In [1]:
```
# Import all the necessary packages:
import pandas as pd
import statistics
import numpy as np
import math
import csv
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
```

Part 1: Data processing

In [2]:
```
# Load the data, use first column as the index of the dataframe
df=pd.read_csv("data/CompleteDataset.csv",index_col=0)
```

```
D:\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2785: DtypeWarning: C
olumns (23,35) have mixed types. Specify dtype option on import or set low_memory=Fal
se.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:
```
# Clean data:
# Conversion of calculations to its integer results easier for further processing.
# Filling of null space by 0.
# Conversion of float numbers to integers

#Function for cleaning:
def clean(x):
    # evaluate sum
    if('+' in str(x).strip()):
        cal = x.split('+')
        return int(cal[0]) + int(cal[1])
    # evaluate subtraction
    elif('-' in str(x).strip()):
        cal = x.split('-')
        return int(cal[0]) + int(cal[1])
    # convert to all float points to integer
    elif str(x).strip().isdigit():
        return int(x)
    # return as 0 if null values or NaN values
    elif pd.isnull(x):
         return 0
    # return other type (e.g. object) as the way it is.
    else:
        return x
# Clean the columns of interest
for column in df.iloc[:,11:74]:
    df[column] = df[column].apply(clean)
for column in df.iloc[:,5:7]:
    df[column] = df[column].apply(clean)
# Save the modfied dataframe
df.to_csv("data/CompleteDataset_clean.csv")
```

In [4]:
```
# Read data from cleaned dataset
filename = "data/CompleteDataset_clean.csv"
df = pd.read_csv(filename)
```

In [5]:
```
# Extract columns of interest:player performance attributes
# (Overall, Finishing, Dribbling, etc.) and player preferred
# subdivided positions (CF, CM, CB, etc.).

columns_needed = ['Acceleration', 'Aggression', 'Agility', 'Balance', 'Ball control',
       'Composure', 'Crossing', 'Curve', 'Dribbling', 'Finishing',
       'Free kick accuracy', 'Heading accuracy', 'Interceptions',
       'Jumping', 'Long passing', 'Long shots', 'Marking', 'Penalties',
       'Positioning', 'Reactions', 'Short passing', 'Shot power',
       'Sliding tackle', 'Sprint speed', 'Stamina', 'Standing tackle',
       'Strength', 'Vision', 'Volleys', 'Overall', 'Value', 'Preferred Positions']
df = df[columns_needed]
```

In [6]:
```
# Assign a preferred postion to each player:
# Each playerat has multiple subdivided under'Preferred Positions' , the first one is s
elected as target.

df['Target_Position'] = df['Preferred Positions'].str.split().str[0]
df = df.drop('Preferred Positions', 1)
# Transform target position into number representation in which
# 0 to 3 represents four categories of positions:
# Forward(0), Midfield(1), Defender(2) and Goalkeeper(3).

