tf.feature_column.input_layer

```
input_layer(
    features,
    feature_columns,
    weight_collections=None,
    trainable=True
)
```

Defined in `tensorflow/python/feature_column/feature_column.py`.

Returns a dense `Tensor` as input layer based on given `feature_columns`.

Generally a single example in training data is described with FeatureColumns. At the first layer of the model, this column oriented data should be converted to a single `Tensor`.

Example:

```
price = numeric_column('price')
keywords_embedded = embedding_column(
    categorical_column_with_hash_bucket("keywords", 10K), dimensions=16)
columns = [price, keywords_embedded, ...]
features = tf.parse_example(..., features=make_parse_example_spec(columns))
dense_tensor = input_layer(features, columns)
for units in [128, 64, 32]:
  dense_tensor = tf.layers.dense(dense_tensor, units, tf.nn.relu)
prediction = tf.layers.dense(dense_tensor, 1)
```

Args:

- `features`: A mapping from key to tensors. `_FeatureColumn`s look up via these keys. For example `numeric_column('price')` will look at 'price' key in this dict. Values can be a `SparseTensor` or a `Tensor` depends on corresponding `_FeatureColumn`.
- `feature_columns`: An iterable containing the FeatureColumns to use as inputs to your model. All items should be instances of classes derived from `_DenseColumn` such as `numeric_column`, `embedding_column`, `bucketized_column`, `indicator_column`. If you have categorical features, you can wrap them with an `embedding_column` or `indicator_column`.
- `weight_collections`: A list of collection names to which the Variable will be added. Note that, variables will also be added to collections `tf.GraphKeys.GLOBAL_VARIABLES` and `ops.GraphKeys.MODEL_VARIABLES`.
- `trainable`: If `True` also add the variable to the graph collection `GraphKeys.TRAINABLE_VARIABLES` (see `tf.Variable`).

Returns:

A `Tensor` which represents input layer of a model. Its shape is (batch_size, first_layer_dimension) and its dtype is `float32`. first_layer_dimension is determined based on given `feature_columns`.

Raises:

- `ValueError`: if an item in `feature_columns` is not a `_DenseColumn`.

Last updated November 2, 2017.