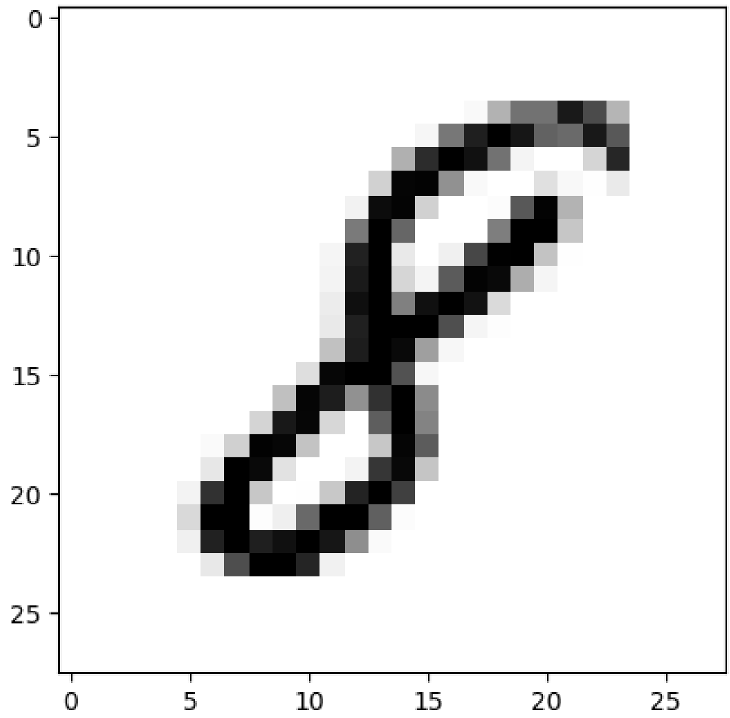
In [2]:
```
import tensorflow as tf
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mn
ist.npz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz)
11490434/11490434 [==============================] - 0s 0us/step
```

In [4]:
```
import matplotlib.pyplot as plt
%matplotlib inline
image_index = 7777 # You may select anything up to 60,000
print(y_train[image_index]) # The label is 8
plt.imshow(x_train[image_index], cmap='Greys')
```

```
8
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x7f2d7a5cca90>
```



In [5]:
```
x_train.shape
```

Out[5]:
```
(60000, 28, 28)
```

In [6]:
```
# Reshaping the array to 4-dims so that it can work with the Keras API
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
# Making sure that the values are float so that we can get decimal points after divis
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
# Normalizing the RGB codes by dividing it to the max RGB value.
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print('Number of images in x_train', x_train.shape[0])
print('Number of images in x_test', x_test.shape[0])
```

```
x_train shape: (60000, 28, 28, 1)
Number of images in x_train 60000
Number of images in x_test 10000
```

In [8]:
```
# Importing the required Keras modules containing model and layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
# Creating a Sequential Model and adding the layers
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten()) # Flattening the 2D arrays for fully connected layers
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(10,activation=tf.nn.softmax))
```

In [9]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x=x_train,y=y_train, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 18s 3ms/step - loss: 0.2137 - accuracy:
0.9359
Epoch 2/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0872 - accuracy:
0.9732
Epoch 3/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0601 - accuracy:
0.9810
Epoch 4/10
1875/1875 [==============================] - 7s 3ms/step - loss: 0.0471 - accuracy:
0.9848
Epoch 5/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0378 - accuracy:
0.9878
Epoch 6/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0311 - accuracy:
0.9898
Epoch 7/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0247 - accuracy:
0.9918
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0241 - accuracy:
0.9918
Epoch 9/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0193 - accuracy:
0.9938
Epoch 10/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0174 - accuracy:
0.9941
```

Out[9]:
```
<keras.callbacks.History at 0x7f2d600e5c10>
```

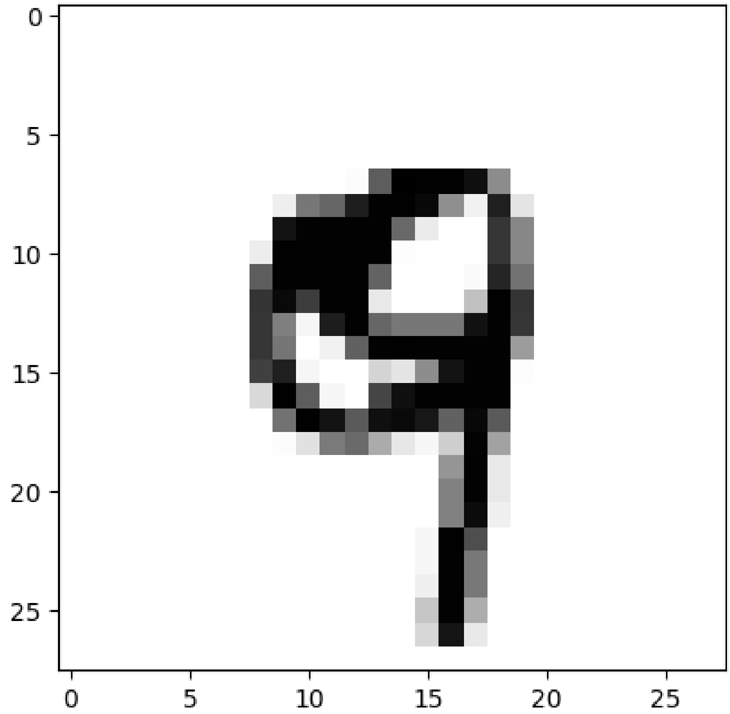
In [10]:
```
model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0670 - accuracy: 0.
9836
```

Out[10]:
```
[0.06704243272542953, 0.9836000204086304]
```

In [11]:
```
image_index = 4444
plt.imshow(x_test[image_index].reshape(28, 28),cmap='Greys')
pred = model.predict(x_test[image_index].reshape(1, 28, 28, 1))
print(pred.argmax())
```

```
1/1 [==============================] - 0s 104ms/step
9
```



In [12]:
```
image_index = 3625
plt.imshow(x_test[image_index].reshape(28, 28),cmap='Greys')
pred = model.predict(x_test[image_index].reshape(1, 28, 28, 1))
print(pred.argmax())
```

```
1/1 [==============================] - 0s 36ms/step
1
```