM USERS WHERE LOGIN = ? AND PASSWORD = ?");
                              st.setString(1, login);
                              st.setString(2, password);
                              ResultSet rs = null;
                              try {
                                    rs = st.executeQuery();
      /* проверка, существует ли пользователь
       с указанным логином и паролем */
                                    return rs.next();
                              } finally {
                                    if (rs != null)
                                          rs.close();
                              }
                        } finally {
                              if (st != null)
                                    st.close();
                        }
                  } finally {
```

```
                    if (cn != null)
                        cn.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
                return false;
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
                return false;
            }
        }
}
```

Страница **main.jsp** показывается пользователю в случае успешной аутентификации в приложении:

```
<!--пример # 31 : сообщение о входе : main.jsp -->
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns:c=http://java.sun.com/jsp/jstl/core
      version="2.0">
<jsp:directive.page contentType="text/html;
      charset=Utf-8" />
      <html><head><title>Welcome</title></head>
      <body><h3>Welcome</h3>
      <hr />
      <c:out value="${user}, Hello!"/>
      <hr />
      <a href="controller">Return to login page</a>
      </body></html>
</jsp:root>
```

Страница **error.jsp** загружается пользователю в случае возникновения ошибок (например, если неправильно введены логин и пароль):

```
<!-- пример # 32 : страница ошибок, предлагающая повторить процедуру ввода
информации : error.jspx -->
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns:c=
      "http://java.sun.com/jsp/jstl/core" version="2.0">
      <jsp:directive.page contentType=
                  "text/html; charset=Utf-8" />
      <html><head><title>Error</title></head>
      <body>
      <h3>Error</h3>
      <hr />
      <jsp:expression>
(request.getAttribute("errorMessage") != null)
? (String) request.getAttribute("errorMessage")
: "unknown error"</jsp:expression>
      <hr />
      <a href="controller">Return to login page</a>
```

```
      </body></html>
</jsp:root>
```

И последнее, что надо сделать в приложении, – это настроить файл **web.xml**, чтобы можно было обращаться к сервлету-контроллеру по имени **controller**, т.е. необходимо настроить mapping.

```
<!--пример # 33 : имя сервлета и путь к нему : web.xml -->
   <?xml version="1.0" encoding="UTF-8"?>
   <web-app id="WebApp_ID" version="2.4" >
      <display-name>Project</display-name>
      <servlet>
            <description>
            </description>
            <display-name>
            Controller</display-name>
            <servlet-name>Controller</servlet-name>
            <servlet-class>
      by.bsu.famcs.jspservlet.Controller</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>Controller</servlet-name>
            <url-pattern>/controller</url-pattern>
      </servlet-mapping>
      <welcome-file-list>
            <welcome-file>index.jsp</welcome-file>
      </welcome-file-list>
   </web-app>
```

В данном случае в поле **<servlet-name>** было занесено имя **controller,** а в поле **<url-pattern>** – соответственно **/controller**.

Запуск примера производится из командной строки Web-браузера при запущенном контейнере сервлетов Tomcat 5.5.*, например в виде:

http://localhost:8082/Project/index.jspx

В этом случае при вызове сервлета в браузере будет отображен путь и имя в виде:

http://localhost:8082/Project/controller

Задания к главе 19

Вариант А

Реализовать приложение, используя технологию взаимодействия JSP и сервлетов. Вся информация должна храниться в базе данных.

1. **Банк**. Осуществить перевод денег с одного счета на другой с указанием реквизитов: Банк, Номер счета, Тип счета, Сумма. Таблицы должны находиться в различных базах данных. Подтверждение о выполнении операции должно выводиться в JSP с указанием суммы и времени перевода.
2. **Регистрация пользователя**. Должны быть заполнены поля: Имя, Фамилия, Дата рождения, Телефон, Город, Адрес. Система должна при-