mapping = {'CF': 0,'ST': 0, 'RW': 0, 'LW': 0, 'RM': 1, 'CM': 1, 'LM': 1, 'CAM': 1, 'CD
M': 1, 'CB': 2, 'LB': 2, 'RB': 2, 'RWB': 2, 'LWB': 2, 'GK': 3}
df = df.replace({'Target_Position': mapping})
df.isnull().values.any()
```

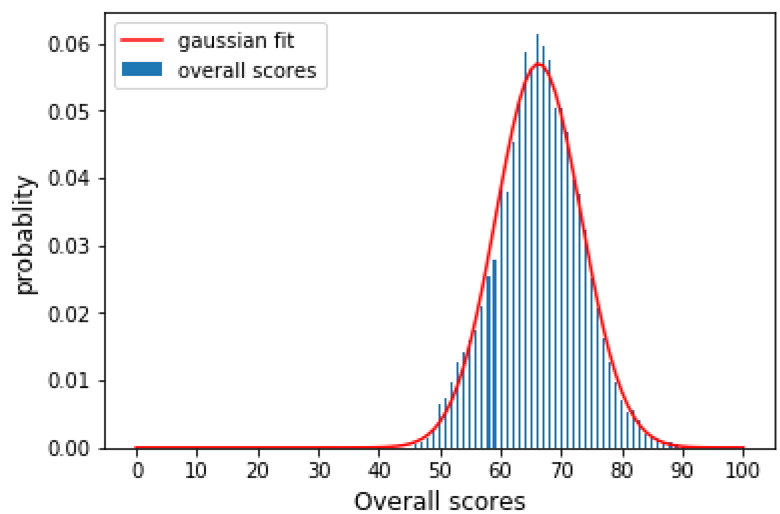
Out[6]:
```
False
```

Part 2: Distribution of Overall scores

In [7]:
```
#simple statistics for overall scores
overall=df.iloc[:,29]
print(overall.describe())
```

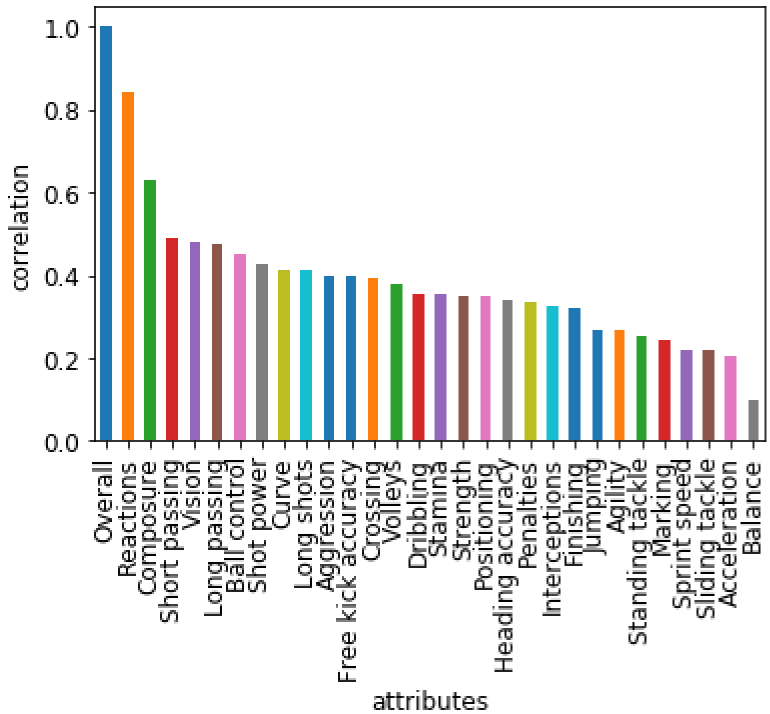
```
count    17981.000000
mean        66.247984
std          6.987965
min         46.000000
25%         62.000000
50%         66.000000
75%         71.000000
max         94.000000
Name: Overall, dtype: float64
```

In [8]:
```
# Plot the overall distribution
# Fit a Gaussian function onto the plot to show the distribution can be approximated as
 a normal distribution.
x=np.linspace(0,100,101)
y=np.array([0]*101)
for i in range(0,len(overall)):
      y[overall[i]]+=1
summation=overall.value_counts().sum()
y_prob=np.zeros(101)
for i in range(0,len(y)):
    y_prob[i] =y[i]*1.0/summation

mean=66.25
variance=49
x=np.linspace(0.0,100.0,101)
y_gaus=np.zeros(101)
for i in range(0,len(y)):
    y_gaus[i]=1/(math.sqrt(2*math.pi*variance))*np.exp(-((i-mean)**2)/(2*variance))


