;; Auto-generated. Do not edit!

(when (boundp 'gazebo\_msgs\_new::ApplyJointEffort)

(if (not (find-package "GAZEBO\_MSGS\_NEW"))

(make-package "GAZEBO\_MSGS\_NEW"))

(shadow 'ApplyJointEffort (find-package "GAZEBO\_MSGS\_NEW")))

(unless (find-package "GAZEBO\_MSGS\_NEW::APPLYJOINTEFFORT")

(make-package "GAZEBO\_MSGS\_NEW::APPLYJOINTEFFORT"))

(unless (find-package "GAZEBO\_MSGS\_NEW::APPLYJOINTEFFORTREQUEST")

(make-package "GAZEBO\_MSGS\_NEW::APPLYJOINTEFFORTREQUEST"))

(unless (find-package "GAZEBO\_MSGS\_NEW::APPLYJOINTEFFORTRESPONSE")

(make-package "GAZEBO\_MSGS\_NEW::APPLYJOINTEFFORTRESPONSE"))

(in-package "ROS")

(defclass gazebo\_msgs\_new::ApplyJointEffortRequest

:super ros::object

:slots (\_joint\_name \_effort \_start\_time \_duration ))

(defmethod gazebo\_msgs\_new::ApplyJointEffortRequest

(:init

(&key

((:joint\_name \_\_joint\_name) "")

((:effort \_\_effort) 0.0)

((:start\_time \_\_start\_time) (instance ros::time :init))

((:duration \_\_duration) (instance ros::time :init))

)

(send-super :init)

(setq \_joint\_name (string \_\_joint\_name))

(setq \_effort (float \_\_effort))

(setq \_start\_time \_\_start\_time)

(setq \_duration \_\_duration)

self)

(:joint\_name

(&optional \_\_joint\_name)

(if \_\_joint\_name (setq \_joint\_name \_\_joint\_name)) \_joint\_name)

(:effort

(&optional \_\_effort)

(if \_\_effort (setq \_effort \_\_effort)) \_effort)

(:start\_time

(&optional \_\_start\_time)

(if \_\_start\_time (setq \_start\_time \_\_start\_time)) \_start\_time)

(:duration

(&optional \_\_duration)

(if \_\_duration (setq \_duration \_\_duration)) \_duration)

(:serialization-length

()

(+

;; string \_joint\_name

4 (length \_joint\_name)

;; float64 \_effort

8

;; time \_start\_time

8

;; duration \_duration

8

))

(:serialize

(&optional strm)

(let ((s (if strm strm

(make-string-output-stream (send self :serialization-length)))))

;; string \_joint\_name

(write-long (length \_joint\_name) s) (princ \_joint\_name s)

;; float64 \_effort

(sys::poke \_effort (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

;; time \_start\_time

(write-long (send \_start\_time :sec) s) (write-long (send \_start\_time :nsec) s)

;; duration \_duration

(write-long (send \_duration :sec) s) (write-long (send \_duration :nsec) s)

;;

(if (null strm) (get-output-stream-string s))))

(:deserialize

(buf &optional (ptr- 0))

;; string \_joint\_name

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setq \_joint\_name (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

;; float64 \_effort

(setq \_effort (sys::peek buf ptr- :double)) (incf ptr- 8)

;; time \_start\_time

(send \_start\_time :sec (sys::peek buf ptr- :integer)) (incf ptr- 4) (send \_start\_time :nsec (sys::peek buf ptr- :integer)) (incf ptr- 4)

;; duration \_duration

(send \_duration :sec (sys::peek buf ptr- :integer)) (incf ptr- 4) (send \_duration :nsec (sys::peek buf ptr- :integer)) (incf ptr- 4)

;;

self)

)

(defclass gazebo\_msgs\_new::ApplyJointEffortResponse

:super ros::object

:slots (\_success \_status\_message ))

(defmethod gazebo\_msgs\_new::ApplyJointEffortResponse

(:init

(&key

((:success \_\_success) nil)

