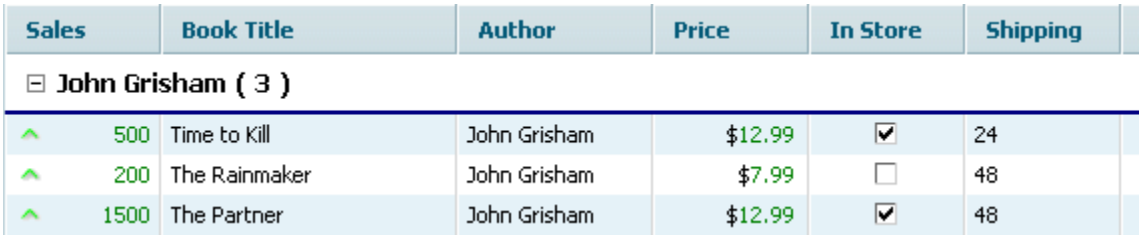
Runtime grouping allows to change the look of data in the grid dynamically, re-organizing it for better presentation of information.
By default, grouping allows the user to show only grouping-key and count of items in the group.

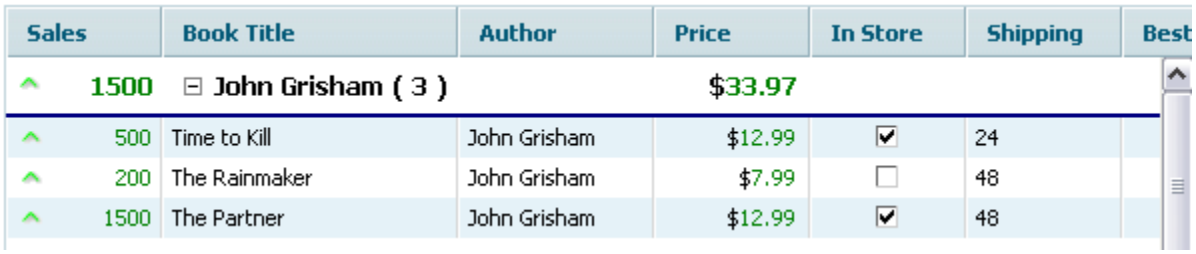
```
<script>
   grid.groupBy(2);
</script>
```



But, in many cases, such look and feel is not enough. The group contains a value that may be aggregated and shown as some kind of total.
The second parameter of **groupBy()** command allows to define the mask used by group-line:

```
<script>

 grid.groupBy(2,["#stat_max","#title","","#stat_total","","#cspan","#cspan","#cspan"]);
</script>
```



The second parameter of **groupBy()** command is an array, each value of which is mapped to the related column.
The possible values are:

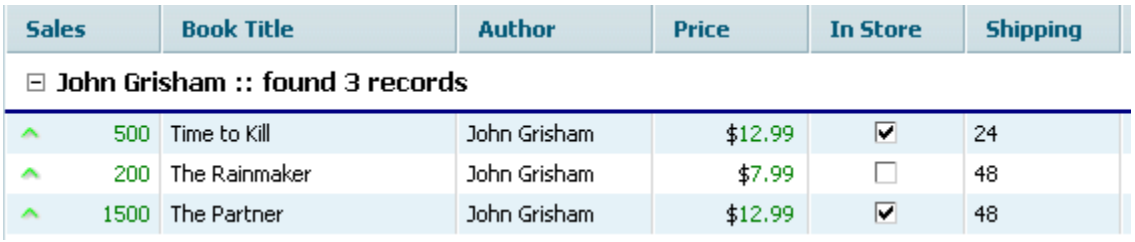
- **title** - will be used for group-key;
- **cspan** - organize colspan with a sibling cell (the same as in **cspan** in header);
- **stat_total** - calculates total of values for the group;
- **stat_max** - calculates maximum value in the group;
- **stat_min** - calculates minimum value in the group;
- **stat_average** - calculates average value in the group;
- **stat_count** - calculates count of records in the group.

Stat-based values are rendered using the same exCell as the related column. This allows to

use **setNumberFormat()** against them (when the source column is of ron|edn type).

In normal mode, the grid allows to redefine the text of group-row with the help of **grid.customGroupFormat**:

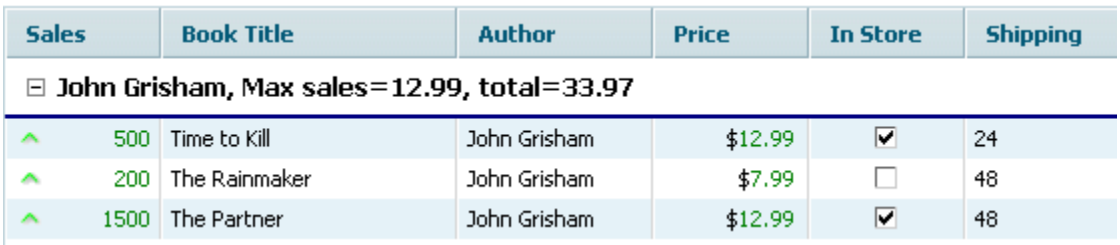
```
<script>
   grid.groupBy(2);
   grid.customGroupFormat=function(name,count){
      return name+" :: found " +count+ " records";
   }
</script>
```



It is possible to add aggregation values to such custom defined group-line as well. It can be done through **groupStat()** method that returns the result of aggregation for the group and accepts the following parameters:

- group name;
- column index;
- name of stat operation (the same as markers above).

```
<script>
   grid.groupBy(2);
   grid.customGroupFormat=function(name,count){
      return name+", Max sales="+grid.groupStat(name,3,"stat_max")+",
total="+grid.groupStat(name,3,"stat_total");
   }
</script>
```



If all stated above is still not enough, there is a way to iterate through all rows in some

group and calculate any custom math.
It can be done using built-in iterator:

```
<script>
   grid.forEachRowInGroup(name,function(id){
        do_something_with_row(id);
   });
</script>
```

The parameter **name** is the key-value of the group.