

Лабораторная работа №4

Сервлеты и JSP-страницы

Цель

Изучить возможности Servlet API и технологии JavaServer Pages для разработки веб-приложений и получить навыки реализации сервлетов и JSP-страниц для обработки запросов пользователей.

Задание

1. Разработать веб-приложение, содержащее один сервлет, выполняющий функции контроллера, и две JSP-страницы: одна страница должна обеспечивать вывод таблицы с данными, вторая – добавление или редактирование данных.
2. Организовать взаимодействие сервлета-контроллера с базой данных через класс DAO.
3. Организовать взаимодействие сервлета с JSP-страницами с помощью атрибутов запроса и диспетчера запросов.
4. Использовать в JSP-страницах скриптовые элементы всех видов и встроенные объекты.
5. Организовать ввод данных пользователя в виде HTML-формы.

Порядок выполнения работы

1. Создайте в среде разработки NetBeans новый проект типа **Web / Web Application** (рис. 1). Введите в поле **Project Name** название проекта, которое по умолчанию определяет также контекст веб-приложения (поле Context Path). Остальные настройки рекомендуется выполнить так, как показано на рис. 1. Нажмите кнопку **Finish**.
2. Обследуйте структуру созданного проекта в окнах **Projects** и **File**, определите ее отличия от структуры standalone-приложения.
3. Скопируйте классы модели данных и класс(ы) DAO из проекта для ЛР3 в проект веб-приложения. Можно скопировать в буфер целиком пакет с нужными классами и затем вставить его из буфера в узел **Source Packages** проекта веб-приложения.
4. Измените способ подключения к базе данных в классе DAO – в рамках сервера приложений вместо менеджера JDBC-драйверов следует использовать источник данных. Так, вместо строки

```
conn = DriverManager.getConnection("jdbc:mysql://localhost/rsubd", "root", "1234");
```

нужно использовать следующий фрагмент кода:

```
InitialContext ctx = new InitialContext();  
DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/clients");  
Connection conn = ds.getConnection();
```

Первые две строки получают источник данных (объект типа `javax.sql.DataSource`) из окружения веб-компонента. Это позволяет изменять используемую базу данных без изменения исходного кода приложения, то есть обеспечивает переносимость приложения. Однако создание источника данных, связанного с указанным именем (в примере – `jdbc/clients`), выполняется специфичным для конкретного сервера приложений способом.

Внимание: Если класс DAO использует источник данных, устанавливаемый извне, то

изменения в него вносить не нужно, см. п.7. Данную ситуацию далее будем называть инъекцией зависимости.

