tf.train.SingularMonitoredSession

Class **SingularMonitoredSession**

Defined in `tensorflow/python/training/monitored_session.py`.

See the guide: Training > Distributed execution

Session-like object that handles initialization, restoring, and hooks.

Please note that this utility is not recommended for distributed settings. For distributed settings, please use `tf.train.MonitoredSession`. The differences between `MonitoredSession` and `SingularMonitoredSession` are:

- `MonitoredSession` handles `AbortedError` and `UnavailableError` for distributed settings, but `SingularMonitoredSession` does not.
- `MonitoredSession` can be created in `chief` or `worker` modes. `SingularMonitoredSession` is always created as `chief`.
- You can access the raw `tf.Session` object used by `SingularMonitoredSession`, whereas in MonitoredSession the raw session is private. This can be used:
  - To `run` without hooks.
  - To save and restore.
- All other functionality is identical.

Example usage:

```
saver_hook = CheckpointSaverHook(...)
summary_hook = SummarySaverHook(...)
with SingularMonitoredSession(hooks=[saver_hook, summary_hook]) as sess:
  while not sess.should_stop():
    sess.run(train_op)
```

Initialization: At creation time the hooked session does following things in given order:

- calls `hook.begin()` for each given hook
- finalizes the graph via `scaffold.finalize()`
- create session
- initializes the model via initialization ops provided by `Scaffold`
- restores variables if a checkpoint exists
- launches queue runners

Run: When `run()` is called, the hooked session does following things:

- calls `hook.before_run()`
- calls TensorFlow `session.run()` with merged fetches and feed_dict
- calls `hook.after_run()`
- returns result of `session.run()` asked by user

Exit: At the `close()`, the hooked session does following things in order:

- calls `hook.end()`
- closes the queue runners and the session
- suppresses `OutOfRange` error which indicates that all inputs have been processed if the `SingularMonitoredSession` is used as a context.

Properties

graph

The graph that was launched in this session.

Methods

__init__

```
__init__(
    hooks=None,
    scaffold=None,
    master='',
    config=None,
    checkpoint_dir=None,
    stop_grace_period_secs=120,
    checkpoint_filename_with_path=None
)
```

Creates a SingularMonitoredSession.

Args:

- `hooks`: An iterable of `SessionRunHook' objects.
- `scaffold`: A `Scaffold` used for gathering or building supportive ops. If not specified a default one is created. It's used to finalize the graph.
- `master`: `String` representation of the TensorFlow master to use.
- `config`: `ConfigProto` proto used to configure the session.
- `checkpoint_dir`: A string. Optional path to a directory where to restore variables.
- `stop_grace_period_secs`: Number of seconds given to threads to stop after `close()` has been called.
- `checkpoint_filename_with_path`: A string. Optional path to a checkpoint file from which to restore variables.

__enter__

```
__enter__()
```

`__exit__`

```
__exit__(
    exception_type,
    exception_value,
    traceback
)
```

`close`

```
close()
```

`raw_session`

```
raw_session()
```

Returns underlying `TensorFlow.Session` object.

`run`

```
run(
    fetches,
    feed_dict=None,
    options=None,
    run_metadata=None
)
```

Run ops in the monitored session.

This method is completely compatible with the `tf.Session.run()` method.

Args:

- `fetches` : Same as `tf.Session.run()` .
- `feed_dict` : Same as `tf.Session.run()` .
- `options` : Same as `tf.Session.run()` .
- `run_metadata` : Same as `tf.Session.run()` .

Returns:

Same as `tf.Session.run()` .

`should_stop`

```
should_stop()
```

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code samples are licensed under the Apache 2.0 License. For details, see our Site Policies. Java is a registered trademark of Oracle and/or its affiliates.

Last updated November 2, 2017.