Components

The SmartWeb framework for facilitating the development of web applications built on the React JavaScript library offers several preset components with connection to the D2000 system via D2Api. Supplied components are located in the *components* directory of the SmartWeb framework and they are further logically divided into directories according to their designation. Some components can be imported alternatively also with CSS styles - in that case, the file name has the prefix "themed". "Themed" versions of components are basically the "unthemed" ones together with preset CSS styles. For example:

```
// themed version of alert component
import alert from 'framework/components/alert/themedAlert'


// alert component version without CSS styles - unthemed, it is necessary to import CSS manually
import alert from 'framework/components/alert/alert'
```

Dialogue Windows (framework/components/alert)

Alert

The *Alert* component serves for displaying dialogue windows. The component uses SweetAlert2 as a source library and its interface is identical with the interface of the SweetAlert2 library. The component has a styled version with the predefined CSS styles too.

```
// themed version of alert component
import alert from 'framework/components/alert/themedAlert'
// alert component version without CSS styles - unthemed, it is necessary to import CSS manually
// import alert from 'framework/components/alert/alert'

// example of usage
alert({title: 'Success', text: 'It was successful!', type: 'success'});
```

Authentication Components (framework/components/auth)

LoginForm

The *LoginForm* component represents a form with which the user logs in to the web application - to the D2000 system. To use the *LoginForm* component, it is necessary to define a text field (html element <*input*>) with the name *"j_username"* as a user name, a text field with the name *"j_password"* for the password of a user and the submitting button (element <*button type="submit"*>). The login operation itself is implemented on the SmartWeb level and after successful authentication, the user is redirected to the page *pages/index.html*.

```
import React from 'react'
import SimplePageContainer from 'framework/components/react/simplePageContainer'
import LoginForm from 'framework/components/auth/loginForm'

// Simple example of LoginForm component usage
SimplePageContainer.renderComponent(class LoginPage extends React.Component {
        render() {
            return (
                <LoginForm {...this.props}>
                    <input type="text" placeholder="User name" required="" name="j_username"/>
                    <input type="password" placeholder="Password" name="j_password"/>
                    <button type="submit">Login</button>
                </LoginForm>
            );
        }
    }
);
```

LogoutLink

The *LogoutLink* component is a simple reference (html element <*a*>); for clicking it, there will be a confirming dialogue of user's logout displayed. After confirming, the user is logged out of the application and redirected to the initial (login) page.

```
import React from 'react';
import LogoutLink from "framework/components/auth/logoutLink";

// Example of LogoutLink component usage as one of menu items
export default class PageHeader extends React.Component {
    render() {
        return (
            <div className="container-fluid">
                <div className="row">
                    <div className="col-xs-4">
                        <a href="#section1">Section 1</a>
                    </div>
                    <div className="col-xs-4">
                        <a href="#section2">Section 2</a>
                    </div>
                    <div className="col-xs-4">
                        <a href="#section3">Section 3</a>
                    </div>
                    <div className="col-xs-4">
                        <LogoutLink>Logout</LogoutLink>
                    </div>
                </div>
            </div>
        );
    }
};
```

Components for Communication with D2000 (framework/components/d2)

DataContainer

It is a basic component of the SmartWeb Framework. This descendant of the *SimplePageContainer* component packs the D2Api interface and automatically follows and ends connection to D2000. It also secures support for keyboard shortcuts to switch debugging mode (CTRL+ALT+D) a to rebundle application (CTRL+ALT+R). It is promoted as an attribute with the name *d2* into other components.

Attribute name	Type	Mandatory	Default value	Description
connection	object			It contains information about the connection to D2000 as an application name and logged in user name. The attribute is set automatically.

TclScheme

This component serves for displaying the D2000 scheme using the technology of a thin client (TCL). The scheme will be entered to the page as the HTML element <*iframe*>. Component attributes are:

Attribute name	Type	Mandatory	Default value	Description
d2	object	yes		Reference to the DataContainer component. It is submitted from the parent container.
datasource	string	yes		It is the name of the D2000 object of the scheme type that should be displayed.

ValueComponent

This component shows the current value of the D2000 object. The value is displayed in the HTML element <*span*>. The following table describes the attributes of the component:

Attribute name	Type	Mandatory	Default value	Description
d2	object	yes		Link to the DataContainer component. It is submitted from the parent container.
datasource	string	yes		It is the name of the D2000 object and its value should be displayed.
returnFields	string[]			List of value attributes that should be sent together with the object value. Possbile attributes are described on the page Serialization of Data between Client and API Interface.
description	string			Manually defined object description - it is preferred to the description defined in D2000.

technicalUnits	string			Manually defined technical object units - they are preferred to those defined in D2000.
numberFormat	string			Format for numerical object values in the form fit for the Numeral.js library. If it is defined, it will be preferred to the format value sent from D2000.
dateFormat	string			Format for values of the absolute time type in the form fit for the Moment.js library. If it is defined, it will be preferred to format value sent from D2000.
missingDescription	string			The description that is used when it was not defined manually nor in D2000.
missingFormattedValue	string			Format value used when the object does not have a valid value.
missingTechnicalUnits	string			Technical units used when they were not defined manually nor in D2000.
className	string			Name of the CSS class that will be applied to the HTML element <*span*> displaying object value.
style	object			CSS style that will be applied to the HTML element <*span*> displaying object value.