plt.bar(x,y_prob,width=0.35,label='overall scores')
plt.plot(x,y_gaus,'r',label='gaussian fit')
plt.legend()
plt.ylabel('probablity',fontsize=12)
plt.xlabel('Overall scores',fontsize=12);
plt.xticks(np.arange(0,101,10))
plt.savefig('overall score distribution gaussian fitted.pdf',bbox_inches = 'tight')
```



Part 3: Correlation between Overall scores and attributes

In [9]:
```
attributes=np.concatenate([np.array([29]),np.arange(0,29)])
df_performance=df.iloc[:,attributes]
corr=df_performance.corr()
overall_corr=corr.iloc[0,:].sort_values(ascending=False)
overall_corr.plot.bar(fontsize=12)
plt.ylabel("correlation",fontsize=12)
plt.xlabel("attributes",fontsize=12)
plt.savefig('correlation of attributes.pdf',bbox_inches = 'tight')
```



Predict Overall -- linear regression

In [15]:
```
df_F = df[df.Target_Position == 0]
df_M = df[df.Target_Position == 1]
df_D = df[df.Target_Position == 2]
df_G = df[df.Target_Position == 3]
position_names = ["Mixed", "Forward", "Midfielder", "Defender", "Goalkeeper"]
df_list = [df, df_F, df_M, df_D, df_G]

for i,t_name in enumerate(position_names):
    df_t = df_list[i]
    print("#", t_name,"\t", df_t.shape[0])
```

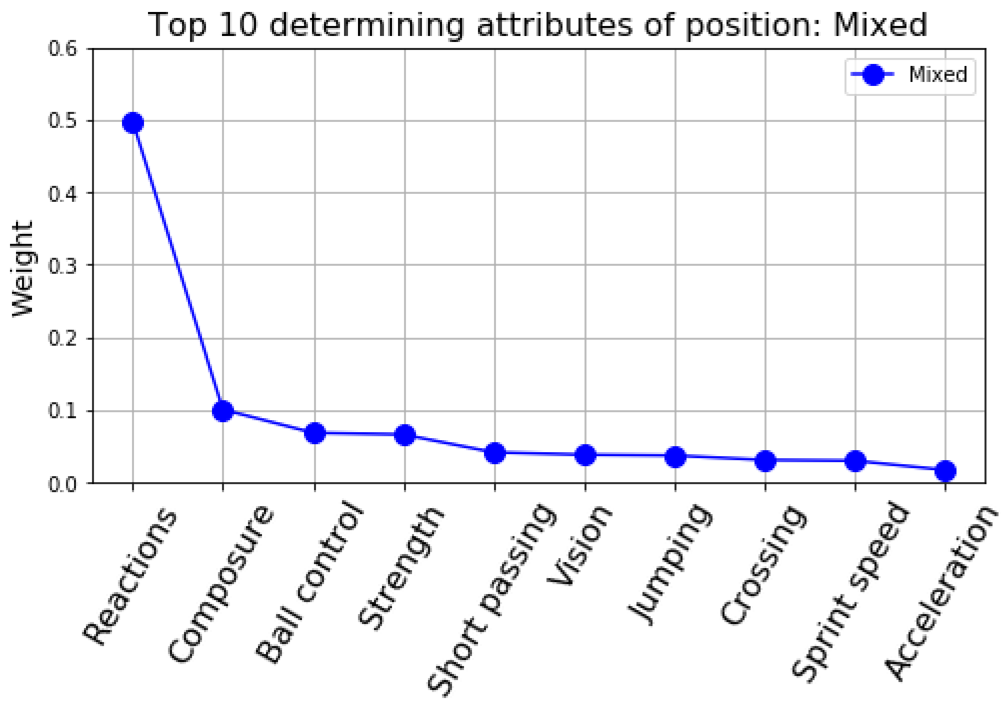
```
# Mixed         17981
# Forward       3338
# Midfielder    7174
# Defender      5440
# Goalkeeper    2029
```

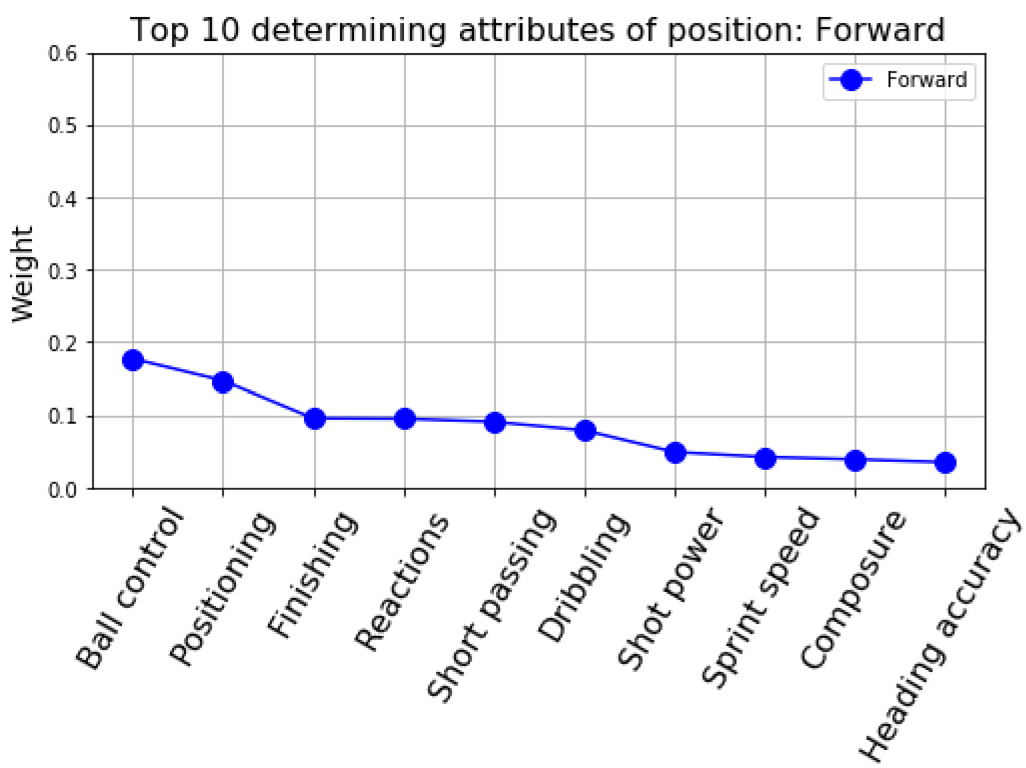
In [16]:
```
data_to_plot = [[] for i in range(5)]

