In [1]:
```
#Importing packages
import warnings
import itertools
import numpy as np
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import pandas as pd
import statsmodels.api as sm
import matplotlib
matplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12
matplotlib.rcParams['text.color'] = 'k'
```

Importing data

In [11]:
```
#Reading the superstore dataset
df = pd.read_excel("Sample - Superstore.xls")
df.head(10)
```

Out[11]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	C
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Bo
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760	Office Supplies	
5	6	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	FUR-FU-10001487	Furniture	Fur
6	7	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	OFF-AR-10002833	Office Supplies	
7	8	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	TEC-PH-10002275	Technology	
8	9	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	OFF-BI-10003910	Office Supplies	
9	10	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	OFF-AP-10002892	Office Supplies	Ap

10 rows × 21 columns

In [16]:
```
#Looking at the summary statistics of the variables with numeric values
df.describe()
```

Out[16]:

	Row ID	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	4997.500000	55190.379428	229.858001	3.789574	0.156203	28.656896
std	2885.163629	32063.693350	623.245101	2.225110	0.206452	234.260108
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	7495.750000	90008.000000	209.940000	5.000000	0.200000	29.364000
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	8399.976000

In [17]:
```
#Looking at summary statistics of variables with non-numeric values
df.describe(include=['object'])
```

Out[17]:

	Order ID	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Region	Product ID	Category	Sub-Category	Product Name
count	9994	9994	9994	9994	9994	9994	9994	9994	9994	9994	9994	9994	9994
unique	5009	4	793	793	3	1	531	49	4	1862	3	17	1850
top	CA-2017-100111	Standard Class	WB-21850	William Brown	Consumer	United States	New York City	California	West	OFF-PA-10001970	Office Supplies	Binders	Staple envelope
freq	14	5968	37	37	5191	9994	915	2001	3203	19	6026	1523	48

We can see from the dataset and the summary statistics that the superstore sells three types of products: Office Supplies, Technology and Furniture. Let us only consider furniture products for now.

In [21]:
```
furniture = df.loc[df['Category'] == "Furniture"]
furniture.head()
```

Out[21]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Ca
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Boo
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	
5	6	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	FUR-FU-10001487	Furniture	Furr
10	11	CA-2014-115812	2014-06-09	2014-06-14	Standard Class	BH-11710	Brosina Hoffman	Consumer	United States	Los Angeles	...	90032	West	FUR-TA-10001539	Furniture	

5 rows × 21 columns

In [22]:
```
#Looking at the first and the last order date for a furniture product
furniture['Order Date'].min(), furniture['Order Date'].max()
```

Out[22]:
```
(Timestamp('2014-01-06 00:00:00'), Timestamp('2017-12-30 00:00:00'))
```

Thus we now have a good 4-year furniture sales data

Data preprocessing

In time series forecasting of sales, we are only concerned with the order date and the sales value of the product. Hence, we can remove the rest of the columns.

In [41]:
```
#Taking a new dataframe with only Order Date and Sales columns.
furniture_new = furniture[['Order Date', 'Sales']]
#Sorting the dataframe by order date
```

```
furniture = furniture.sort_values('Order Date')
#Inspecting for missing values
furniture.isnull().sum()
```

Out[41]:
```
Order Date    0
Sales         0
dtype: int64
```

Let us aggregate the sales data by the order date.

In [47]:
```
furniture1 = furniture_new.groupby('Order Date')['Sales'].sum().reset_index()
```

In [55]:
```
#Looking at the dataframe
furniture1.head(10)
```

Out[55]:

	Order Date	Sales
0	2014-01-06	2573.820
1	2014-01-07	76.728
2	2014-01-10	51.940
3	2014-01-11	9.940
4	2014-01-13	879.939
5	2014-01-14	61.960
6	2014-01-16	127.104
7	2014-01-19	181.470
8	2014-01-20	1413.510
9	2014-01-21	25.248

In [57]:
```
#Setting Order Date as index
furniture1 = furniture1.set_index('Order Date')
furniture1.index
```

Out[57]:
```
DatetimeIndex(['2014-01-06', '2014-01-07', '2014-01-10', '2014-01-11',
               '2014-01-13', '2014-01-14', '2014-01-16', '2014-01-19',
               '2014-01-20', '2014-01-21',
               ...
               '2017-12-18', '2017-12-19', '2017-12-21', '2017-12-22',
               '2017-12-23', '2017-12-24', '2017-12-25', '2017-12-28',
               '2017-12-29', '2017-12-30'],
              dtype='datetime64[ns]', name='Order Date', length=889, freq=None)
```

We will use average daily sales value for a month to work with the dataframe. We will use start of each month as the timestamp.

