Validation UX

For servoy components you can use the angular classes (https://docs.angularjs.org/api/ng/directive/form) to set appearance of the component when it is not valid, by default servoy adds this class to the default css:

```
.ng-invalid {
        background-color: #f90000;
}
```

You can overwrite this value by providing your own .ng-invalid class.

This ng-invalid class is set on the component when a validation error occurs on the server or the onDataChange returns false.

In the NGClient the default components also accepts a string as the return value of a n onDataChange, this (i18n) string value is then shown as the tooltip of the component, to let the user know what is really wrong.

For webcomponent developers (that want to implement the same behavior in webcomponents)

The servoy default components are using the default angular by using the ngModel directive (https://docs.angularjs.org/api/ng/directive/ngModel) and its controller (https://docs.angularjs.org/api/ng/type/ngModel.NgModelController) to set the field to invalid (or valid)

First the component has this in the spec:

```
 "dataProviderID" : { "type":"dataprovider", "scope" :"design", "ondatachange": { "onchange":"
onDataChangeMethodID", "callback":"onDataChangeCallback"}},
```

This means that for the dataprovider that is attached to that property, that can have a ondatachange event, that ondatachange event return value will be send to the callback method called 'onDataChangeCallback'.

The code of the onDataChangeCallback is then as follows:

```
link:function($scope, $element, $attrs, ngModel) {
                        var storedTooltip = false;
                        // fill in the api defined in the spec file
                        $scope.api.onDataChangeCallback = function(event, returnval) {
                                var stringValue = typeof returnval == 'string'
                                if(!returnval || stringValue) {
                                        $element[0].focus();
                                        ngModel.$setValidity("", false);
                                        if (stringValue) {
                                                if ( storedTooltip == false)
                                                        storedTooltip = $scope.model.toolTipText;
                                                $scope.model.toolTipText = returnval;
                                        }
                                }
                                else {
                                        ngModel.$setValidity("", true);
                                        $scope.model.toolTipText = storedTooltip;
                                        storedTooltip = false;
                                }
                        }
```

In the link function the system asks also for the ngModel of this component, then in the callback we are looking if the returnval is a boolean or a string value. If it is a string value or a false then we call the ngModel method $setValidity() to let the system know that this component is in an invalid state.

The string value is then set as the tooltip (and existing tooltip is remembered, ngModel controller then sets the class ng-invalid on this component.

If there is no return value (or true) then we are setting the validity to true again and the ng-invalid class will be removed from the classes of this component. Also the tooltip is restored to the normal value.