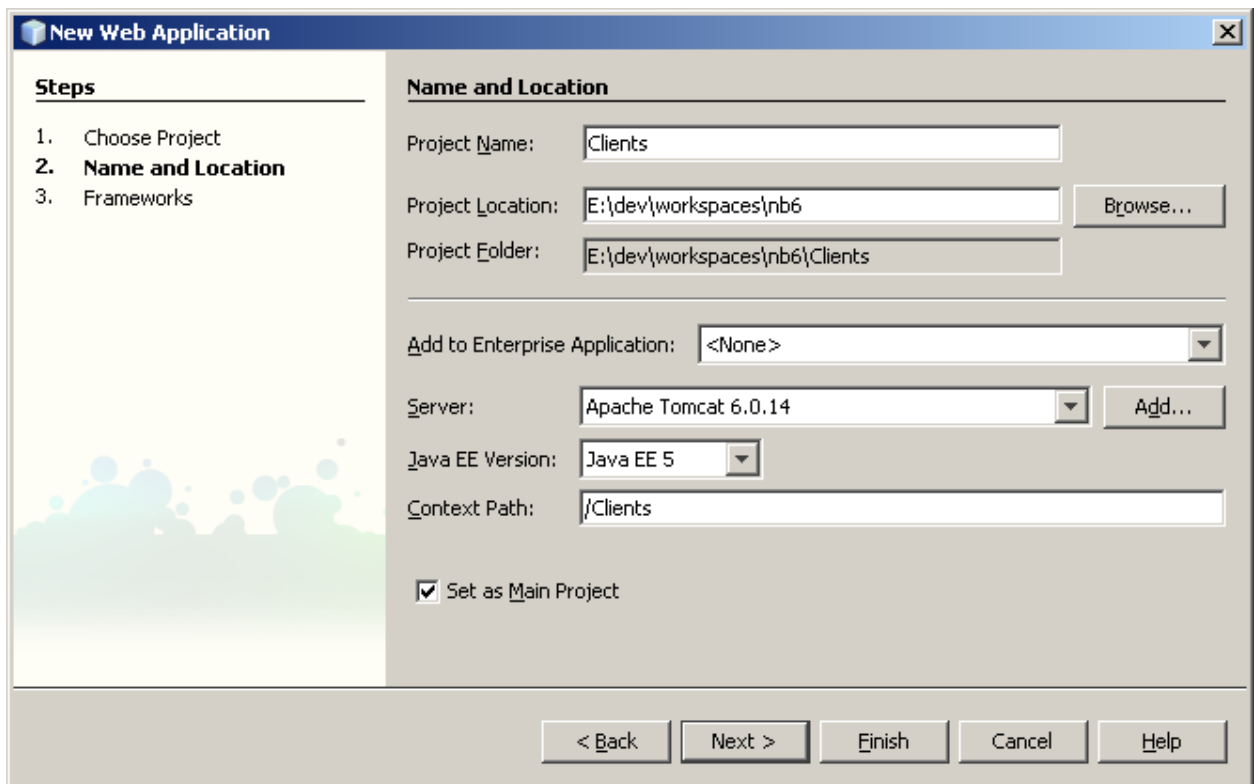


Рис. 1. Настройка нового проекта веб-приложения

5. Создайте в проекте новый сервлет (файл типа **Web / Servlet**). На первом шаге мастера создания сервлета (рис. 2) укажите имя класса сервлета (поле **Class Name**) и пакет, в котором он будет располагаться (поле **Package**). Второй шаг мастера создания сервлета (рис. 3) позволяет добавить в установочный дескриптор следующие сведения о сервлете: имя сервлета (поле **Servlet Name**), псевдонимы сервлета (поле **URL Pattern(s)**) и параметры инициализации. Мастер автоматически формирует имя сервлета и псевдоним по имени класса сервлета. Нажмите кнопку **Finish**.