Components of Datepicker (framework/components/datetime)

BootstrapDatePicker

It is the component of a datepicker based on the Bootstrap Datepicker component which has the following attributes:

Attribute name	Type	Mandatory	Default value	Description
name	string			Unique name for the HTML element <*input*>.
className	string			Name of the CSS class that will be applied to the HTML element <*input*> displaying the chosen date.
defaultValue	string date			Default value. It can be entered as a text or as a date object.
disabled	bool		false	It forbids or allows modification.
enableKeyboardEdit	bool		true	It allows or forbids modification by a keyboard.

Example of a usage

```
// themed version of alert component
import DatePicker from 'framework/components/datetime/themedBootstrapDatePicker'
// component version without CSS styles - unthemed, it is necessary to manually import CSS
// import DatePicker from 'framework/components/datetime/bootstrapDatePicker'

// usage example in JSX
<div className="row">
  <div className="col-md-2">
      <DatePicker defaultValue={new Date()}/>
  </div>
</div>
```

BootstrapDateRangePicker

The *BootstrapDateRangePicker* component is an extension of the datepicker. It displays two dates by which the interval is defined. Apart from attributes of the packed component Bootstrap Datepicker, there are the following attributes:

Attribute name	Type	Mandatory	Default value	Description
name	string			Unique name for the HTML element <*input*>.
className	string			Name of the CSS class that will be applied to the HTML element <*input*> displaying the chosen date.
defaultFromValue	string date			The default value of the interval beginning. It can be entered as a text or as a date object.
defaultToValue	string date			The default value of the interval end. It can be entered as a text or as a date object.
betweenDatesText	string		"do"	Text displayed between dates.

disabled	bool		false	It forbids or allows modification.
enableKeyboardEdit	bool		true	It allows or forbids modification by a keyboard.

Components of Lists (framework/components/list)

LazyList

The *LazyList* component implements a list that supports gradual reading of content when scrolling downward. In its basic form, it displays content using elements * *. For final usage, an overload of *renderItem* and *render* methods, which have are responsible for depicting the list content, is expected. The component recognizes the following attributes:

Attribute name	Type	Mandatory	Default value	Description

loadHandler	function		function (size, callback) {
			
			callback ([]);
			
			}
				The function that is called when there is a request for reading more data. The *size* parameter defines the maximal number of elements that should be read and the *callback* parameter is a function using which read data will be added to the end of the list.

batchSize	number		10	The default number of elements which is read in one batch. It is submitted as a parameter into the *loadHandler* fu nction.

Basic Components Widening React (framework/components/react)

ConfigurableComponent

This component serves as a wrapper for components that are not native React components. By creating a descendant of this component and by overloading methods *configureComponent*, *render* and *wrappedInstance,* it is possible to simply add to them the support of displaying using the React framework.

RootElement

A component that replaces the HTML element with the id = "root" for the chosen React component. This component has only internal usage.

SimplePageContainer

A basic container of a web page. It secures support for keyboard shortcuts to switch debugging mode (CTRL+ALT+D) and to rebundle applications (CTRL+ALT+R). It is used for React pages that do not need a connection to D2000. For creating web pages which require a connection to D2000, the descendant of the *DataContainer* component serves.

Table Component (framework/components/table)

DataTable and ThemedDataTable

The *Table* component encompasses the DataTables library. The interface of this library has these additional attributes:

Attribute name	Type	Mandatory	Default value	Description
className	string			The name of the CSS class which will be applied to the main HTML element displaying the table.
style	object			CSS style which will be applied to the main HTML element displaying table.
width	string			Width of the table.
height	string			Height of the table.
renderFooter	bool		false	Column names or flags will be displayed in the footer of the table.

```
import React from 'react'
import {DataContainer} from '../custom/components'
import Table from 'framework/components/table/themedDataTable'
// component version without CSS styles - unthemed, it is necessary to manually import CSS
// import Table from 'framework/components/table/dataTable'

// Example of using ThemedDataTable component that reads data by calling RPC
DataContainer.renderComponent(class IndexPage extends React.Component {

    async loadData(data, callback, settings) {
        const d2Api = this.props.d2;
        const response = await d2Api.rpc("E.SmartWebApiTutorial", "GetPersonData", 1000, {
            type: 'record',
            structType: 'SD.Arr_Person',
            returnAs: 'result',
            value: []
        }).call();

        callback({
            data: response.result,
            recordsTotal: response.result.length
        });
    }

    render() {
        return (
            <div>
                <Table className="table table-striped table-bordered dt-responsive nowrap dataTable" width="
100%"
                       ajax = {(data, callback, settings) => {
                           this.loadData(data, callback, settings);
                       }}
                       columns = {[
                           {title: "Name", data: "name.value"},
                           {title: "Age", data: "age.value"}
                       ]}
                       select={true}
                       ref = {(component) => {
                           this.table = component;
                       }}/>
            </div>
        );
    }
});
```