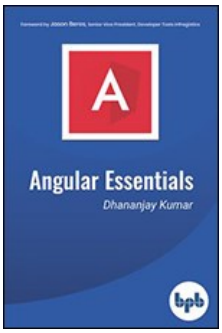

Angular Essentials: The Essential Guide to Learn Angular

by Dhananjay Kumar

Chapter 14: Ignite UI for Angular

Ignite UI for Angular is Angular material based UI components, which help you to create enterprise application faster. There are more than 50 components in Ignite UI for Angular library. However, in this chapter, we will focus on one of most popular Ignite UI for Angular component called `igxGrid`. Ignite UI for Angular Grid is fastest Angular Grid and very easy to work with. In this chapter following topics will be covered:

- Introduction of Ignite UI for Angular
- Ignite UI for Angular Grid
- Add igx-grid to an Angular project
- Add igx-grid component
- Reading a Grid in the Component Class
- Configuring Columns
- Creating Column Templates
- Enable Pagination
- Enable Sorting
- Enable Filtering

Ignite UI for Angular

Ignite UI for Angular is material based Angular components, which help you to write enterprise angular application faster. There are more than 50 components in Ignite UI for Angular library. You can learn more about Ignite UI for Angular here: https://www.infragistics.com/products/ignite-ui-angular

You can use high performance grid, data visualizations charts, calendars, List View, and so on. from Ignite UI for Angular library in your enterprise application to speed up development process and render high performance application. Most popular components in Ignite UI for Angular library are as follows:

- Data Grid
- Tree Grid
- List View
- Combo
- Pie Chart
- Financial Chart
- Category Chart
- Navigation Drawer
- Navbar
- TimePicker
- DatePicker

There are many other components, directives, and services available than above mentioned list in Ignite UI for Angular library.

Ignite UI for Angular Grid

The Ignite UI for Angular Grid is the fastest Angular Grid. It is material-based and can be used as a native component in an Angular application. Ignite UI for Angular comes with rich a set of features, including:

- Virtualization
- Editing
- Paging
- Filtering
- Sorting
- Group By
- Summary
- Column moving , pinning, hiding, template
- Searching
- Selection
- Export to Excel
- Copy Paste from Excel
- Conditional Cell Styling

Learn more about Ignite UI forAngular Grid here: https://www.infragistics.com/products/ignite-ui-angular/angular/components/grid.html

Ignite UI for Angular, which may also be referred to as **<igx-grid>** or **IgxGridComponent** throughout this chapter.

Add igx-grid to an Angular Project

There are three ways to add an **igx-grid** to an Angular project:

1. If starting a new project, use the Ignite UI CLI to scaffold the project. You can use command line options to add the **igx-grid,** including dependency installation.
2. In an existing project, you can use the Ignite UI for Angular Toolbox Extension to add an **igx-grid** in the project. Learn how, in this blog post.
3. You can use **npm** to install Ignite UI for Angular dependencies in your project.

If you have created a project using the Angular CLI, you can use **npm** to install Ignite UI for Angular dependencies to your project. Use **npm** to install dependencies of Ignite UI for Angular and HammerJS.

npm install igniteui-angular

npm install Hammerjs

After successful installation, you need to modify the **angular.json** file as shown in *code listing 14.1.*

Code Listing 14.1

```
      "styles":   [
         "src/styles.css",
            "node_modules/igniteui-angular/styles/
igniteui-angular.css" ],
         "scripts":   ["node_modules/hammerjs/hammer.
min.js"  ]
```

In addition, you need to modify **style.css** to import **Material Icons** as Ignite UI for Angular, so you can use them. To do that,

open the **style, css** file and, in the top section, add the code as shown in *code listing 14.2.*

Code Listing 14.2

```
@import                    url('https://fonts.googleapis.com/
icon?family=Material+Icons');
```

As the last step, you need to import **hammerjs** in **main.ts**. Open the **main.ts** file and add the code shown in *code listing 14.3.*

Code Listing 14.3

```
import  'hammerjs';
```

At this point, you have installed dependencies of Ignite UI for Angular in your project and configured it to use Ignite UI for Angular components.