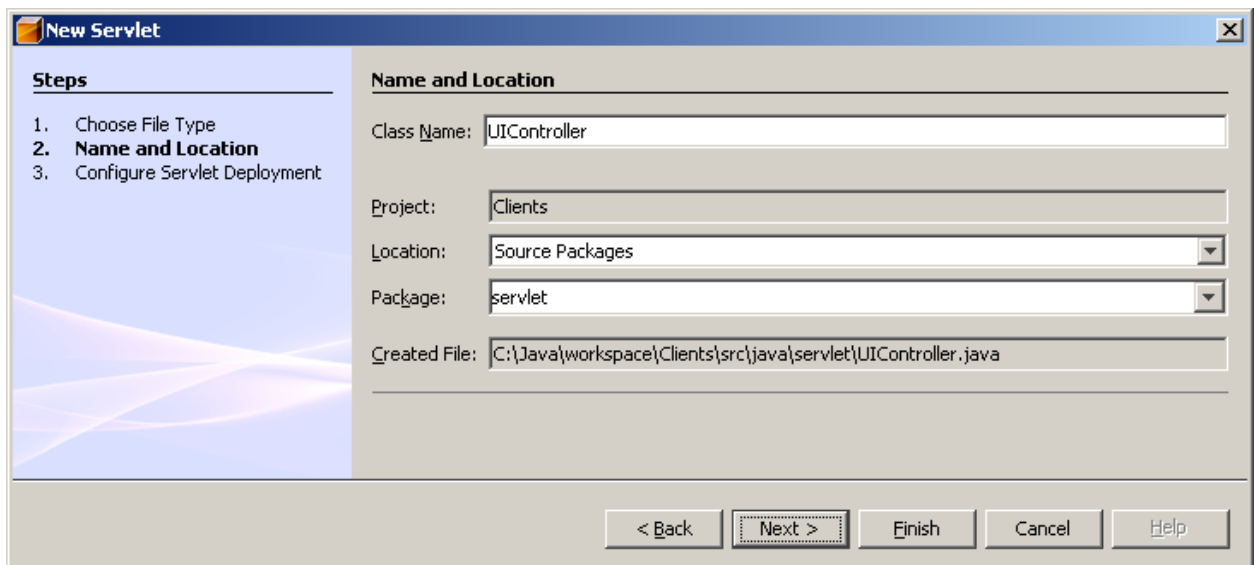


Рис. 2. Создание сервлета: шаг 1

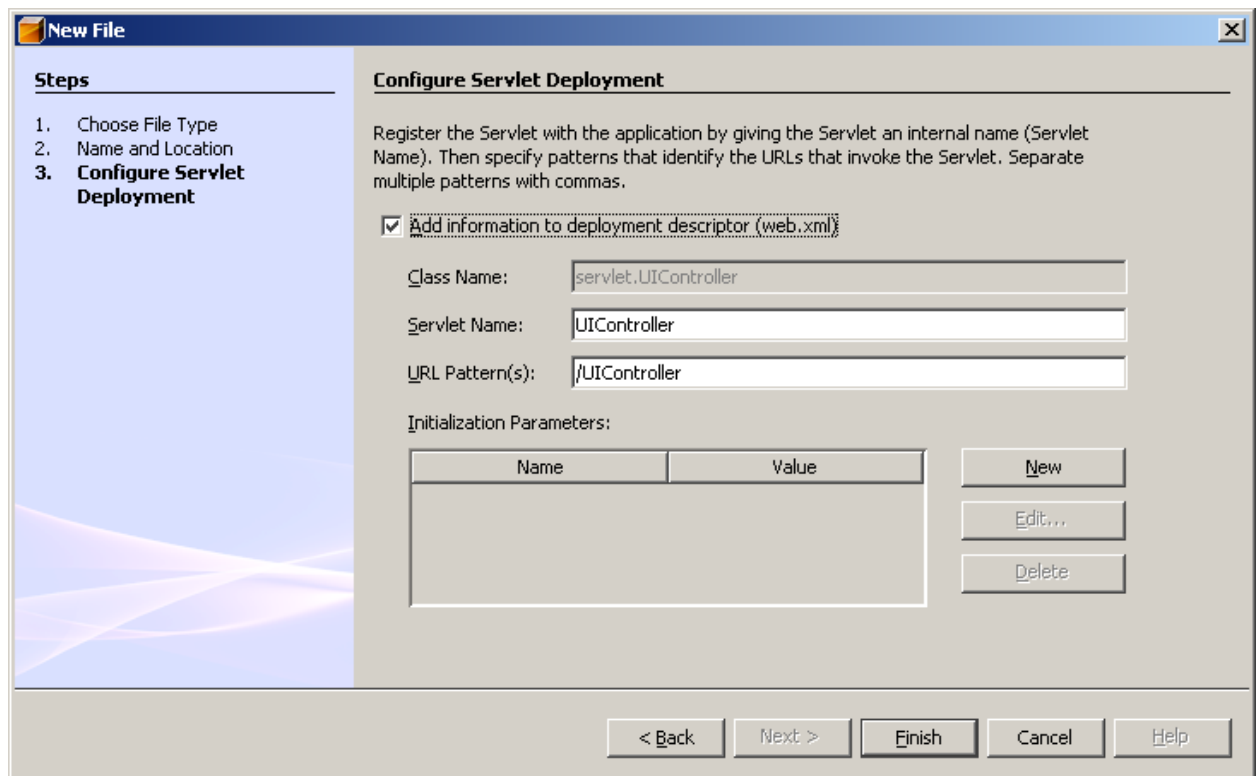


Рис. 3. Создание сервлета: шаг 2

6. В рамках метода `processRequest()` созданного сервлета реализуйте выборку параметров запроса, обращение к классу DAO для получения или изменения данных и перенаправление запроса на соответствующую JSP-страницу.
7. Данный пункт выполняется, если для класса DAO используется инъекция зависимости. В ЛРЗ установку источника данных для DAO-класса выполнял тестовый клиент, а в текущей работе это будет делать сервлет. Сервлет может получить источник данных от контейнера одним из двух способов. Первый из них описан в п.4 и заключается в явной инициализации JNDI-контекста и получении источника данных по определенному JNDI-имени. Второй способ подразумевает неявное выполнение тех же действий за счет использования аннотации `@Resource`. Для этого в классе сервлета нужно объявить поле, аналогичное следующему:


```
@Resource(name = "jdbc/clients")
private DataSource ds;
```

