Best Practice: Avoid Dependencies between Tests to Run Tests in Parallel

Dependencies between tests prevent tests from being able to run in parallel. And running tests in parallel is by far the best way to speed up the execution of your entire test suite. It's much easier to add a virtual machine than to try to figure out how to squeeze out another second of performance from a single test.

What are dependencies? Imagine a test suite with two tests:

Java Example of Test Dependencies

```
@Test
public void testLogin()
{
    // do some stuff to trigger a login
    assertEquals("My Logged In Page", driver.getTitle());
}

@Test
public void testUserOnlyFunctionality()
{
    driver.findElement(By.id("userOnlyButton")).click();
    assertEquals("Result of clicking userOnlyButton", driver.findElement(By.id("some_result")));
}
```

PHP Example of Test Dependencies

```
<?php
function testLogin()
{
    // do some stuff to trigger a login
    $this->assertEquals("My Logged In Page", $this->title());
}

function testUserOnlyFunctionality()
{
    $this->byId('userOnlyButton')->click();
    $this->assertTextPresent("Result of clicking userOnlyButton");
}
```

In both of these examples, `testLogin()` triggers the browser to log in and asserts that the login was successful. The second test clicks a button on the logged-in page and asserts that a certain result occurred.

This test suite works fine as long as the tests run in order. But second test makes an assumption that you are already logged in, which creates a dependency on the first test. If these tests run at the same time, or if the second one runs before the first one, the browser's cookies will not yet allow Selenium to access the logged-in page, and the second test fails. You can get rid of this dependency by making sure that each test can run independently independently of the others, as shown in these examples.

Java Example of Corrected Test Dependency

```
public void doLogin()
{
        //do some stuff to trigger a login
        assertEquals("My Logged In Page", driver.getTitle());
}

@Test
public void testLogin()
{
        this.doLogin();
}

@Test
public void testUserOnlyFunctionality()
{
        this.doLogin();
        driver.findElement(By.id("userOnlyButton")).click();
        assertEquals("Result of clicking userOnlyButton", driver.findElement(By.id("some_result")));
}
```

PHP Example of Corrected Test Dependency

```
<?php
function doLogin()
{
    // do some stuff to trigger a login
    $this->assertEquals("My Logged In Page", $this->title());
}

function testLogin()
{
    $this->doLogin();
}

function testUserOnlyFunctionality()
{
    $this->doLogin();
    $this->byId('userOnlyButton')->click();
    $this->assertTextPresent("Result of clicking userOnlyButton");
}
```

The main point is that it is dangerous to assume any state when developing tests for your app. Instead, you should find ways to quickly generate desired states for individual tests. In the example, this is accomplished with the `doLogin()` function, which generates a logged-in state instead of assuming it. You might even want to develop an API for the development and test versions of your app that provides URL shortcuts that generate common states. For example, a URL that's only available in test that creates a random user account and logs it in automatically.