Before we move ahead, keep in mind that you can use the Ignite UI for Angular Toolbox from Visual Studio Marketplace to right click, add a desired Ignite UI for Angular component, and install all required dependencies in your project.

Add igx-grid Component

To add an. igx-grid or Ignite UI for Angular Grid component, first import the required module in the **AppModule** as shown in *code listing 14.4.*

Code Listing 14.4

```
import { BrowserModule } from '@angular/platform-
browser';
import { NgModule } from '@angular/core';
import { IgxGridModule } from 'igniteui-angular';
import { AppComponent } from './app.component';

@NgModule({
     declarations: [
        AppComponent
     ],
     imports: [
        BrowserModule,
        IgxGridModule.forRoot()
     ],
     providers: [],
     bootstrap: [AppComponent]
})
export class AppModule { }
```

Keep in mind that you need to call **forRoot()** on **IgxGridModule** to register it with the application root module.

To start, we are going to bind local data in the igx-grid. To do that, in **AppComponent** class, add a **getData** function as shown in *code listing 14.5.*

Code Listing 14.5

```
getData()  {
   return   [
      {
         "name":  "Leticia Grisewood",
         "email":   "lgrisewoodbnSyoutube.com",
         "company":   "Innotype",
         "position":   "Administrative Assistant III",
         "city":   "Aiken",
         "post_code":   "2 98 05",
         "state":  "SC",
         "country":   "United States",
         "created_on":  "Sat Mar 31 2018 00:00:00 GMT-0400
(Eastern Daylight Time)",
         "last_activity": "Tue Apr 24 2018 00:00:00 GMT-
0400   (Eastern Daylight Time)",
```

```
          "estimated_sales":  "2026416",
          "actual_sales":  "714549",
          "tags": "demo"
       }'
       {
          "name":   "Roderic Gwilt",
          "email":   "rgwilt6dOox.ac.uk",
          "company":   "Minyx",
          "position":  "Developer IV",
          "city":   "Akron",
          "post_code":   "44393",
          "state":  "OH",
          "country":   "United States",
          "created_on": "Fri Apr 06 2018 00:00:00 GMT-0400
(Eastern Daylight Time)",
          "last_activity": "Sat Apr 21 2018 00:00:00 GMT-
0400   (Eastern Daylight Time)",
          "estimated_sales":  "4575528",
          "actual_sales":   "365964",
          "tags": "cold"
       },
       {
            "name": "Marlee Roote",
            "email": "mroote6v@vk.com",
            "company": "Skalith",
            "position": "Tax Accountant",
            "city": "Albany",
            "post_code": "12210",
            "state": "NY",
            "country": "United States",
            "created_on": "Sun Dec 17 2017 00:00:00 GMT-0500
(Eastern Standard Time)",
            "last_activity": "Mon Jan 01 2018 00:00:00 GMT-
0500 (Eastern Standard Time)",
            "estimated_sales": "120282",
            "actual_sales": "3000442",
            "tags": "Invalid Date"
          },
          {
            "name": "Jaine Schustl",
            "email": "jschustlav@utexas.edu",
            "company": "Gigazoom",
            "position": "Food Chemist",
            "city": "Albany",
            "post_code": "12232",
            "state": "NY",
            "country": "United States",
            "created_on": "Tue Sep 12 2017 00:00:00 GMT-0400
(Eastern Daylight Time)",
            "last_activity": "Thu Sep 28 2017 00:00:00 GMT-
0400 (Eastern Daylight Time)",
            "estimated_sales": "2002221",
            "actual_sales": "2596208",
            "tags": "pro"
          },
          {
            "name": "Kally Foux",
            "email": "kfoux9c@e-recht24.de",
            "company": "Jaxspan",
            "position": "Nurse",
            "city": "Albuquerque",
            "post_code": "87195",
            "state": "NM",
            "country": "United States",
          "created_on": "Thu Mar 29 2018 00:00:00 GMT-0400
(Eastern Daylight Time)",
            "last_activity": "Fri Mar 30 2018 00:00:00 GMT-
0400 (Eastern Daylight Time)",
            "estimated_sales": "1952070",
            "actual_sales": "1023200",
            "tags": "cold"
          }
        ];
    }
```