 В данном примере `jdbc/clients` – это JNDI-имя в окружении веб-компонента (`java:comp/env`), которому соответствует объект источника данных. Значение данного поля устанавливается контейнером после вызова конструктора по умолчанию для создания экземпляра сервлета и ДО вызова метода `init()`. Соответственно, полученный таким образом источник данных можно использовать во всех методах сервлета.
8. Проведите настройку веб-сервера для работы с базой данных.
 - 1) Сделайте JDBC-драйвер доступным всем приложениям, выполняющимся на веб-сервере. Для этого поместите JAR-файл драйвера в папку `%JAVA_HOME%\jre\ext` (тогда драйвер станет доступным во всех приложениях, запускающихся с использованием данного JDK), либо в папку, находящиеся в которой JAR-файлы доступны всем приложениям в рамках сервера (расположение данной папки зависит от сервера приложений). Для встроенного в среду разработки NetBeans 6 веб-сервера Apache Tomcat 6 такой папкой является `<CATALINA_HOME>\lib`.

- 2) Создайте источник данных. Для этого откройте специфический установочный дескриптор для Tomcat (файл `context.xml` в папке **Configuration Files**) и добавьте в него элемент **Resource** следующего вида:

```
<Context>
    <Resource name="JNDI-имя"
        auth="Container" type="javax.sql.DataSource"
        maxActive="100" maxIdle="30" maxWait="10000"
        username="имя_пользователя" password="пароль"
        driverClassName="имя_класса_JDBC-драйвера"
        url="JDBC URL"/>
</Context>
```

Пример определения источника данных для подключения к СУБД MySQL:

```
<Resource name="jdbc/clients"
    auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000"
    username="javauser" password="javadude"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/javatest"/>
```

- 3) JNDI-имена, указанные при создании источника данных и в исходном коде программы, не обязательно должны совпадать, так как представляют собой различные объекты.

Первое из них обозначает конкретный ресурс в конкретном сервере приложений (источник данных является лишь одним из множества видов ресурсов), поэтому оно должно быть уникальным в рамках сервера приложений. Совпадение JNDI-имен ресурсов в различных серверах приложений не означает эквивалентности самих ресурсов, то есть одинаковые JNDI-имена источников данных в двух разных экземплярах веб-сервера Tomcat могут определять подключения к различным базам данных.

JNDI-имена, используемые в веб-приложении, представляют собой ссылки на ресурсы, которые должны быть уникальными в рамках веб-приложения и которые необходимо привязать к ресурсам сервера приложений при установке приложения.

Внимание: При использовании веб-сервера Tomcat JNDI-имена, указанные при создании источника данных и в исходном коде программы, **ДОЛЖНЫ** совпадать.

- 4) Каждую ссылку на ресурс следует описать в установочном дескрипторе, тогда во время работы веб-приложения в контексте компонента (`java:comp/env`) по ссылке можно будет получить необходимый ресурс. Чтобы добавить ссылку на источник данных, откройте установочный дескриптор (файл `web.xml` в папке **Configuration Files**), выберите страницу **References**, раскройте пункт **Resource References** и нажмите кнопку **Add**. В диалоге создания ссылки на ресурс (рис. 4) введите в поле **Resource Name** JNDI-имя, использованное в классе DAO (см. п.4) либо в классе сервлета (см. п.7). Из вводимого JNDI-имени нужно исключить префикс `java:comp/env/`, обозначающий окружение компонента. Нажмите кнопку **ОК**, убедитесь, что ссылка на источник данных появилась в таблице. Сохраните файл `web.xml`.