# select which types of position
for i,t_name in enumerate(position_names):
    print("\n")
    print(t_name)
    print('Preparing data for training and testing...')

    df_sin = df_list[i]
    X_train, X_test, y_train, y_test = train_test_split(df_sin.iloc[:,:-3], df_sin.iloc
[:,-3], test_size=0.1, random_state=0)
    print('X train shape: {}'.format(X_train.shape))
    print('X test shape: {}'.format(X_test.shape))
    print('y train shape: {}'.format(y_train.shape))
    print('y test shape: {}'.format(y_test.shape))

    clf = LinearRegression().fit(X_train, y_train)
    acc = clf.score(X_test, y_test)
    print ('Linear Regression R^2 coef: {}'.format(acc))

    feature_result = [(X_train.columns[i], clf.coef_[i]) for i in range(len(clf.coef_
))]
    feature_result_sort = sorted(feature_result, key = lambda x : x[1], reverse = True
)
    for pair in (feature_result_sort[:10]):
        print(pair)

    data_to_plot[i] = feature_result_sort[:10]
```

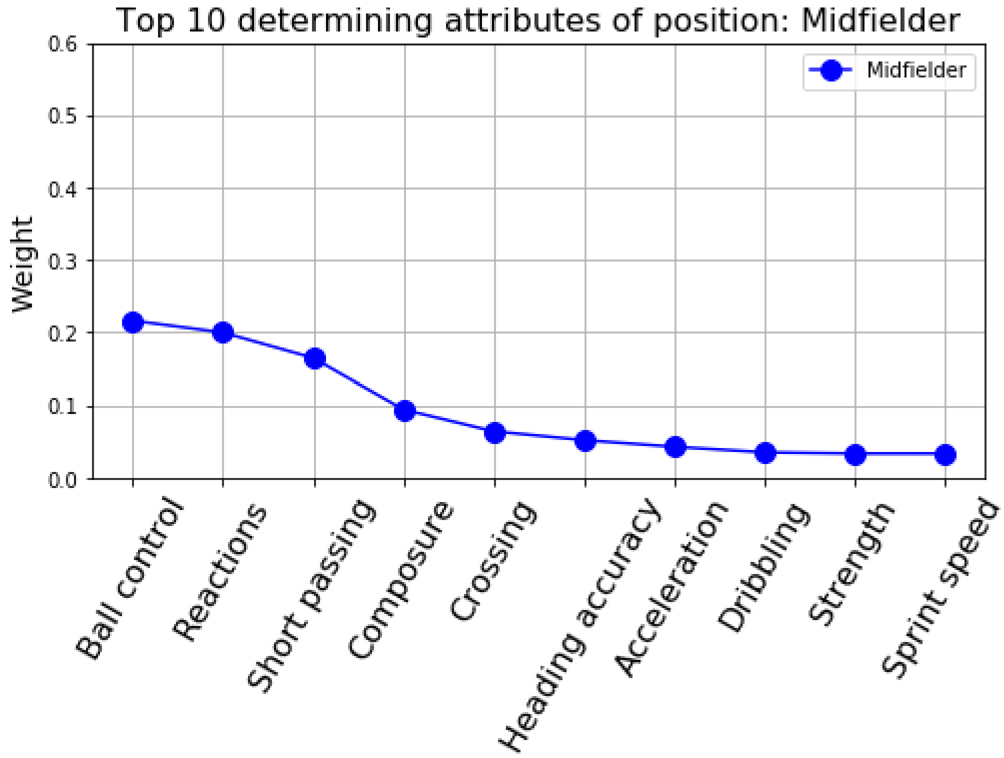
```
Mixed
Preparing data for training and testing...
X train shape: (16182, 29)
X test shape: (1799, 29)
y train shape: (16182,)
y test shape: (1799,)
Linear Regression R^2 coef: 0.7700273879214501
('Reactions', 0.4976124208480736)
('Composure', 0.10070997666906735)
('Ball control', 0.06857163291770287)
('Strength', 0.06624314222754851)
('Short passing', 0.04159958257420677)
('Vision', 0.03829656644653301)
('Jumping', 0.03725527572093278)
('Crossing', 0.031012417321654773)
('Sprint speed', 0.03005699867511938)
('Acceleration', 0.017230757274359555)


Forward
Preparing data for training and testing...
X train shape: (3004, 29)
X test shape: (334, 29)
y train shape: (3004,)
y test shape: (334,)
Linear Regression R^2 coef: 0.9705404535361363
('Ball control', 0.17732490401785622)
('Positioning', 0.1473182562638019)
('Finishing', 0.09569358044454984)
('Reactions', 0.09518441428884088)
('Short passing', 0.09096897079926951)
('Dribbling', 0.07914394954146871)
('Shot power', 0.04927934592523177)
('Sprint speed', 0.04239753494045078)
('Composure', 0.03922582973524693)
('Heading accuracy', 0.03519380255840841)


