As alternative to re-implementing tags (as suggested by [@CarterG81](https://answers.unity.com/users/190288/carterg81.html)) just use a (possibly empty) component instead of using tags at all - this is far more flexible, and you can search using [FindObjectsOfType](https://docs.unity3d.com/ScriptReference/Object.FindObjectsOfType.html) just the same - without any additional framework/code.

If you need to classify objects in the first place, such as by adding tags, it's likely you'll discover at a later time that those objects have more in common than just a name - and you'll need a behavior for that anyhow; this way, you have a placeholder for that already.

In my space shooter tutorial project, I use a Shootable behavior, which I apply to anything that can be shot - for example:

1. public class Shootable : MonoBehaviour {
2. public GameObject explosion;
4. public void Explode()
5. {
6. Destroy(gameObject);
7. Instantiate(explosion, transform.position, transform.rotation);
8. }
9. }

The controller for my shot is responsible for checking collisions - it checks if objects it collides with are Shootable, and if so, destroys itself, and tells the Shootable to explode:

1. private void OnTriggerEnter(Collider other)
2. {
3. var shootable = other.gameObject.GetComponent<Shootable>();
4. if (shootable) {
5. Destroy(gameObject);
6. shootable.Explode();
7. }
8. }

This way, shootables are responsible for generating their own explosions - you can elaborate on this concept and also make them responsible for counting damage, deciding whether to destroy themselves, and any other effects that may be specific to each kind of shootable object.