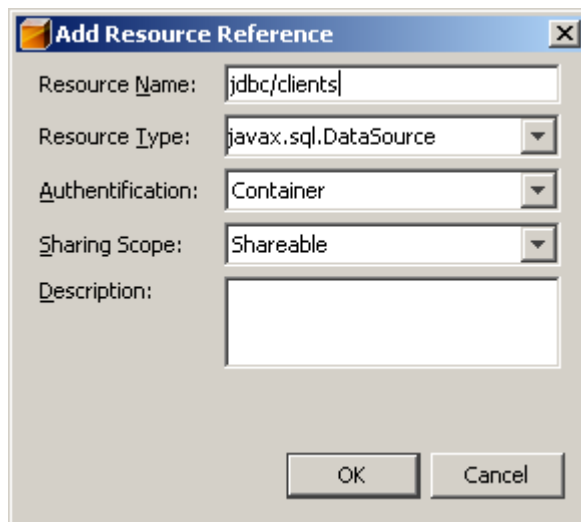


Рис. 4. Создание ссылки на источник данных

9. Создайте в проекте новую JSP-страницу (файл типа **Web / JSP**). В мастере создания JSP-страницы (рис. 5) укажите имя JSP-файла без расширения (поле **JSP File Name**). Нажмите кнопку **Finish**. Открывается окно редактирования JSP-страницы (рис. 6).
10. Справа от окна редактирования размещается окно палитры **Palette**, которое позволяет создавать типовые элементы страниц путем перетаскивания элементов в нужное место JSP-страницы. Создайте шаблон таблицы, перетащив элемент палитры **HTML / Table** в окно редактирования страницы. При этом появляется диалог вставки таблицы (рис. 7), который позволяет указать основные параметры таблицы (количество строк и столбцов, размер границы, ширину таблицы, расстояние между ячейками, внутренний пробел в ячейке). Так как таблица с данными будет формироваться динамически, то нужно установить количество строк равным 1. Нажмите **OK**.
11. Отредактируйте JSP-страницу таким образом, чтобы она выполняла свое предназначение – вывод данных в виде таблицы. Для этого следует использовать скриптовые элементы. Получение данных с помощью класса DAO должен осуществлять сервлет-контроллер.

