

# Context Menu

The context menu is not a control itself - it is part of existing controls. Context menus are supported by the following controls:

- TEXTGRID2
- TEXTGRIDSSS2
- TREENODE2
- CLIENTTREE

All these controls have items that are represented by corresponding objects on the server side:

- For the text grid controls, each row of the grid is represented by a certain object on the server side that is managed inside a `TEXTGRIDCollection`.
- For the tree controls, each node of the tree is represented by a certain object (derived from `NODEInfo`) that is managed inside a `TREECollection`.

If the user clicks with the right mouse button on an item, then the method `reactOnContextMenuRequest()` is called inside the item's server-side object. In principle, your server-side implementation could do anything as reaction inside the implementation of the method - i.e. you do not have to open a context menu, but you can also do every other thing that you are able to do inside an adapter implementation.

To open a context menu, you proceed as follows:

- You define a tree collection.
- You add menu nodes to the tree collection - exactly in the same way as you do it with a normal `MENU` control.
- You pass the tree collection to the adapter via its method `showPopupMenu()`.

This chapter covers the following topics:

- Example: Context Menu with a Text Grid
- Context Menu with a Tree

---

## Example: Context Menu with a Text Grid

* □	File Name	XML (last change)	□ ▲
	13_AsciiProtocol.xml	20061228075924000	
	13_DynamicPopupField.xml	20061228075924000	
	13_IconsInTextgrid.xml	20061228075924000	
	13_Row	20061228075924000	
	14_Abs	20061228075924000	
	14_Acc	20061228075924000	
	14_Are	20061228075924000	
	14_Co	20061228075923000	
	14_Co	20061228075924000	
	14_ControlLibraryDyn.xml	20061228075923000	
	14_DefaultMultiLanguage.xml	20061228075924000	
	14_DynamicColumnStyle.xml	20061228075924000	
	14_FieldWithTextAndId.xml	20061228075924000	

There is no special specification necessary inside the layout definition of the corresponding TEXTGRID2/TEXTGRIDSSS2 control. All you have to do is to implement the `reactOnContextMenuRequest()` method inside the class representing the items of the grid:

```
public class MenuAdapter
    extends Model
{
    // -----
    // inner classes
    // -----

    /** class used for pop-up menu. */
    public class MyMenuNodeInfo
        extends MENUNODEInfo
    {
        public MyMenuNodeInfo(String text) { super(text); }
        public MyMenuNodeInfo(String text, String image) { super(text, image); }
        public void reactOnSelect()
        {
            outputMessage("S", "Menu Item \"" + getText() + "\" selected!");
        }
    }

    /** class represents one row within the text grid. */
    public class Line
    {
        String m_name;
        CTimeStamp m_htmlChange;
        CTimeStamp m_xmlChange;
        public Line(String name, CTimeStamp xmlChange, CTimeStamp htmlChange)
        {
            m_name = name;
            m_xmlChange = xmlChange;
            m_htmlChange = htmlChange;
        }
        public String getName() { return m_name; }
        public CTimeStamp getHtmlChange() { return m_htmlChange; }
        public CTimeStamp getXmlChange() { return m_xmlChange; }

        /** This method will be called if the line will be clicked with
         * the right mouse button.*/
        public void reactOnContextMenuRequest()
    }
}
```

```

    {
        // prepare the appropriate popu menu content
        TREECollection menu = new TREECollection();
        menu.addTopNode(new MyMenuNodeInfo(
            "Open in Layout Painter",
            "../HTMLBasedGUI/images/open.gif"),true);
        menu.addTopNode(new MyMenuNodeInfo(
            "Generate Adapter Code",
            "../HTMLBasedGUI/images/java.gif"),true);
        menu.addTopNode(new MyMenuNodeInfo(
            "Maintain Literals",
            "../HTMLBasedGUI/images/literals.gif"),true);
        menu.addTopNode(new MyMenuNodeInfo(
            "&SEPARATOR"),true);
        menu.addTopNode(new MyMenuNodeInfo(
            "Remove (XML & HTML)",
            "../HTMLBasedGUI/images/remove.gif"),true);

        // open the poup menu
        showPopupMenu(menu);
    }
}

...
...
...
...
}

```

Pay attention: the `showPopupMenu()` method is provided by the class `Adapter` - in the implementation example, it is directly accessed from the class `Line`. The class `Line` is an inner class and consequently has full access to all the methods of its surrounding class.

## Context Menu with a Tree

The implementation of a context menu in the tree is absolutely the same as with a text grid - this time you have to implement the `reactOnContextMenuRequest()` method inside the class representing one tree node.