

Providing UI Converters from Plugins

Contributing UI Converters

A UI converter can be contributed by a Java Servoy client plugin.

To do this, the plugin class implementing the `IClientPlugin` (see [Implement the Plugin Interface](#)) must also implement interface `IUIConverterProvider` to produce `IUIConverter` instances in the `getUIConverters()` method, see [api docs](#).

A UI converter class implementing the `IUIConverter` interface has to implement the following three methods that will be displayed in the **Edit text format property** dialog:

- `convertFromObject()`
Convert from UI value to dataprovider value
- `convertToObject()`
Converts from dataprovider value to UI value
- `getToObjectType()`
The UI data type (so the resulting type of the `convertToObject()`), being one of the `IColumnTypes` constants: `TEXT`, `INTEGER`, `NUMBER`, `DATETIME` or `MEDIA`

A UI converter class can define properties in the `getDefaultProperties()` method, which will be displayed in the **Edit text format property** dialog as well.

The dataprovider types that the UI converter supports are to be specified in the `getSupportedDataproviderTypes()` method.

The implemented UI converters returned by the `getUIConverters()` method of the plugin class implementing the `IUIConverterProvider` interface will be then checked by the plugin manager which will add them to the list of Servoy UI converters.

Example

This is an example of a plugin that delivers a converter from joda time (which has support for time periods) to java-util-date (which is supported by Servoy UI elements) time.

```
private static final IUIConverter[] UI_CONVERTERS = new IUIConverter[] { JodaToDateConverter.INSTANCE };

public IUIConverter[] getUIConverters()
{
    return UI_CONVERTERS;
}

static class NrToJodaConverter implements ITypedColumnConverter
{
    static final NrToJodaConverter INSTANCE = new NrToJodaConverter();

    public Object convertToObject(Map<String, String> props, int column_type, Object dbvalue) throws Exception
    {
        if (dbvalue == null)
        {
            return null;
        }
        return new DateTime(Utils.getAsLong(dbvalue));
    }

    public Object convertFromObject(Map<String, String> props, int column_type, Object obj) throws Exception
    {
        if (obj instanceof DateTime)
        {
            return Long.valueOf(((DateTime)obj).getMillis());
        }
        return obj; // or throw an exception
    }

    public Map<String, String> getDefaultProperties()
    {
        return null;
    }

    public int[] getSupportedColumnTypes()
    {
        return new int[] { IColumnTypes.INTEGER };
    }

    public String getName()
    {

```

```

        return getClass().getSimpleName();
    }

    public int getToObjectType(Map<String, String> props)
    {
        // Servoy does not understand joda time natively.
        return IColumnTypes.MEDIA;
    }
}

static class JodaToDateConverter implements IUIConverter
{
    static final JodaToDateConverter INSTANCE = new JodaToDateConverter();

    public Object convertFromObject(Map<String, String> props, int input_type, Object converted) throws Exception
    {
        if (converted instanceof Date)
        {
            return new DateTime(((Date)converted).getTime());
        }
        return converted; // or return null or throw an exception
    }

    public Object convertToObject(Map<String, String> props, int input_type, Object input) throws Exception
    {
        if (input instanceof DateTime)
        {
            return new Date(((DateTime)input).getMillis());
        }
        return input; // or return null or throw an exception
    }

    public Map<String, String> getDefaultProperties()
    {
        return null;
    }

    public int[] getSupportedDataproviderTypes()
    {
        // Servoy does not understand joda time natively, therefore the dataprovider type will be MEDIA.
        return new int[] { IColumnTypes.MEDIA };
    }

    public String getName()
    {
        return getClass().getSimpleName();
    }

    public int getToObjectType(Map<String, String> props)
    {
        return IColumnTypes.DATETIME;
    }
}

```