# Store Form Field Data to Multiple Tables (v4)

You may want to store certain fields from your form to other tables with the use of the Beanshell Form Binder. **Figure 1** shows an example of a form where the first 3 fields are to be stored in another data source in addition to the original form data table.

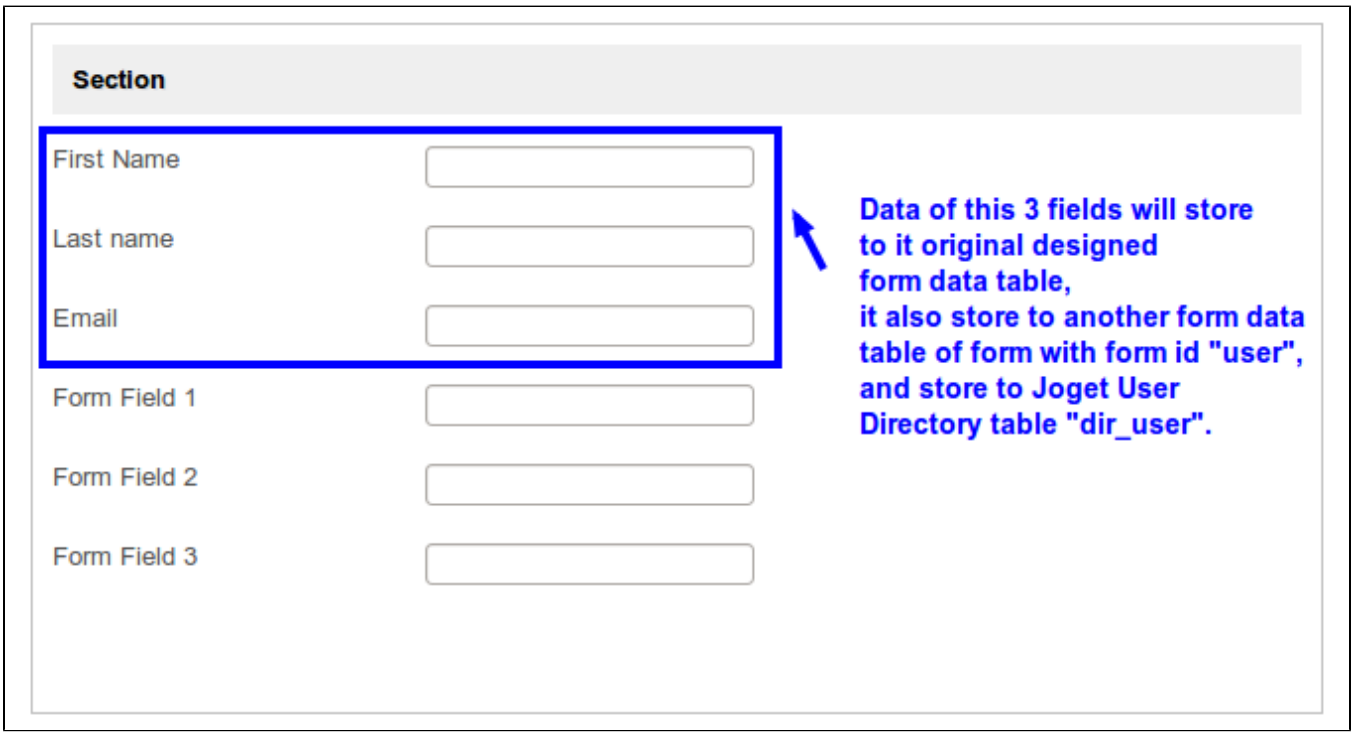


**Figure1: Form with Field to Store**

The quick and easy approach in addressing this requirement is to make use of Beanshell Form Binder in the section's Store Binder. Edit the section.

