# Scenario 02 – Slow Pages

The objectives of this scenario is to highlight the use of SQL Profiler and optional Developer Dashboard and ILSpy. Scenario 02 includes a custom Page Layout with a “navigation” control. This control will call a stored procedure in a custom database – AdventureWorks2012. The sproc is simulated to be slow and thus affecting the page load time. The Page Layout is used in the About and Company pages, but the default page is not affected.

## Origin

The idea for this scenario came from a RAP in Abu Dhabi, where they were they were complaining about poor performance in SharePoint in general. Looking through the RAP findings, the environment looked to be in good shape, and OOB sites ran fine. I attached a SQL Profiler and showed only procedures taking more than 2 sec. – one procedure from a custom database came out, and it turned out to be included in the master page on all sites and provided the navigation logic.

## Prerequisite Lessons

For the students to work with this scenario they should have been through the **ULS** and **SQL Profiler** lessons. Another relevant lesson could be **Developer Dashboard**, but there is also an opportunity to revisit this scenario after the Developer Dashboard lesson to show how it could also have been tracked down.

## Tasks

By leveraging demonstrated troubleshooting tools, the students should complete the following steps as part of this troubleshooting scenario:

1. Find the reason for the pages to render slowly
2. Identify the difference between these pages and e.g. the front page (default.aspx)
3. Find possible ways to resolve the issue
4. Implement a solution without changing the .Net code
5. What options could the developers have considered in order to avoid the performance issue?

## Answers and key discussion points

This section highlights some of the key discussion points of the scenario

### Answers

1. The News Page Layout has a control that calls a Stored Procedure. This sproc has a delay statement of four seconds (to simulate poorly written code logic or missing maintenance).
2. The front page uses a different Page Layout
3. Optimize the sproc or apply a better code pattern (see key discussion points)
4. Remove the delay in the sproc
5. See discussion points

### Use of Custom Databases

Sometimes custom databases are used in custom SharePoint solutions or third party solutions. These can have a significant impact on performance if they are not optimized correctly, however it might not be very visible from a user perspective. The tools mentioned here can help track down these performance issues.

### Controls can be inline in Page Layouts and Master Pages

It is not always that controls are directly visible from the UI. Custom controls can be embedded in the Page Layouts or Master Pages and even added as Delegate Controls. When looking for differences between pages performing and not performing, it is important to have that in mind.

### What can be different

Use the “what can be different” method to determine the difference between the working and non-working pages. E.g. master pages, page layouts, web parts, SharePoint Designer customizations, content size, etc.

Pay attention to the fact that OOB Master Pages and Page Layouts can have been modified directly on disk or through SharePoint Designer.

### Alternative code patterns

If the developers know that there is a chance of intermediate long running operations, then several patterns can be applied. The code can be run async client side – this way the page will load and the result of this control will just be delayed.

Caching the result will also ensure that only few users are affected.

Another option could be to limit the time it can run – e.g. 1 second, and then return a message like “This operation is taking longer than expected, please return back later to try again” to the user.

There is also an opportunity to log this in more details. The logging around this code is actually good, but could be enhanced with timings like demonstrated in the Developer Dashboard lesson.