`getData()` function returns an array with different contacts. Create a local variable, set as a data source of igx-grid, and assign result of **`getData()`** function in that as shown in the *code listing 14.6.*

Code Listing 14.6

```
   localData: any =   [];

   ngOnInit() {
      this.localData = this.getData();
   }
```

Instead of **`ngOnlnit()`** life cycle, you can also call **`getData()`** function inside constructor. Putting everything together, the **`AppComponent`** code should look like as shown in *code listing 14.7.*

Code Listing 14.7

```
import { Component, OnInit } from '@angular/core';

@Component({
     selector: 'app-root',
     templateUrl: './app.component.html',
     styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {

     localData: any = [];
     ngOnInit() {
        this.localData = this.getData();
     }
   getData() {
     return [
      {
           "name": "Leticia Grisewood",
           "email": "lgrisewoodbn@youtube.com",
           "company": "Innotype",
           "position": "Administrative Assistant III",
           "city": "Aiken",
           "post_code": "29805",
           "state": "SC",
           "country": "United States",
           "created_on": "Sat Mar 31 2018 00:00:00 GMT-0400
(Eastern Daylight Time)",
           "last_activity": "Tue Apr 24 2018 00:00:00 GMT-
0400 (Eastern Daylight Time)",
        "estimated_sales": "2026416",
        "actual_sales": "714549",
           "tags": "demo"
         },
         {
           "name": "Roderic Gwilt",
           "email": "rgwilt6d@ox.ac.uk",
           "company": "Minyx",
           "position": "Developer IV",
           "city": "Akron",
           "post_code": "44393",
           "state": "OH",
           "country": "United States",
           "created_on": "Fri Apr 06 2018 00:00:00 GMT-0400
(Eastern Daylight Time)",
           "last_activity": "Sat Apr 21 2018 00:00:00 GMT-
0400 (Eastern Daylight Time)",
           "estimated_sales": "4575528",
           "actual_sales": "365964",
           "tags": "cold"
         },
         {
           "name": "Marlee Roote",
           "email": "mroote6v@vk.com",
           "company": "Skalith",
           "position": "Tax Accountant",
           "city": "Albany",
           "post_code": "12210",
           "state": "NY",
           "country": "United States",
           "created_on": "Sun Dec 17 2017 00:00:00 GMT-0500
(Eastern Standard Time)",
           "last_activity": "Mon Jan 01 2018 00:00:00 GMT-
```

```
0500 (Eastern Standard Time)",
            "estimated_sales": "120282",
            "actual_sales": "3000442",
            "tags": "Invalid Date"
          },
          {
            "name": "Jaine Schustl",
            "email": "jschustlav@utexas.edu",
            "company": "Gigazoom",
            "position": "Food Chemist",
            "city": "Albany",
            "post_code": "12232",
            "state": "NY",
            "country": "United States",
            "created_on": "Tue Sep 12 2017 00:00:00 GMT-0400
(Eastern Daylight Time)",
            "last_activity": "Thu Sep 28 2017 00:00:00 GMT-
0400 (Eastern Daylight Time)",
            "estimated_sales": "2002221",
            "actual_sales": "2596208",
            "tags": "pro"
          },
          {
            "name": "Kally Foux",
            "email": "kfoux9c@e-recht24.de",
            "company": "Jaxspan",
            "position": "Nurse",
            "city": "Albuquerque",
            "post_code": "87195",
            "state": "NM",
            "country": "United States",
            "created_on": "Thu Mar 29 2018 00:00:00 GMT-0400
(Eastern Daylight Time)",
            "last_activity": "Fri Mar 30 2018 00:00:00 GMT-
0400 (Eastern Daylight Time)",
            "estimated_sales": "1952070",
            "actual_sales": "1023200",
            "tags": "cold"
          }
      ];
    }
}
```

On the **`AppComponet`** template, you can add igx-grid as shown in *code listing 14.8*.