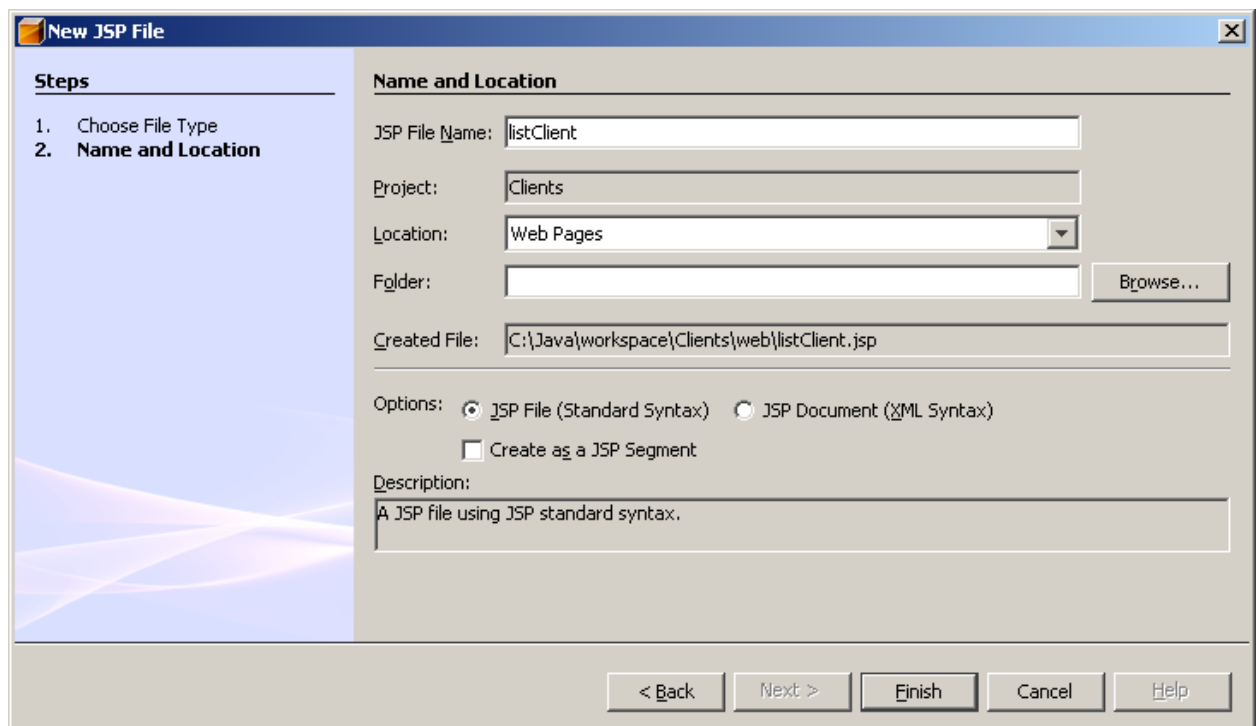


Рис. 5. Создание JSP-страницы

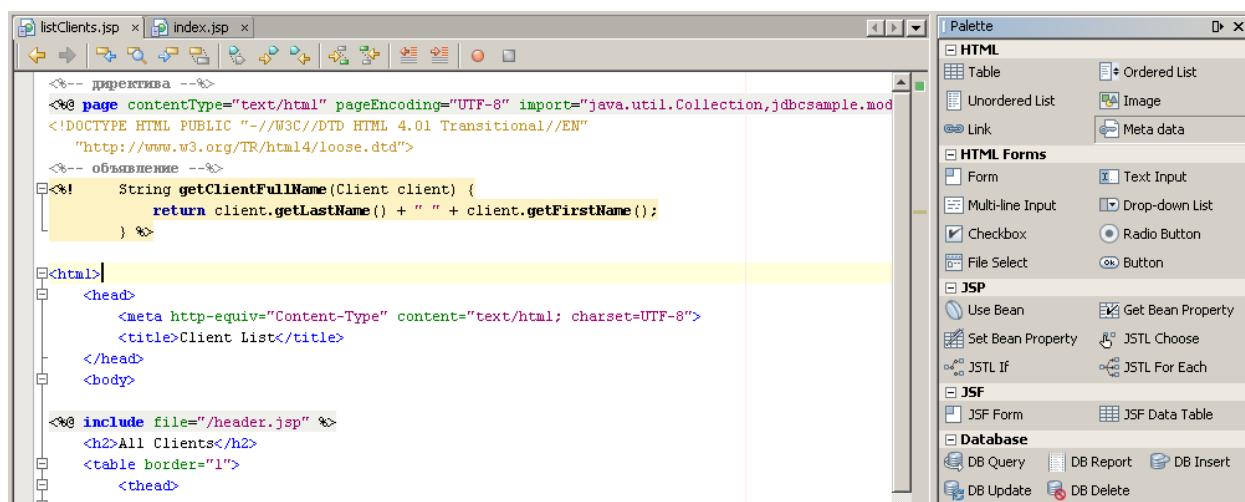


Рис. 6. Редактирование JSP-страницы

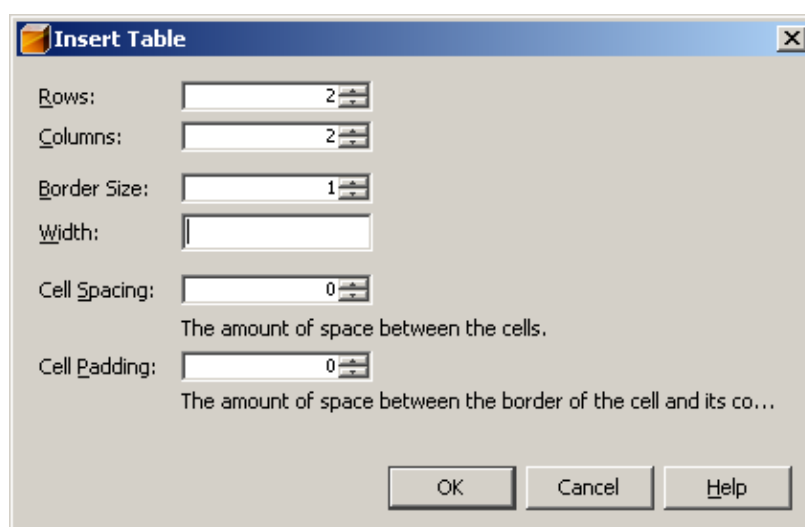
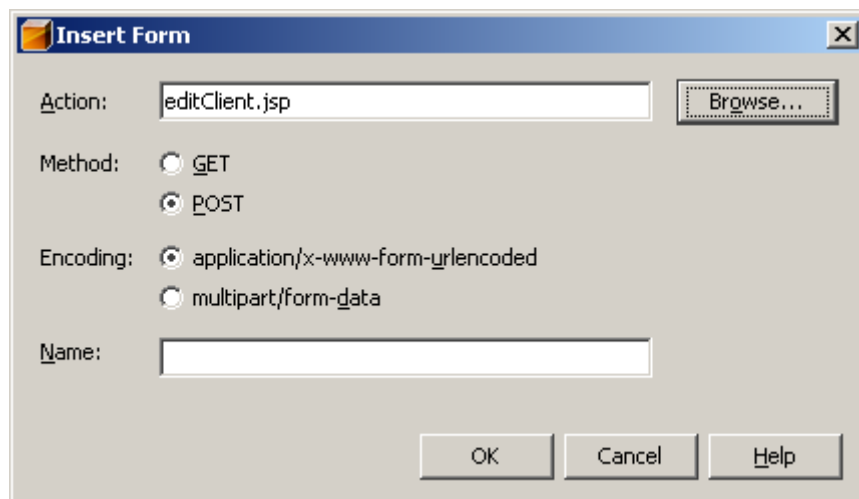