In [85]:
```
y = furniture['Sales'].resample('MS').mean()
y.head()
```

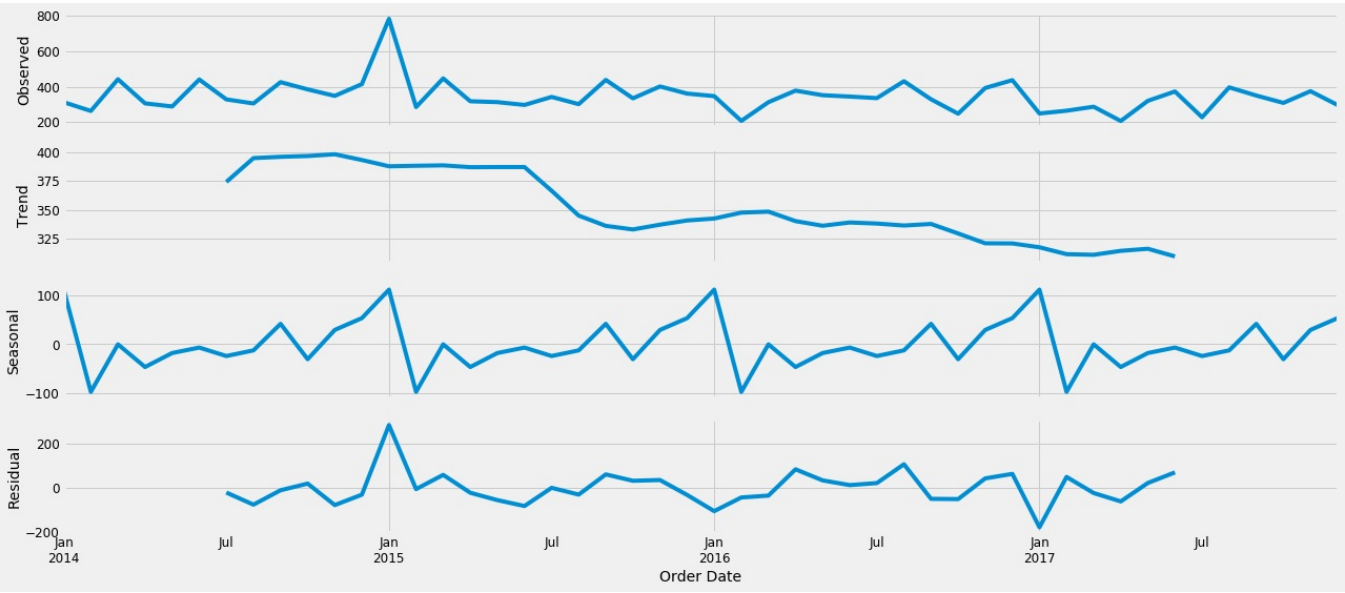
Out[85]:
```
206.04450227272727
Order Date
2014-01-01    312.126250
2014-02-01    262.808286
2014-03-01    441.635030
2014-04-01    305.570654
2014-05-01    288.032792
Freq: MS, Name: Sales, dtype: float64
```

Visualizing furniture sales data

In [62]:
```
y.plot(figsize=(15, 6))
plt.show()
```

We see a seasonal pattern in the furniture sales at the superstore. The sales fall at the beginning of the year and increase by the end of the year. There is an upward trend in furniture sales over a year with some dips in the mid-year period.

In [64]:
```
from pylab import rcParams
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
fig = decomposition.plot()
plt.show()
```



Repeating patterns can be seen in the 'Seasonal' graph indicating a seasonal trend in furniture sales. We can see that furniture sales are unstable in nature.