Code Listing 14.8

```
<igx-grid #grid1 id="grid1"
          [data]="localData"
          [autoGenerate]="true">
</igx-grid>
```

Bare minimum properties you need to set:

- Data: determines the DataSource of the igx-grid.
- Autogenerate: determines whether columns will be generated automatically or not. If it is set to false, you need to configure columns manually.

If everything works as expected, you should see a simple igx-grid rendered in the Angular application as shown in the *figure 14.1*.

Figure 14.1

Reading a Grid in the Component Class

There may be requirements to read the igx-grid in the component class.

That can be useful, in using **`igxGridComponent`** properties and events in the class. To do this, first import **`IgxGridComponent`** as shown in the *code listing 14.9.*

Code Listing 14.9

```
import { IgxGridComponent } from 'igniteui-angular';
```

Next, create **ViewChild** type local variable as shown in *code listing 14.10.*

Code Listing 14.10

```
   @ViewChild(,grid1', { read: IgxGridComponent })
   public grid: IgxGridComponent;
```

`gridl` is a template reference variable name. Now, you have reference of **`IgxGridComponent`** and you can use that in **`ngOnInit()`** or **ngAfterViewInit()** life cycle hook as shown in *code listing 14.11.*

Code Listing 14.11

```
   ngOnInit() {
      console.log(this.grid);
   }
```

If everything works as expected, you should see an **`IgxGridComponent`** in the browser console as shown *figure 14.2*.

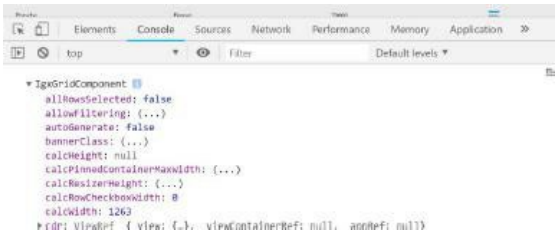

Figure 14.2

Configuring Columns

To configure columns manually, first set the **`AutoGenerate`** value property to false. You need to us**`e igx-column`** **or** **`IgxColumnComponent`** to configure the columns. To do that, modify the igx-grid as shown in *code listing 14.12.*

Code Listing 14.12

```
<igx-grid #grid1 id="grid1"
          [data]="localData"
          [autoGenerate]="false">
          <igx-column field="name" header="Name"></igx-
column>
          <igx-column field="email"  header="Email"></
igx-column>     figure
          <igx-column field="company"  header="Company"></
igx-column>
            <igx-column field="position" header="Position"></
igx-column>
            <igx-column field="city"  header="City"></igx-
column>
            <igx-column field="post_code"  header="Postal
Code"></igx-column>
            <igx-column field="state" header="State"></igx-
column>
            <igx-column field="country"  header="Country"></
igx-column>
            <igx-column field="created_on"  header="Created
On"></igx-column>
            <igx-column field="last_activity" header="Last
```

```
Activity"></igx-column>
            <igx-column field="estimated_sales"
 header="Estimated Sales"></igx-column>
            <igx-column field="actual_sales"  header="Actual
Sales"></igx-column>
            <igx-column field="tags"  header="Tags"></igx-
column>
</igx-grid>
```

Besides header and field properties, there are many other properties that can be set on `IgxColumnComponent`. We will cover that in further sections.

If everything works as expected, you should see a simple igx-grid rendered in Angular application as shown in the *figure 14.3*.