Рис. 7. Вставка таблицы в JSP-страницу

12. Изучите исходный код класса сервлета, полученного при трансляции JSP-страницы. Для этого выберите в контекстном меню редактора JSP-страницы пункт **View Servlet**.
13. Запустите веб-приложение, выбрав в контекстном меню проекта пункт **Run Project**. Введите в открывшемся окне браузера URL запроса с параметрами и убедитесь в правильном функционировании созданной страницы. Выполните несколько запросов, в том числе с неправильными данными. Просмотрите в окне HTTP Monitor обработанные запросы.
14. Создайте JSP-страницу для добавления или редактирования данных. На ней необходимо разместить форму для ввода данных пользователя. Для этого могут пригодиться элементы палитры, объединенные в группу **HTML Forms**. Для вставки формы перетащите элемент **Form** в окно редактирования, при этом откроется диалог вставки формы (рис. 8). Укажите в поле **Action** относительный адрес JSP-страницы или сервлета, который нужно вызвать для обработки данных с формы, и выберите метод отправки HTTP-запроса. Нажмите **OK**.
15. Добавьте в форму поля для ввода данных (элемент **Text Input**, рис. 9) и кнопку для отправки запроса (элемент **Button**, рис. 10).
16. Реализуйте бизнес-логику для добавления или редактирования данных в сервлет-контроллере. В последнем случае может понадобиться специальный параметр для определения действия, которое должен выполнить сервлет. Такой параметр можно передать с помощью «скрытого» элемента формы.

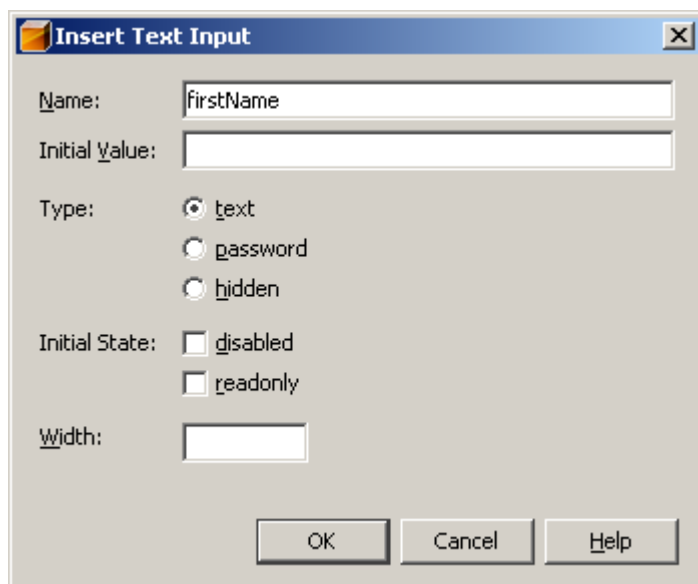
17. Запустите и проверьте корректность функционирования веб-приложения.



The 'Insert Form' dialog box is shown with the following settings:

- Action:** editClient.jsp
- Method:** ☒ GET, ☐ POST
- Encoding:** ☒ application/x-www-form-urlencoded, ☐ multipart/form-data
- Name:** (empty text box)
- Buttons:** OK, Cancel, Help