Midfielder
Preparing data for training and testing...
X train shape: (6456, 29)
X test shape: (718, 29)
y train shape: (6456,)
y test shape: (718,)
Linear Regression R^2 coef: 0.8998457216695079
('Ball control', 0.216272682862331)
('Reactions', 0.1999465502661954)
('Short passing', 0.16494574355280603)
('Composure', 0.09431280906348814)
('Crossing', 0.06431197585008698)
('Heading accuracy', 0.052081537089898294)
('Acceleration', 0.04292410002580306)
('Dribbling', 0.03527165446993157)
('Strength', 0.03362792762771143)
('Sprint speed', 0.03358937644197347)
```

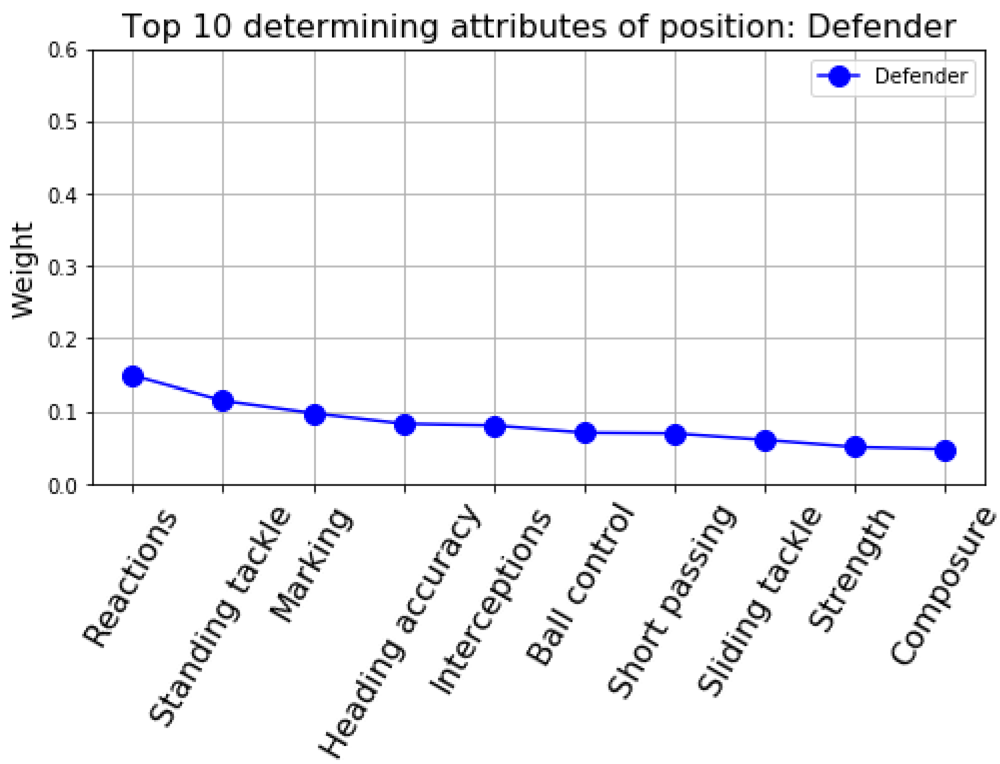
```
Defender
Preparing data for training and testing...
X train shape: (4896, 29)
X test shape: (544, 29)
y train shape: (4896,)
y test shape: (544,)
Linear Regression R^2 coef: 0.9330976484982253
('Reactions', 0.1490617719761642)
('Standing tackle', 0.11473428250519126)
('Marking', 0.0975329854790554)
('Heading accuracy', 0.08303701553291024)
('Interceptions', 0.08070460888703632)
('Ball control', 0.07061401385512571)
('Short passing', 0.06970442249698793)
('Sliding tackle', 0.06076803187412069)
('Strength', 0.05091871316557301)
('Composure', 0.047896292381827694)