Figure 14.3

As you notice, now the header is changed. While configuring columns manually, you can opt to exclude certain columns. For example, you can emit the tags, **actual_sales** columns by excluding it in columns configuration as shown in the *code listing 14.13.*

Code Listing 14.13

```
<igx-grid #grid1 id="grid1"
            [data]="localData"
            [autoGenerate]="false">
            <igx-column field="name" header="Name"></igx-
column>
            <igx-column field="email"  header="Email"></
igx-column>
            <igx-column field="company"  header="Company"></
igx-column>
            <igx-column field="position" header="Position"></
igx-column>
            <igx-column field="city"  header="City"></igx-
column>
            <igx-column field="post_code"  header="Postal
Code"></igx-column>
            <igx-column field="state" header="State"></igx-
column>
            <igx-column field="country"  header="Country"></
igx-column>
            <igx-column field="created_on"  header="Created
On"></igx-column>
            <igx-column field="last_activity" header="Last
Activity"></igx-column>
               <igx-column field="estimated_sales"
header="Estimated Sales"></igx-column>
</igx-grid>
```

If everything works as expected, you should see a simple igx-grid rendered in Angular application as shown in the *figure 14.4*.

Figure 14.4

Creating Column Templates

While creating an enterprise level application, you may have to create custom templates for the columns. You need custom

templates to display other components or customize a design inside a column. For example, if you need to display a progress bar, a data chart, or an image inside a column, you can do that by creating a custom column template. **`IgxColumnComponent`** supports custom templates.

Let us modify the data source and add a new property rating as shown in *code listing 14.14.*

Code Listing 14.14

```
getData()   {
   return [
      {
        "name": "Leticia Grisewood",
     "email": "lgrisewoodbn@youtube.com",
     "company": "Innotype",
           "position": "Administrative Assistant III",
           "city": "Aiken",
           "post_code": "29805",
           "state": "SC",
           "country": "United States",
           "created_on": "Sat Mar 31 2018 00:00:00 GMT-0400
(Eastern Daylight Time)",
           "last_activity": "Tue Apr 24 2018 00:00:00 GMT-
0400 (Eastern Daylight Time)",
           "estimated_sales": "2026416",
           "actual_sales": "714549",
           "tags": "demo",
           "rating": "7"
         },
         {
           "name": "Roderic Gwilt",
           "email": "rgwilt6d@ox.ac.uk",
           "company": "Minyx",
           "position": "Developer IV",
           "city": "Akron",
           "post_code": "44393",
           "state": "OH",
           "country": "United States",
           "created_on": "Fri Apr 06 2018 00:00:00 GMT-0400
(Eastern Daylight Time)",
           "last_activity": "Sat Apr 21 2018 00:00:00 GMT-
0400 (Eastern Daylight Time)",
           "estimated_sales": "4575528",
           "actual_sales": "365964",
           "tags": "cold",
           "rating": "8"
         },
         {
           "name": "Marlee Roote",
           "email": "mroote6v@vk.com",
           "company": "Skalith",
           "position": "Tax Accountant",
           "city": "Albany",
           "post_code": "12210",
           "state": "NY",
           "country": "United States",
           "created_on": "Sun Dec 17 2017 00:00:00 GMT-0500
(Eastern Standard Time)",
           "last_activity": "Mon Jan 01 2018 00:00:00 GMT-
0500 (Eastern Standard Time)",
           "estimated_sales": "120282",
           "actual_sales": "3000442",
           "tags": "Invalid Date",
           "rating": "9"
         },
         {
           "name": "Jaine Schustl",
           "email": "jschustlav@utexas.edu",
           "company": "Gigazoom",
           "position": "Food Chemist",
           "city": "Albany",
           "post_code": "12232",
           "state": "NY",
           "country": "United States",
           "created_on": "Tue Sep 12 2017 00:00:00 GMT-0400
(Eastern Daylight Time)",
           "last_activity": "Thu Sep 28 2017 00:00:00 GMT-
```

```
0400 (Eastern Daylight Time)",
            "estimated_sales": "2002221",
            "actual_sales": "2596208",
            "tags": "pro",
            "rating": "7"
          },
          {
            "name": "Kally Foux",
            "email": "kfoux9c@e-recht24.de",
            "company": "Jaxspan",
            "position": "Nurse",
            "city": "Albuquerque",
            "post_code": "87195",
            "state": "NM",
            "country": "United States",
            "created_on": "Thu Mar 29 2018 00:00:00 GMT-0400
(Eastern Daylight Time)",
            "last_activity": "Fri Mar 30 2018 00:00:00 GMT-
0400 (Eastern Daylight Time)",
            "estimated_sales": "1952070",
            "actual_sales": "1023200",
            "tags": "cold",
            "rating": "6.5"
          }
       ];
      }
```