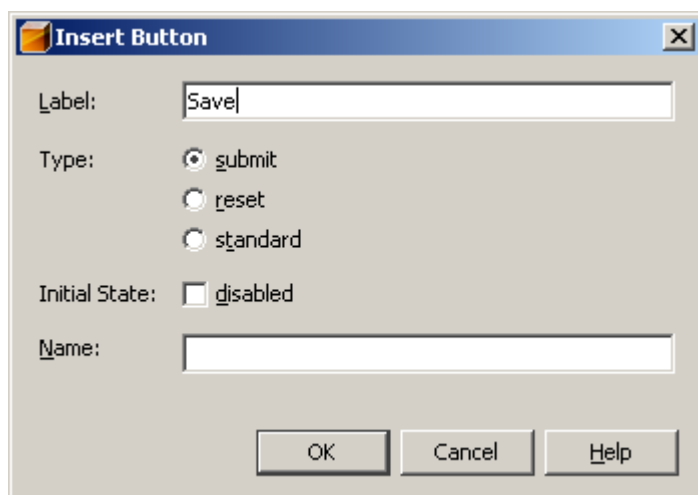
Рис. 8. Вставка формы в JSP-страницу



The 'Insert Text Input' dialog box is shown with the following settings:

- Name:** firstName
- Initial Value:** (empty text box)
- Type:** ☒ text, ☐ password, ☐ hidden
- Initial State:** ☐ disabled, ☐ readonly
- Width:** (empty text box)
- Buttons:** OK, Cancel, Help

Рис. 9. Вставка текстового поля в JSP-страницу



The 'Insert Button' dialog box is shown with the following settings:

- Label:** Save
- Type:** ☒ submit, ☐ reset, ☐ standard
- Initial State:** ☐ disabled
- Name:** (empty text box)
- Buttons:** OK, Cancel, Help

Рис. 10. Вставка кнопки в JSP-страницу

Настройка сервера приложений GlassFish V2

Использование сервера приложений GlassFish V2 для установки веб-приложения имеет следующие особенности:

8.1) JAR-файлы, находящиеся в папке `<GLASSFISH_HOME>\lib\`, доступны всем приложениям в рамках сервера.

8.2) Создание источника данных включает в себя два шага: создание пула соединений и создание JDBC-ресурса. (В сервере Tomcat пул соединений также создается, но неявно.)

1. Создайте в проекте файл типа **GlassFish / JDBC Resource**. На шаге 2 мастера создания JDBC-ресурса установите переключатель **Create New JDBC Connection Pool** (создать новый пул соединений) и введите JNDI-имя источника данных в поле **JNDI Name**. Нажмите кнопку **Next**.

New JDBC Resource

Steps

1. Choose ...
2. **General Attributes - JDBC Resource**
3. Properties
4. Choose Database Connection
5. Add Connection Pool Properties
6. Add Connection Pool Optional Properties

General Attributes

Provide configuration information for the JDBC Resource.
Either choose an existing JDBC Connection Pool, or create a new JDBC Connection Pool.
Fields with an * mark are required.

☐ Use Existing JDBC Connection Pool

< No JDBC Connection Pool >

☒ Create New JDBC Connection Pool

JNDI Name:* jdbc/clients

Object Type: user

Enabled: true

Description:

< Back Next > Finish Cancel Help

2. Пропустите следующий шаг Properties, нажав кнопку **Next**.
3. На шаге 4 введите название пула соединений в поле **JDBC Connection Pool Name**, установите переключатель **New Configuration using Database** и в выпадающем списке выберите пункт, соответствующий используемой СУБД. Нажмите кнопку **Next**.

New JDBC Resource

Steps

1. Choose ...
2. General Attributes - JDBC Resource
3. Properties
- 4. Choose Database Connection**
5. Add Connection Pool Properties
6. Add Connection Pool Optional Properties

Choose Database Connection

Provide configuration information for the JDBC Connection Pool.
Either choose an existing database connection to extract information, or enter the config fields with an * mark are required.

JDBC Connection Pool Name:*

☐ Extract from Existing Connection:

☒ New Configuration using Database:

☐ XA (Global Transaction)

< Back Next > Finish Cancel Help

4. На шаге 5 в таблице Properties установите значения свойств **URL** (JDBC URL для подключения к базе данных), **User** (имя пользователя СУБД) и **Password** (пароль). Нажмите кнопку **Finish**.

New JDBC Resource

Steps

1. Choose ...
2. General Attributes - JDBC Resource
3. Properties
4. Choose Database Connection
- 5. Add Connection Pool Properties**
6. Add Connection Pool Optional Properties

Add Connection Pool Properties

Enter the Datasource Classname, URL, and User to continue.
Hit the Enter key to save values in the Properties table.

Datasource Classname:

Resource Type:

Description:

Properties:

Name	Value
URL	jdbc:mysql://localhost:3306/clients?relaxA...
User	root
Password	1234

Add Remove

< Back Next > Finish Cancel Help