((:status\_message \_\_status\_message) "")

)

(send-super :init)

(setq \_success \_\_success)

(setq \_status\_message (string \_\_status\_message))

self)

(:success

(&optional \_\_success)

(if \_\_success (setq \_success \_\_success)) \_success)

(:status\_message

(&optional \_\_status\_message)

(if \_\_status\_message (setq \_status\_message \_\_status\_message)) \_status\_message)

(:serialization-length

()

(+

;; bool \_success

1

;; string \_status\_message

4 (length \_status\_message)

))

(:serialize

(&optional strm)

(let ((s (if strm strm

(make-string-output-stream (send self :serialization-length)))))

;; bool \_success

(if \_success (write-byte -1 s) (write-byte 0 s))

;; string \_status\_message

(write-long (length \_status\_message) s) (princ \_status\_message s)

;;

(if (null strm) (get-output-stream-string s))))

(:deserialize

(buf &optional (ptr- 0))

;; bool \_success

(setq \_success (not (= 0 (sys::peek buf ptr- :char)))) (incf ptr- 1)

;; string \_status\_message

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setq \_status\_message (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

;;

self)

)

(defclass gazebo\_msgs\_new::ApplyJointEffort

:super ros::object

:slots ())

(setf (get gazebo\_msgs\_new::ApplyJointEffort :md5sum-) "c0039811b8cc919490b3cff748cdf46b")

(setf (get gazebo\_msgs\_new::ApplyJointEffort :datatype-) "gazebo\_msgs\_new/ApplyJointEffort")

(setf (get gazebo\_msgs\_new::ApplyJointEffort :request) gazebo\_msgs\_new::ApplyJointEffortRequest)

(setf (get gazebo\_msgs\_new::ApplyJointEffort :response) gazebo\_msgs\_new::ApplyJointEffortResponse)

(defmethod gazebo\_msgs\_new::ApplyJointEffortRequest

(:response () (instance gazebo\_msgs\_new::ApplyJointEffortResponse :init)))

(setf (get gazebo\_msgs\_new::ApplyJointEffortRequest :md5sum-) "c0039811b8cc919490b3cff748cdf46b")

(setf (get gazebo\_msgs\_new::ApplyJointEffortRequest :datatype-) "gazebo\_msgs\_new/ApplyJointEffortRequest")

(setf (get gazebo\_msgs\_new::ApplyJointEffortRequest :definition-)

"# set urdf joint effort

string joint\_name # joint to apply wrench (linear force and torque)

float64 effort # effort to apply

time start\_time # optional wrench application start time (seconds)

# if start\_time < current time, start as soon as possible

duration duration # optional duration of wrench application time (seconds)

# if duration < 0, apply wrench continuously without end

# if duration = 0, do nothing

# if duration < step size, assume step size and

# display warning in status\_message

---

bool success # return true if effort application is successful

string status\_message # comments if available

")

(setf (get gazebo\_msgs\_new::ApplyJointEffortResponse :md5sum-) "c0039811b8cc919490b3cff748cdf46b")

(setf (get gazebo\_msgs\_new::ApplyJointEffortResponse :datatype-) "gazebo\_msgs\_new/ApplyJointEffortResponse")

(setf (get gazebo\_msgs\_new::ApplyJointEffortResponse :definition-)

"# set urdf joint effort

string joint\_name # joint to apply wrench (linear force and torque)

float64 effort # effort to apply

time start\_time # optional wrench application start time (seconds)

# if start\_time < current time, start as soon as possible

duration duration # optional duration of wrench application time (seconds)

# if duration < 0, apply wrench continuously without end

# if duration = 0, do nothing

# if duration < step size, assume step size and

# display warning in status\_message

---

bool success # return true if effort application is successful

string status\_message # comments if available

")

(provide :gazebo\_msgs\_new/ApplyJointEffort "c0039811b8cc919490b3cff748cdf46b")