Time series forecasting with ARIMA

We use Grid Search method to find optimal set of model parameters yielding best performance

In [66]:
```
#Setting seasonality, trend and noise parameters
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12)- for x in list(itertools.product(p, d, q))]
```

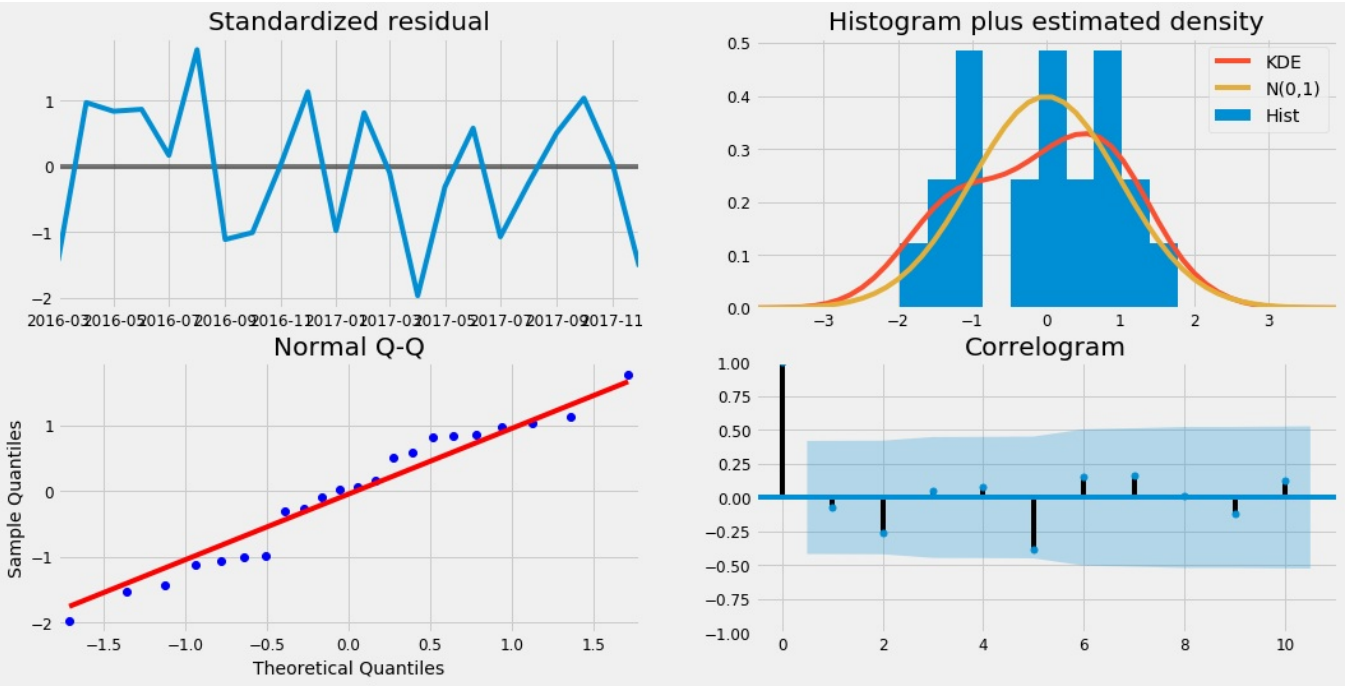
In [71]:
```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y, order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)
            results = mod.fit()
            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:688.3112416991725
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:1280.1520441779817
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:426.20464898591723
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:452.8238602501692
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:1387.6836464199887
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:295.446456472235
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:641.1240680665852
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:2692.4087255633453
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:416.5496453364602
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:455.11628880659146
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:130042.21621256301
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:297.446406625009
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:580.6389269668506
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:1128.5891633615913
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:433.35521367147123
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:437.46040107889746
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:1351.3219110378925
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:294.1158452476943
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:538.8964721447202
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:2612.5561401839586
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:412.8726989904938
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:419.1839792724931
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:2715.9299413497197
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:273.4032454636512
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:592.7661498749849
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:1157.1249947711885
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:427.7241159738327
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:420.41145389920376
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:1574.7909219615644
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:273.58403955659367
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:553.271941780426
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:3176.9407647283633
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:418.5261783089145
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:422.40035833231576
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:2812.7879703685544
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:275.4775740530797
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:566.5211872508684
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:1376.1438169992553
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:430.3305770817938
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:397.36689816515684
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:1447.5880143218897
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:273.4415398792928
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:540.6257368397363
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:3208.750786473174
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:414.7726173041459
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:391.83032556451644
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:2524.070305286992
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:263.9371084381272
```

Thus ARIMA(1, 1, 1)x(1, 1, 0, 12) yields lowest AIC value of 263.9371084381272. Hence, we should consider it to be optimal solution. Now let us fit this model.

In [73]:
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 0, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
results.plot_diagnostics(figsize=(16, 8))
plt.show()
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0771      0.259     -0.298      0.766      -0.584       0.430
ma.L1         -1.0000      0.434     -2.305      0.021      -1.850      -0.150
ar.S.L12      -0.0007      0.029     -0.025      0.980      -0.057       0.056
sigma2      6323.9742   6.86e-05   9.22e+07      0.000    6323.974    6323.974
==============================================================================
```

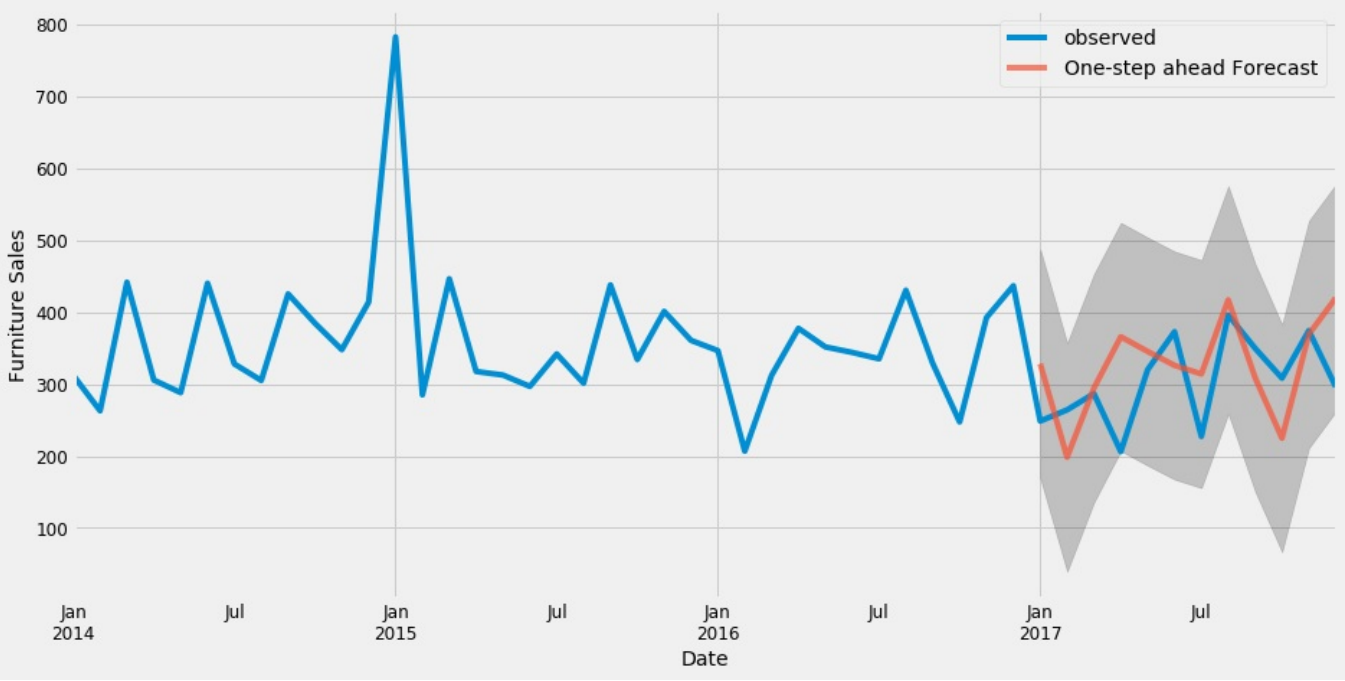


Looking at the model diagnostics, we can see that model residuals are close to normally distributed.

Model validation

To understand the model accuracy, we will predict sales values from the year 2017. We will then compare it to actual sales data from 2017 and find out the accuracy.

In [74]:
```
pred = results.get_prediction(start=pd.to_datetime('2017-01-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2014':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('Furniture Sales')