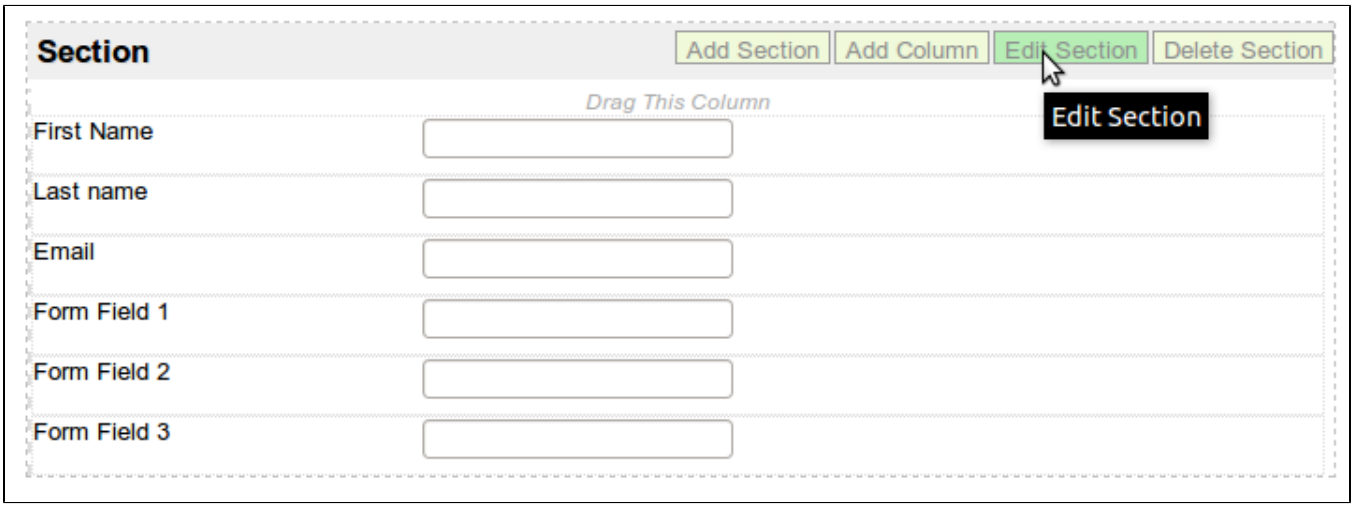


**Figure 2: Configure Section Properties to Determine How Data Will Be Handled**

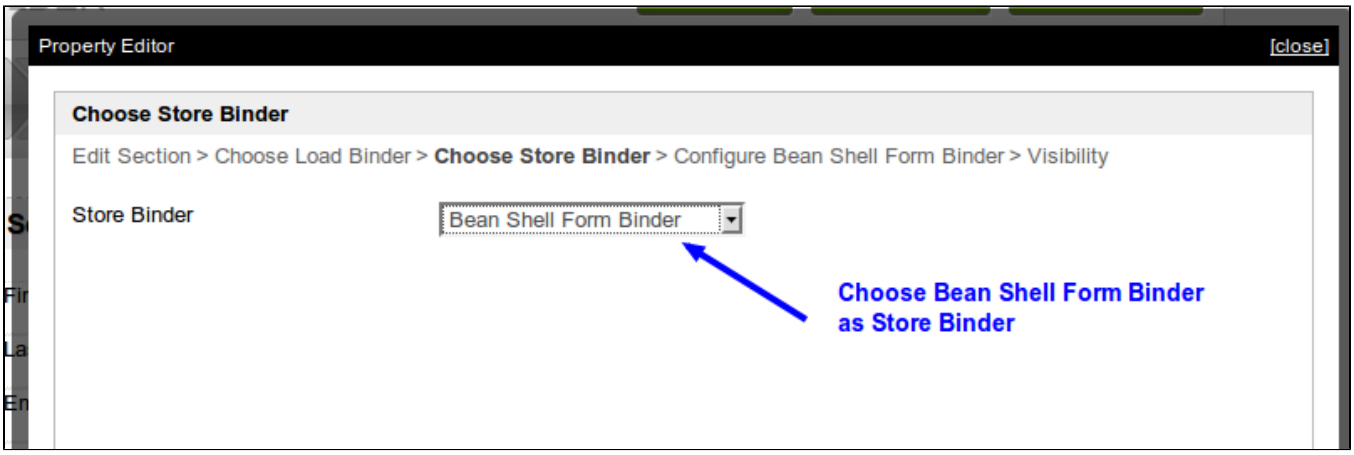


Figure 3: Choose Beanshell Form Binder as the Store Binder

In Store Binder, choose "Bean Shell Form Binder" as the store binder.

Configure Bean Shell Form Binder with your own coding to store the fields as intended, as shown in the figure below.

