XML

```
<position name="joint1_actuator" joint="joint1" kp="1000" kv="10"/>
```

actuator/position (*)

This element creates a position servo with an optional first-order filter. The underlying **general** attributes are set as follows:

Attribute	Setting	Attribute	Setting
dyntype	none or filterexact	dynprm	timeconst 0 0
gaintype	fixed	gainprm	kp 0 0
biastype	affine	biasprm	0 -kp -kv

This element has one custom attribute in addition to the common attributes:

- name, class, group, ctrllimited, forcelimited, ctrlrange, forcerange, lengthrange, gear, cranklength, joint, jointinparent, tendon, cranksite, slidersite, site, refsite, user

  Same as in actuator/ general.
- kp: real, “1”

  Position feedback gain.
- kv: real, “0”

  Damping applied by the actuator. When using this attribute, it is recommended to use the implicitfast or implicit integrators.

```
<velocity name="left_rear_wheel_joint_actuator" joint="left_rear_wheel_joint"
/>
```

actuator/velocity (*)

This element creates a velocity servo. Note that in order create a PD controller, one has to define two actuators: a position servo and a velocity servo. This is because MuJoCo actuators are SISO while a PD controller takes two control inputs (reference position and reference velocity). When using this actuator, it is recommended to use the implicitfast or implicit integrators. The underlying **general** attributes are set as follows:

Attribute	Setting	Attribute	Setting
dyntype	none	dynprm	1 0 0
gaintype	fixed	gainprm	kv 0 0
biastype	affine	biasprm	0 0 -kv

> This element has one custom attribute in addition to the common attributes:
>
> - name, class, group, ctrllimited, forcelimited, ctrlrange, forcerange, lengthrange, gear, cranklength, joint, jointinparent, tendon, cranksite, slidersite, site, refsite, user
>
>   Same as in actuator/ general.
> - kv: real, “1”
>
>   Velocity feedback gain.

wheel

```
<geom type="mesh" contype="1" conaffinity="1" group="1" density="0" rgba="1 1 1
1" mesh="right_front_wheel_link" friction="2 1 0.5" condim="4"/>
```

> ### body/geom (*)
>
> This element creates a geom, and attaches it rigidly to the body within which the geom is defined. Multiple geoms can be attached to the same body. At runtime they determine the appearance and collision properties of the body. At compile time they can also determine the inertial properties of the body, depending on the presence of the inertial element and the setting of the inertiafromgeom attribute of compiler. This is done by summing the masses and inertias of all geoms attached to the body with geom group in the range specified by the inertiagrouprange attribute of compiler. The geom masses and inertias are computed using the geom shape, a specified density or a geom mass which implies a density, and the assumption of uniform density.
>
> Geoms are not strictly required for physics simulation. One can create and simulate a model that only has bodies and joints. Such a model can even be visualized, using equivalent inertia boxes to represent bodies. Only contact forces would be missing from such a simulation. We do not recommend using such models, but knowing that this is possible helps clarify the role of bodies and geoms in MuJoCo.
>
> contype: int, “1”
>
> This attribute and the next specify 32-bit integer bitmasks used for contact filtering of dynamically generated contact pairs. See Collision detection in the Computation chapter. Two geoms can collide if the contype of one geom is compatible with the conaffinity of the other geom or vice versa. Compatible means that the two bitmasks have a common bit set to 1.

control

```python
# 定义关节控制函数
def control_joints(joint_ids, positions):
    for joint_id, position in zip(joint_ids, positions):
        data.ctrl[joint_id] = position
```

joint_id from actuator id

data

freejoint的qpos一共有七个数，其中，qpos[0~2]是freejoint自身坐标系原点在世界坐标系的位置，qpos[3~6]是自身坐标系相对于世界坐标系的四元数；qvel一共有六个数，其中，qvel[0~2]是freejoint速度在世界坐标系下的表示，而qvel[3~5]是freejoint角速度在自身坐标系下的表示（之前自己一直认为角速度也是在世界坐标系下的表示）

`physics.data.qpos[:]`：**位置**（position）。`qpos` 表示物理引擎中物体的广义位置向量（包括位置和关节角度）。通过给 `qpos` 赋值，可以直接设置物体在环境中的位置或姿态（如坐标、角度等）。

`physics.data.qvel[:]`：**速度**（velocity）。`qvel` 表示物体的广义速度向量，包括线速度和角速度。赋值给 `qvel` 会设置物体的速度状态，影响其随时间的动态行为。

`physics.data.ctrl[:]`：**控制量**（control）。`ctrl` 通常用于设置控制输入，比如机器人关节的驱动力矩或力。这一行代码赋值的是控制信号，影响仿真中机器人的运动。

MuJoCo 学习笔记：概述 Overview - wghou09 - 博客园 (cnblogs.com)