We wish to display ratings in a Linear Progress Bar. Ignite UI for Angular provides a Linear Progress Bar component. To use **`igx-linear-bar`** inside **`IgxColumnComponent`**, you need to create a template for the column, which can be done as shown in *code listing 14.15.*

Code Listing 14.15

```
<igx-column field="rating" header="Ratings">
            <ng-template igxCell let-value>
                  <igx-linear-bar  [value]="value"
[max]="10"></igx-linear-bar>
            </ng-template>
         </igx-column>
```

As you can see, we are using **`<ng-template>`** inside **`<igx-column>`**. Inside the template, **`<igx-linear-bar>`** is used. You can read the column value using the [value] property binding and passing *value* to it. Each of the grid columns can be templated separately. The column expects **`ng-template`** tags decorated with one of the grid module directives.

Keeping everting together, **`igx-grid`** with template column will look like the *code listing 14.16.*

Code Listing 14.16

```
<igx-grid #grid1 id="grid1" [data]="localData"
[autoGenerate]="false">
  <igx-column field="name" header="Name"></igx-column>
    <igx-column field="email" header="Email"></igx-
column>
    <igx-column field="company" header="Company"></igx-
column>
   <igx-column field="position" header="Position"></igx-
column>
    <igx-column field="city" header="City"></igx-column>
    <igx-column field="post_code" header="Postal Code"></
igx-column>
 <igx-column field="state" header="State"></igx-column>
   <igx-column field="country" header="Country"></igx-
column>
  <igx-column field="created_on" header="Created On"></
igx-column>
        <igx-column field="last_activity" header="Last
Activity"></igx-column>
    <igx-column field="estimated_sales" header="Estimated
Sales"></igx-column>
        <igx-column field="actual_sales" header="Actual
Sales"></igx-column>
    <igx-column field="tags" header="Tags"></igx-column>
```

```
    <igx-column field="rating" header="Ratings">
        <ng-template igxCell let-value>
           <igx-linear-bar [value]="value" [max]="10"></igx-
linear-bar>
       </ng-template>
     </igx-column>
</igx-grid>
```

If everything works as expected, you should see a simple igx-grid rendered in Angular application as shown in the *figure 14.5*.

Figure 14.5

Enable Pagination

Configuring the Ignite UI for Angular Grid for Pagination is very simple. To enable pagination, you just have to do property binding of **paging** and **`perPage`** properties of the **`igxGridComponent`** as shown in *code listing 14.17.*

Code Listing 14.17

```
<igx-grid #grid1 id="grid1" [data]="localData"
        [autoGenerate]="false"
        [paging]="true"
        [perPage]="5">
    <igx-column field="name" header="Name"></igx-column>
  <igx-column field="email" header="Email"></igx-column>
    <igx-column field="company" header="Company"></igx-
column>
   <igx-column field="position" header="Position"></igx-
column>
    <igx-column field="city" header="City"></igx-column>
   <igx-column field="post_code" header="Postal Code"></
igx-column>
  <igx-column field="state" header="State"></igx-column>
     <igx-column field="country" header="Country"></igx-
column>
  <igx-column field="created_on" header="Created On"></
igx-column>
     <igx-column field="last_activity" header="Last
Activity"></igx-column>
  <igx-column field="estimated_sales" header="Estimated
Sales"></igx-column>
     <igx-column field="actual_sales" header="Actual
Sales"></igx-column>
  <igx-column field="tags" header="Tags"></igx-column>
     <igx-column field="rating" header="Ratings">
        <ng-template igxCell let-value>
           <igx-linear-bar [value]="value" [max]="10"></igx-
linear-bar>
     </ng-template>
  </igx-column>
</igx-grid>
```

As you might have noticed in previous code snippets:

1. Set paging property to true.
2. To set per page rows, set a number to **perPage** property.

Also, to display pagination, let us update the data source with more rows. If everything works as expected, you should see a simple igx-grid rendered in the Angular application as shown in the *figure 14.6*.

Figure 14.6

Enable Sorting

In the Ignite UI for Angular Grid, **sorting** is enabled at the column level. Thus, you can have a mix of sort enabled and disabled columns. For the igx-grid, you can configure sorting at the igx-column level as shown in the *code listing 14.18.*

Code Listing 14.18

```
<igx-grid #grid1 id="grid1"
         [data]="localData"
           [autoGenerate]="false">
  <igx-column field="make" header="Maker Company"></igx-
column>
        <igx-column field="model" sortable='true'
header="Model"></igx-column>
  <igx-column field="price" sortable='true' header="Price
in USD"></igx-column>
        <igx-column field="rating" sortable='true'
header="Ratings">
     <ng-template igxCell let-value>
        <igx-linear-bar [value]="value" [max]="10"></igx-
linear-bar>
        </ng-template>
     </igx-column>
</igx-grid>
```

You have to set value of sortable to true, to configure columns for sorting.

As you can see, we have a combination of columns configured for sorting and not sorting.

If everything works as expected, you should see a simple igx-grid rendered in the Angular application as shown in the *figure 14.7*.

Figure 14.7

Some points you should keep in mind about sorting:

1. Sorting does not change the underlying data source
2. You can set the initial sorting state using the `sortingExpressions` property of the igx-grid
3. You can perform Remote Sorting as well
4. You can also use `IgxGridComponent` API to perform sorting.

Enabling Filtering

In the Ignite UI for Angular Grid, you can enable filtering by setting the **`allowFiltering`** property to true on **`<igx-grid>`**. You can set it following the *code listing 14.19*.

Code Listing 14.19

```
<igx-grid #grid1 id="grid1" [data]="localData"
        [autoGenerate]="false"
        [allowFiltering]="true">
  <igx-column field="make" header="Maker Company"></igx-
column>
  <igx-column field="model" header="Model"></igx-column>
  <igx-column field="price" header="Price in USD"></igx-
column>
  <igx-column field="rating" header="Ratings">
     <ng-template igxCell let-value>
        <igx-linear-bar [value]="value" [max]="10"></igx-
linear-bar>
        </ng-template>
     </igx-column>
</igx-grid>
```

If everything works as expected, you should see a simple igx-grid rendered in the Angular application as shown in the *figure 14.8*.

Figure 14.8

You can disable filtering at the column level by switching the **`filterable`** input to false. You can disable filtering for the Ratings column as shown in *code listing 14.20*.

Code Listing 14.20

```
<igx-grid #grid1 id="grid1" [data]="localData"
        [autoGenerate]="false"
        [allowFiltering]="true">
  <igx-column field="make" header="Maker Company"></igx-
column>
  <igx-column field="model" header="Model"></igx-column>
  <igx-column field="price" header="Price in USD"></igx-
column>
        <igx-column [filterable]="false" field="rating"
header="Ratings">
     <ng-template igxCell let-value>
       <igx-linear-bar [value]="value" [max]="10"></igx-
linear-bar>
        </ng-template>
     </igx-column>
</igx-grid>
```

You should see a simple igx-grid rendered in the Angular application as shown in the *figure 14.9*.

Figure 14.9

Summary

Now you have all you need to get started with the high performance igx-grid in an Angular application. In this tutorial, we covered configuring Ignite UI for Angular in a project, creating a grid, configuring columns, pagination, sorting, filtering.

In addition to these features, the grid also offers:

- Virtualization
- Editing
- Group By
- Summaries
- Column Moving
- Column Pinning
- Multi Column Headers

Configuring these features is also very easy. In this chapter, you learnt about the following:

- Introduction of Ignite UI for Angular
- Ignite UI for Angular Grid
- Add igx-grid to an Angular project
- Add igx-grid component
- Reading a Grid in the Component Class
- Configuring Columns
- Creating Column Templates
- Enable Pagination
- Enable Sorting
- Enable Filtering