Code used in this example:

```
import org.joget.apps.app.service.*;
import org.joget.apps.app.model.*;
import org.joget.apps.form.model.*;
import org.joget.apps.form.service.*;
import java.sql.*;
import java.util.*;

public FormRowSet storeData() {
    normalStoring(element, rows, formData);

    //store only needed field by create new Form Row Set
    FormRow originalRow = rows.get(0);

    FormRowSet newRows = new FormRowSet();
    FormRow newRow = new FormRow();

    newRow.put("firstName", originalRow.getProperty("firstName"));
    newRow.put("lastName", originalRow.getProperty("lastName"));
    newRow.put("email", originalRow.getProperty("email"));
    newRows.add(newRow);

    String id = "#currentUser.username#";

    //Store
    storeToOtherFormDataTable(element, newRows, formData, id);
    StoreUsingJDBC(element, newRows, formData, id);

    return rows;
}

//this function will put all the data gather from the element's childs to it's parent store binder
public void normalStoring(Element element, FormRowSet rows, FormData formData) {
    if (rows != null && !rows.isEmpty()) {
        // find parent that have store binder
        Element parent = element.getParent();
        while (parent.getStoreBinder() == null && parent.getParent() != null) {
            parent = parent.getParent();
        }

        FormStoreBinder storeBinder = parent.getStoreBinder();
        if (storeBinder != null) {
            FormRowSet parentRows = formData.getStoreBinderData(storeBinder);
            FormRow currentRow = rows.get(0);
            if (parentRows != null && parentRows.size() == 1 && rows.size() == 1) {
                FormRow parentRow = parentRows.get(0);
```

```
                parentRow.putAll(currentRow);
            } else {
                parentRows = new FormRowSet();
                FormRow parentRow = new FormRow();
                parentRow.putAll(currentRow);
                parentRows.add(parentRow);

                formData.setStoreBinderData(storeBinder, parentRows);
            }
        }
    }
}

//this function will store rows data to a form's data table
public void storeToOtherFormDataTable(Element element, FormRowSet rows, FormData formData, String id) {
    AppService appService = (AppService) FormUtil.getApplicationContext().getBean("appService");

    String formId = "user"; // the table of database is configured in the form with id "user"
    AppDefinition appDef = AppUtil.getCurrentAppDefinition();

    appService.storeFormData(appDef.getId(), appDef.getVersion().toString(), formId, rows, id);
}

//this function will store rows data to external source using JDBC
public void StoreUsingJDBC(Element element, FormRowSet rows, FormData formData, String id) {
    Connection con = null;
    try {
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        con = DriverManager.getConnection("jdbc:mysql://localhost:3307/jwdb?characterEncoding=UTF-8", "root",
"");

        if(!con.isClosed()){
            //manually handle insert and update by checking the data is exist or not
            String selectQuery = "SELECT username FROM dir_user WHERE username=?";
            PreparedStatement stmt = con.prepareStatement(selectQuery);
            stmt.setString(1, id);
            ResultSet rs = stmt.executeQuery();

            Boolean isExist = false;
            if (rs.next()) {
                isExist = true;
            }

            FormRow row = rows.get(0);

            if (isExist) {
                String updateQuery = "UPDATE dir_user SET firstName = ?, lastName = ?, email = ? WHERE username
= ?";
                PreparedStatement ustmt = con.prepareStatement(updateQuery);
                ustmt.setString(1, row.getProperty("firstName"));
                ustmt.setString(2, row.getProperty("lastName"));
                ustmt.setString(3, row.getProperty("email"));
                ustmt.setString(4, id);
                ustmt.executeUpdate();
            } else {
                String insertQuery = "INSERT INTO dir_user (id, username, firstName, lastName, password, email)
values (?, ?, ?, ?, 'md5(password)', ?)";
                PreparedStatement istmt = con.prepareStatement(insertQuery);
                istmt.setString(1, id);
                istmt.setString(2, id);
                istmt.setString(3, row.getProperty("firstName"));
                istmt.setString(4, row.getProperty("lastName"));
                istmt.setString(5, row.getProperty("email"));
                istmt.executeUpdate();
            }
        }
    } catch (Exception ex) {
        System.err.println("Exception: " + ex.getMessage());
    } finally {
        try {
            if(con != null)
```

```
                con.close();
            } catch(SQLException e) {}
        }
    }

return storeData();
```

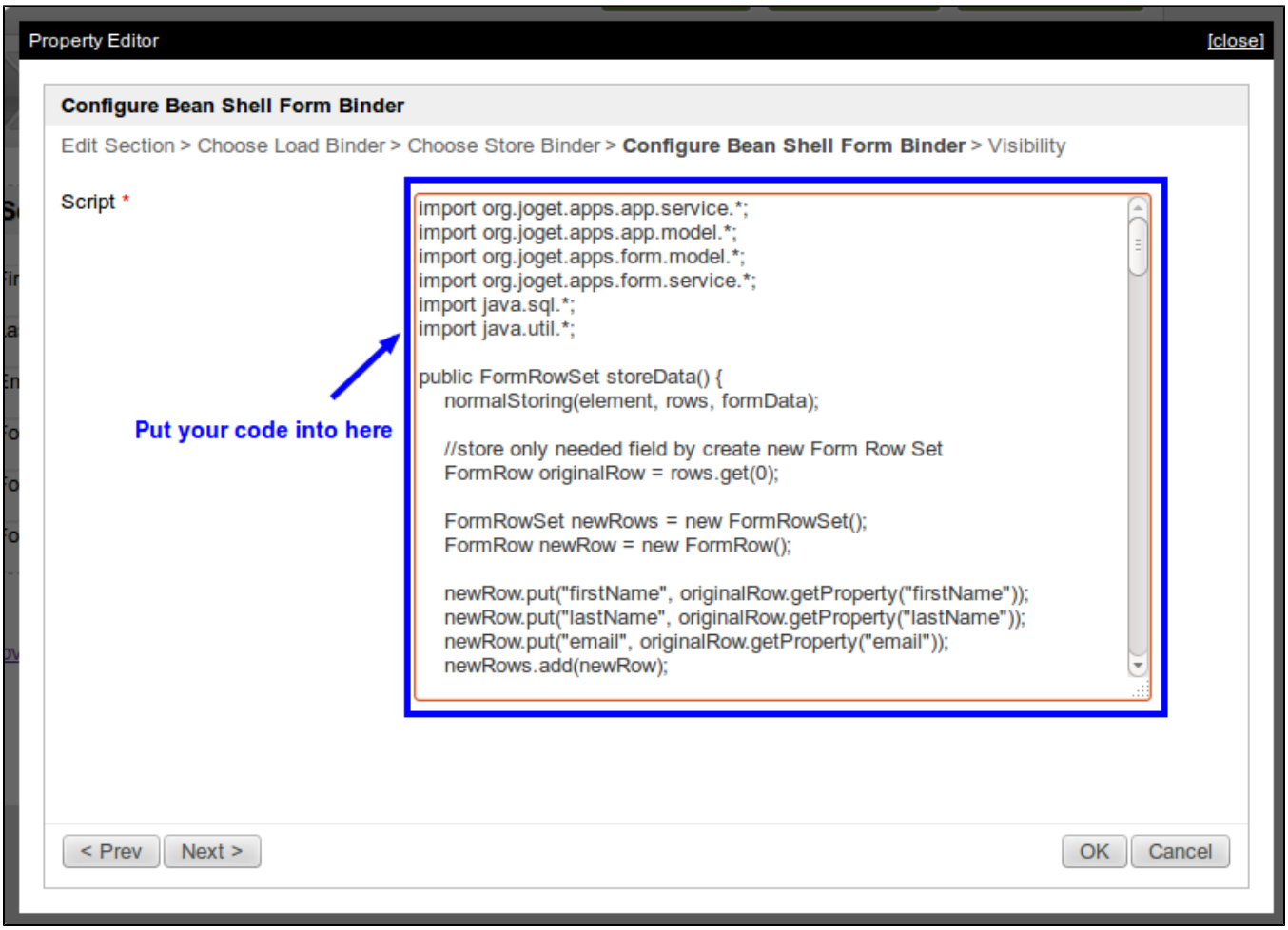


Figure 4: Populate Beanshell Form Binder with the Necessary Codes

If the coding is properly written and tested, you will get this result:

