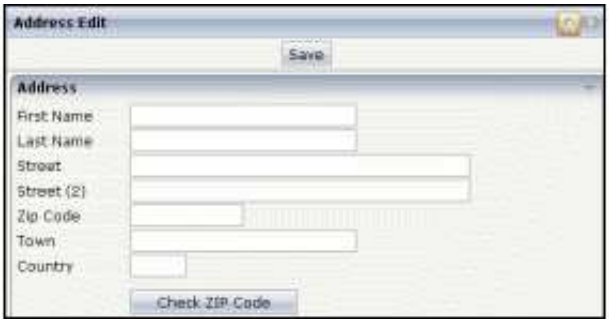
Customized Layout - Example

Customized Layout - Example

Let us take the address maintenance example that was shown in the previous section.



This chapter covers the following topics:

- XML Layout
- Java Adapter Code

XML Layout

The XML layout looks as follows:

```
<page model="CustomizedLayoutAdapter">
    <titlebar name="Address Edit">
        <persedit persprop="paInfo">
        </persedit>
    </titlebar>
    <header withdistance="false">
        <button name="Save" method="onSave">
        </button>
    </header>
    <pagebody>
        <rowarea name="Address" visibleprop="adressAreaVisible">
            <itr visibleprop="firstNameVisible">
                <label name="First Name" width="100">
                </label>
                <field valueprop="firstName" width="200">
                </field>
            </itr>
            <itr visibleprop="lastNameVisible">
                <label name="Last Name" width="100">
                </label>
                <field valueprop="lastName" width="200">
                </field>
            </itr>
            <itr visibleprop="streetVisible">
                <label name="Street" width="100">
                </label>
                <field valueprop="street" width="300">
                </field>
            </itr>
            <itr visibleprop="street2Visible">
                <label name="Street (2)" width="100">
                </label>
                <field valueprop="street2" width="300">
```

```
                </field>
            </itr>
            <itr visibleprop="zipCodeVisible">
                <label name="Zip Code" width="100">
                </label>
                <field valueprop="zipCode" width="100">
                </field>
            </itr>
            <itr visibleprop="townVisible">
                <label name="Town" width="100">
                </label>
                <field valueprop="town" width="200">
                </field>
            </itr>
            <itr visibleprop="countryVisible">
                <label name="Country" width="100">
                </label>
                <field valueprop="country" width="50">
                </field>
            </itr>
            <vdist height="10">
            </vdist>
            <itr visibleprop="checkZipCodeVisible">
                <hdist width="100">
                </hdist>
                <button name="Check ZIP Code" method="onCheckZIPCode">
                </button>
            </itr>
        </rowarea>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
    <personalization>
        <persproposal property="town" comment="Town">
        </persproposal>
        <persproposal property="country" comment="Country">
        </persproposal>
        <persfilter property="adressAreaVisible" group="Areas" comment="Address Area">
        </persfilter>
        <persfilter property="firstNameVisible" group="Address fields" comment="First Name">
        </persfilter>
        <persfilter property="lastNameVisible" group="Address fields" comment="Last Name">
        </persfilter>
        <persfilter property="streetVisible" group="Address fields" comment="Street">
        </persfilter>
        <persfilter property="street2Visible" group="Address fields" comment="Street (2)">
        </persfilter>
        <persfilter property="zipCodeVisible" group="Address fields" comment="Zip Code">
        </persfilter>
        <persfilter property="townVisible" group="Address fields" comment="Town">
        </persfilter>
        <persfilter property="countryVisible" group="Address fields" comment="Country">
        </persfilter>
        <persfilter property="checkZipCodeVisible" group="Address functions"
                    comment="Check ZIP Code">
        </persfilter>
    </personalization>
</page>
```

What are the personalization aspects of the XML layout?

- In the TITLEBAR definition, you see a special PERSEDIT control. This control is rendered as an icon. It is only available if the session context indicates an administrative session (details will be provided later). When choosing this icon, the personalization maintenance appears. The control

references to a PERSPROP property `paInfo`.

- In the various ITR definitions (each ITR holding a label and a field), there are references to VISIBLEPROP properties.
- In the PERSONALIZATION section, there is a list of PERSFILTER definitions. Each definition indicates a personalizable property and holds some additional information: a comment and a group. The group is used to structure the properties in a tree inside the personalization maintenance page. The comment is used as text for the property. Since personalization is not an end-user task but an administrative task, group and comment are not language-dependent and should be kept in the default language of your application.

Java Adapter Code

The adapter code is:

```
// This class is a generated one.

import java.util.Iterator;
import java.util.Properties;

import com.softwareag.cis.pers.Personalization;
import com.softwareag.cis.pers.PersonalizationAdapterInfo;
import com.softwareag.cis.pers.PersonalizationScenarioSequence;
import com.softwareag.cis.server.Adapter;

public class CustomizedLayoutAdapter
    extends Adapter
{
    PersonalizationAdapterInfo m_paInfo = new PersonalizationAdapterInfo(this);
    String m_lastName;
    String m_firstName;
    String m_street;
    String m_street2;
    String m_zipCode;
    String m_town;
    String m_country;

    public PersonalizationAdapterInfo getPaInfo() { return m_paInfo; }

    public String getLastName() { return m_lastName; }
    public void setLastName(String value) { m_lastName = value; }

    public String getFirstName() { return m_firstName; }
    public void setFirstName(String value) { m_firstName = value; }

    public String getStreet() { return m_street; }
    public void setStreet(String value) { m_street = value; }

    public String getStreet2() { return m_street2; }
    public void setStreet2(String value) { m_street2 = value; }

    public String getZipCode() { return m_zipCode; }
    public void setZipCode(String value) { m_zipCode = value; }

    public String getTown() { return m_town; }
    public void setTown(String value) { m_town = value; }

    public String getCountry() { return m_country; }
    public void setCountry(String value) { m_country = value; }
...
    /** initialisation - called when creating this instance*/
```

```
    public void init()
    {
        // switch maintenance on
        Personalization.switchPersonalizationMaintenanceOn(findSessionContext());
        // set up scenarios
        PersonalizationScenarioSequence pss = new PersonalizationScenarioSequence("customer");
        Personalization.defineScenarioSequenceInContext(findSessionContext(),pss);
        // transfer proposal values
        m_paInfo.applyProposals(this);
    }
...
}
```

You see that personalization does not affect the adapter too much:

- There is a member (`m_paInfo`) of type `PersonalizationAdapterInfo` that is made accessible via `getPAInfo()`. This property passes certain information about personalization to the PERSEDIT control inside the page.

- In the `init()` method, there is the call of the method `Personalization.switchPersonalization-MaintenanceOn()`. As parameter, the session context is passed. This method specifies in a certain session context parameter that the current session is a session in which you want to maintain personalization data. It should normally be called during a certain logon page of your application in which you decide by certain logon parameters that now the administrative user logs on in a special administrative mode.

- In the `init()` method, there is the setting of the current personalization scenario. In the example, the scenario sequence contains one scenario: the "customer" scenario. Scenarios are just names that are used as references into the personalization at runtime. The setting of the scenario sequence normally should happen also as part of the logon procedure to your application.

All the rest (the filtering of properties, the calling of the maintenance pop-up, the storing of personalization data) is done automatically. You, the developer, do not have to take care of it.