plt.legend()
plt.show()
```



The forecast also predicts an upward trend in furniture sales over the year. Let us look at the RMSE value.

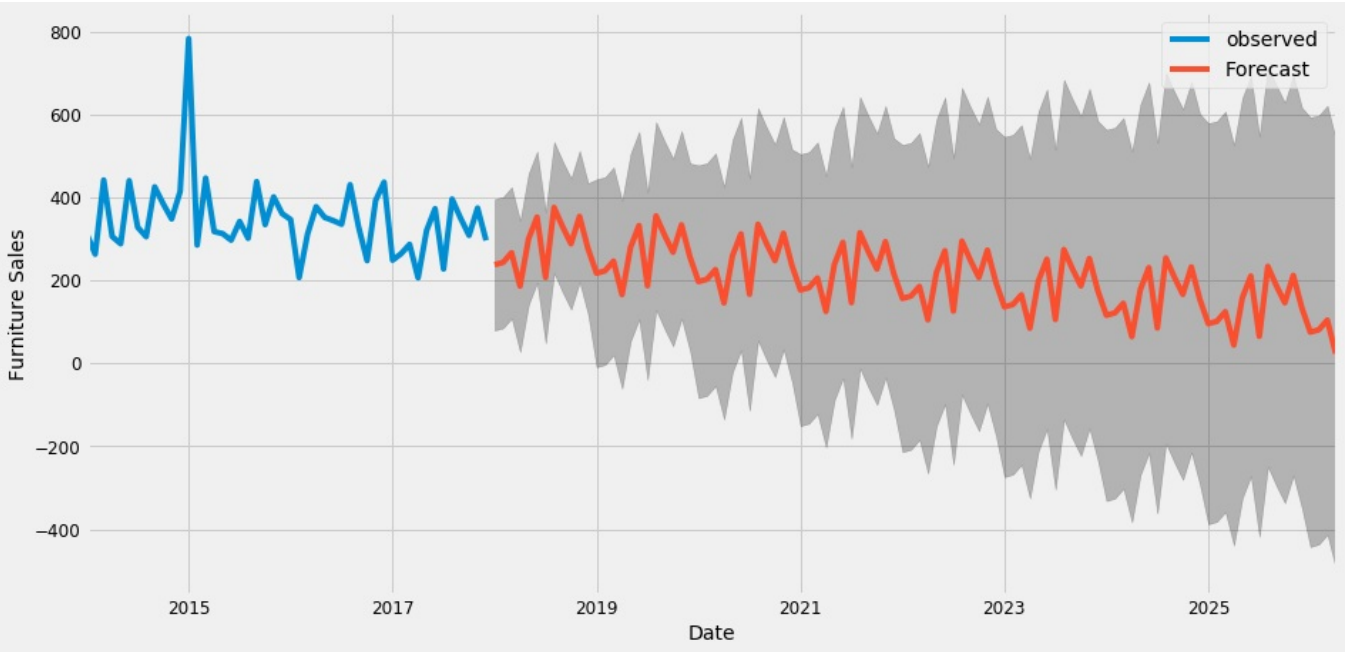
In [82]:
```
y_forecasted = pred.predicted_mean
y_truth = y['2017-01-01':]
mse = ((y_forecasted - y_truth) ** 2).mean()
print(mse)
rmse = format(round(np.sqrt(mse), 2))
print(rmse)
```

```
5911.261145885236
76.88
```

The model was able to forecast the average daily furniture sales in the test set within 76.88 of the real sales. Hence, it is a pretty good model to build further upon.

Forecasting future sales

In [86]:
```
pred_uc = results.get_forecast(steps=100)
pred_ci = pred_uc.conf_int()
ax = y.plot(label='observed', figsize=(14, 7))
pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_xlabel('Date')
ax.set_ylabel('Furniture Sales')
plt.legend()
plt.show()
```



The model forecasts similar seasonality in furniture sales in the future. The confidence interval increases as we move further into the

future.