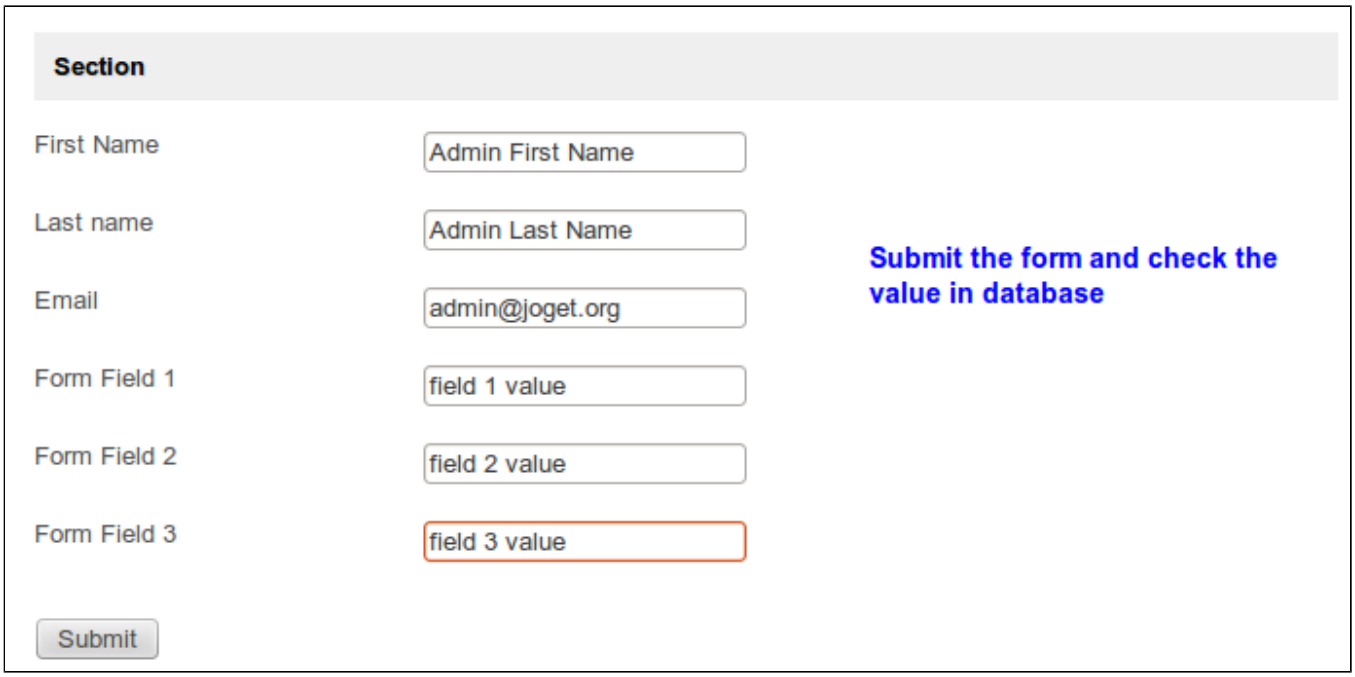


Figure 5: Fill and Submit Form

Check the data in the database.

```
mysql> select * from app_fd_customer \G
*************************** 1. row ***************************
          id: 2ce5be05-7f001010-ccf5ae80-e6d32b25
 dateCreated: 2012-06-27 15:44:11
dateModified: 2012-06-27 15:44:11
  c_lastName: Admin Last Name
    c_field3: field 3 value
    c_field2: field 2 value
     c_email: admin@joget.org
    c_field1: field 1 value
 c_firstName: Admin First Name
1 row in set (0.01 sec)

mysql> select * from app_fd_user \G
*************************** 1. row ***************************
          id: admin
 dateCreated: 2012-06-27 15:44:10
dateModified: 2012-06-27 15:44:10
  c_lastName: Admin Last Name
     c_email: admin@joget.org
 c_firstName: Admin First Name
1 row in set (0.00 sec)

mysql> select * from dir_user where username = 'admin' \G
*************************** 1. row ***************************
       id: admin
 username: admin
 password: 21232f297a57a5a743894a0e4a801fc3
firstName: Admin First Name
 lastName: Admin Last Name
    email: admin@joget.org
   active: 1
 timeZone: 0
1 row in set (0.00 sec)

mysql>
```

Figure 6: Data Stored Correctly in the Tables