Goalkeeper
Preparing data for training and testing...
X train shape: (1826, 29)
X test shape: (203, 29)
y train shape: (1826,)
y test shape: (203,)
Linear Regression R^2 coef: 0.7845672544274247
('Reactions', 0.5366588288391403)
('Volleys', 0.06712244740434369)
('Vision', 0.06231215527577779)
('Jumping', 0.04640943358575749)
('Dribbling', 0.03916041900406958)
('Interceptions', 0.027091080968377982)
('Composure', 0.025111832217296296)
('Strength', 0.023664130204054907)
('Ball control', 0.02301121733730714)
('Sprint speed', 0.022350000799053284)
```

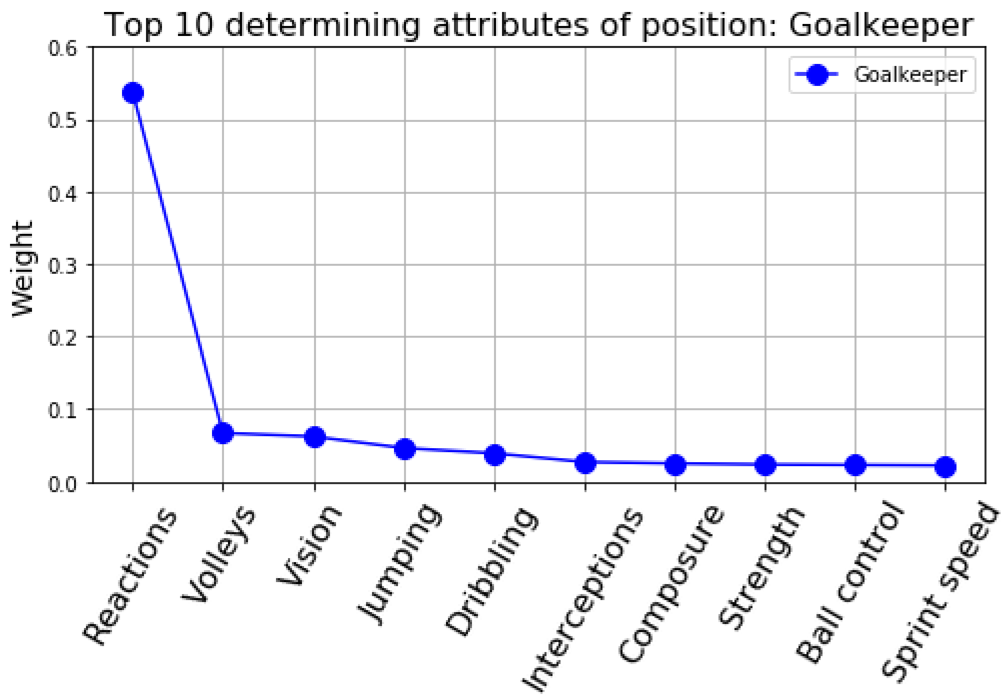
In [17]:
```
%matplotlib inline

for i,t_name in enumerate(position_names):

    name_list = [p[0] for p in data_to_plot[i]]
    coef_list = [p[1] for p in data_to_plot[i]]

    plt.figure(figsize=(8,4))
    plt.plot(name_list, coef_list, 'b-o', markersize=10, label=t_name)
    plt.ylim([0,0.6])
    plt.title("Top 10 determining attributes of position: " + t_name, fontsize=16)
    plt.ylabel('Weight', fontsize=14)
    plt.legend()
    plt.grid(True)
    plt.xticks(fontsize=16, rotation=60)
    plt.savefig('fig/top10'+t_name+'.pdf',bbox_inches = 'tight')
```


Top 10 determining attributes of position: Mixed
Mixed
Weight
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Reactions
Composure
Ball control
Strength
Short passing
Vision
Jumping
Crossing
Sprint speed
Acceleration


Top 10 determining attributes of position: Forward
Forward
Weight
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Ball control
Positioning
Finishing
Reactions
Short passing
Dribbling
Shot power
Sprint speed
Composure
Heading accuracy


Top 10 determining attributes of position: Midfielder
Midfielder
Weight
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Ball control
Reactions
Short passing
Composure
Crossing
Heading accuracy
Acceleration
Dribbling
Strength
Sprint speed


Top 10 determining attributes of position: Defender
Defender
Weight
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Reactions
Standing tackle
Marking
Heading accuracy
Interceptions
Ball control
Short passing
Sliding tackle
Strength
Composure



Classify position -- logistic regression

In [18]:

```
X_train, X_test, y_train, y_test = train_test_split(df.iloc[:,:-3], df.iloc[:,-1], test
_size=0.1, random_state=0)

print('X train shape: {}'.format(X_train.shape))
print('X test shape: {}'.format(X_test.shape))
print('y train shape: {}'.format(y_train.shape))
print('y test shape: {}'.format(y_test.shape))
```

```
X train shape: (16182, 29)
X test shape: (1799, 29)
y train shape: (16182,)
y test shape: (1799,)
```

In [19]:

```
clf = LogisticRegression().fit(X_train, y_train)
acc = clf.score(X_test, y_test)
print ('Logistic Regression accuracy: {}'.format(acc))
```

```
Logistic Regression accuracy: 0.8132295719844358
```

In [ ]: