Tutorial: How to Triage and Diagnose a Problem in an OpenShift Application with DX Application Performance Management

OpenShift Container Platform is a platform for developing and running containerized applications. It is designed to allow applications and the data centers that support them to expand from just a few machines and applications to thousands of machines that serve millions of clients.

In a previous tutorial, we showed how to enroll and onboard an OpenShift application into DX Application Performance Management. Once the said application was deployed, we explained briefly and in simple steps how to set up monitoring and observability counters that capture a holistic view of the application performance.

In this tutorial, we build on top of our previous knowledge. We demonstrate how DX APM can help to identify the potential troublesome cause of the app that has been recently deployed in production. You will see how the automated capabilities of the service can speed up triaging and diagnosing the root of the problem by drilling down to the problematic area of code.

Let's get started...

Interacting with the Application and Finding Some Problems

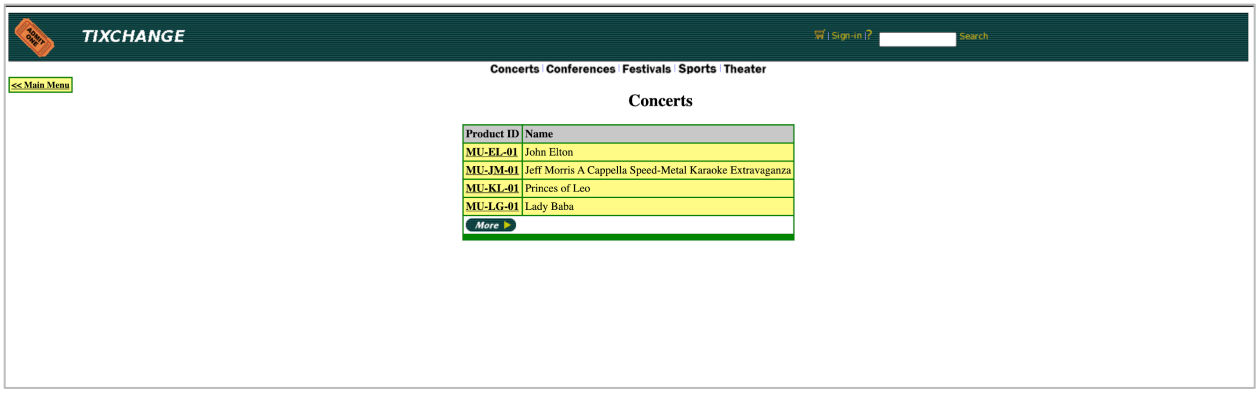
The application we deployed in the previous tutorial is a typical E-Commerce platform for selling all sorts of concert and event tickets. For the purpose of this tutorial, we will focus on troubleshooting this same application, but the triaging steps in DX APM can be applied to any application you are monitoring.

You can see a picture of the initial page view below:

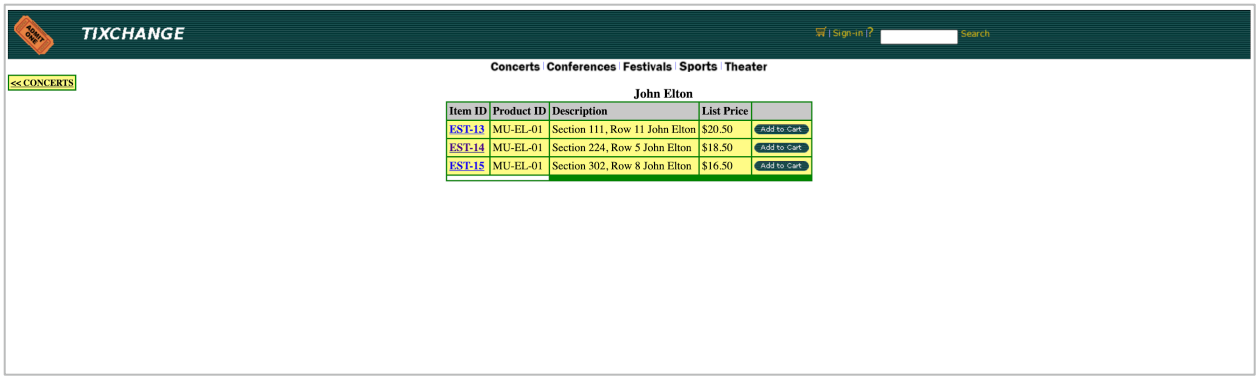
The UI may look like it had some better days, but let's say for the purpose of this exercise that we never had any issues of complaints about that.

From the side panel on the left, we can see the main event categories where we can browse and purchase some tickets. Let's buy a few of them to record our experience.

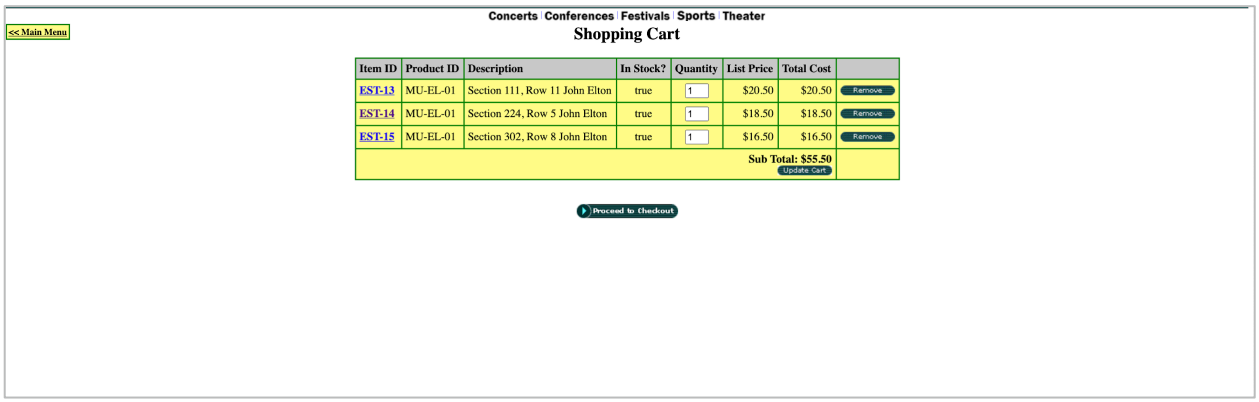
Click on **Concerts, and** then you will see the list of Concert events:

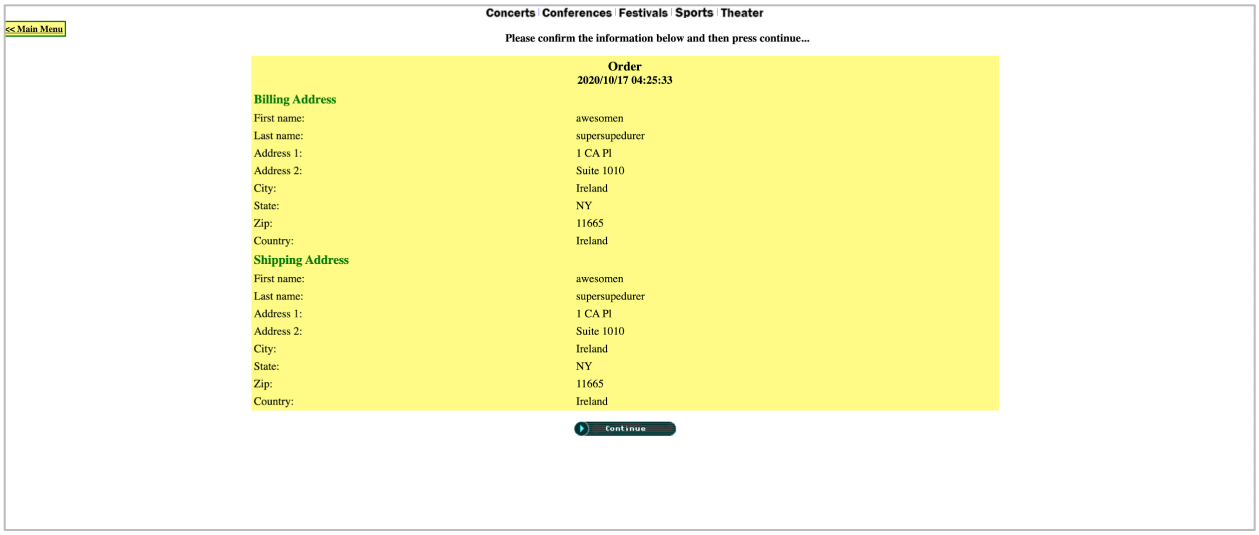


Next, try to add a few tickets to your shopping basket. Click on Elton John and pick one of each ticket seats on the next screen:

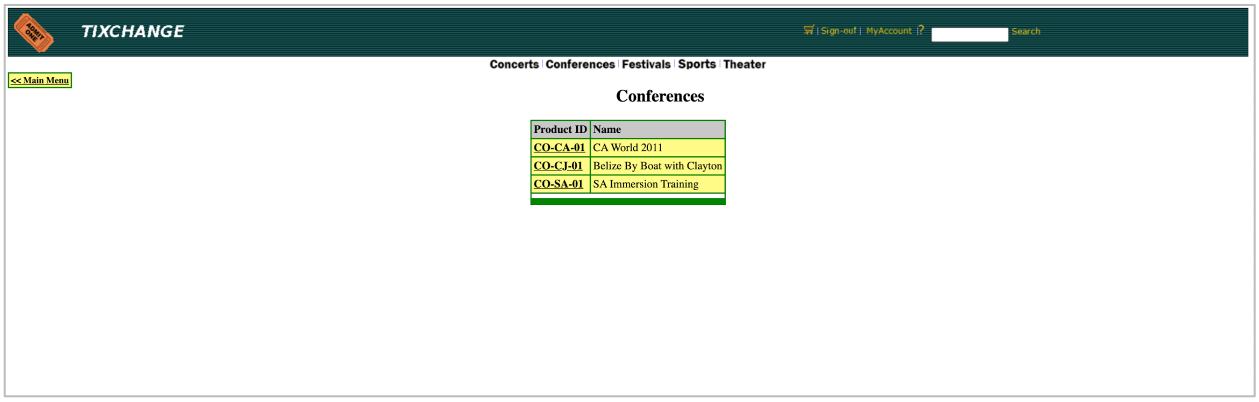


Proceed to checkout and complete the purchase:

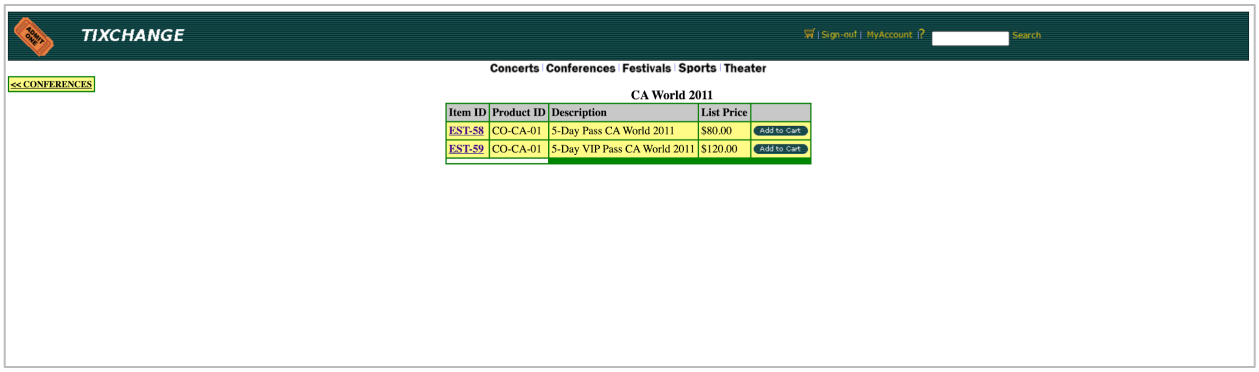




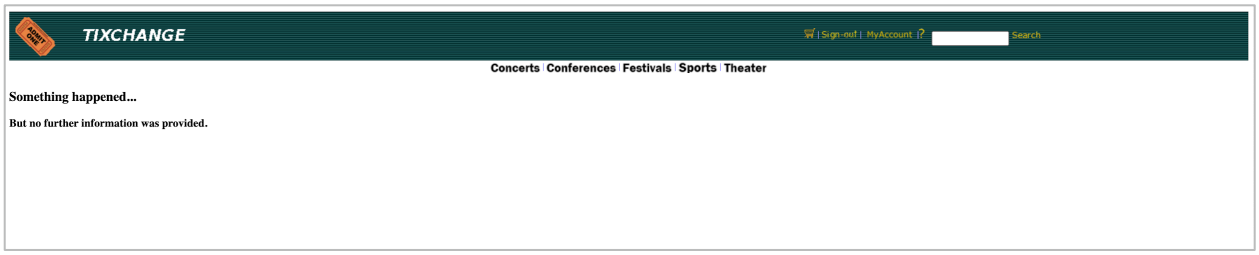
So far, so good. Now let's say you enjoyed the experience, and you left positive feedback about the shopping experience. On a later date, you want to attend a conference. This time you log in again into the same platform, only now you click on **Conferences:**



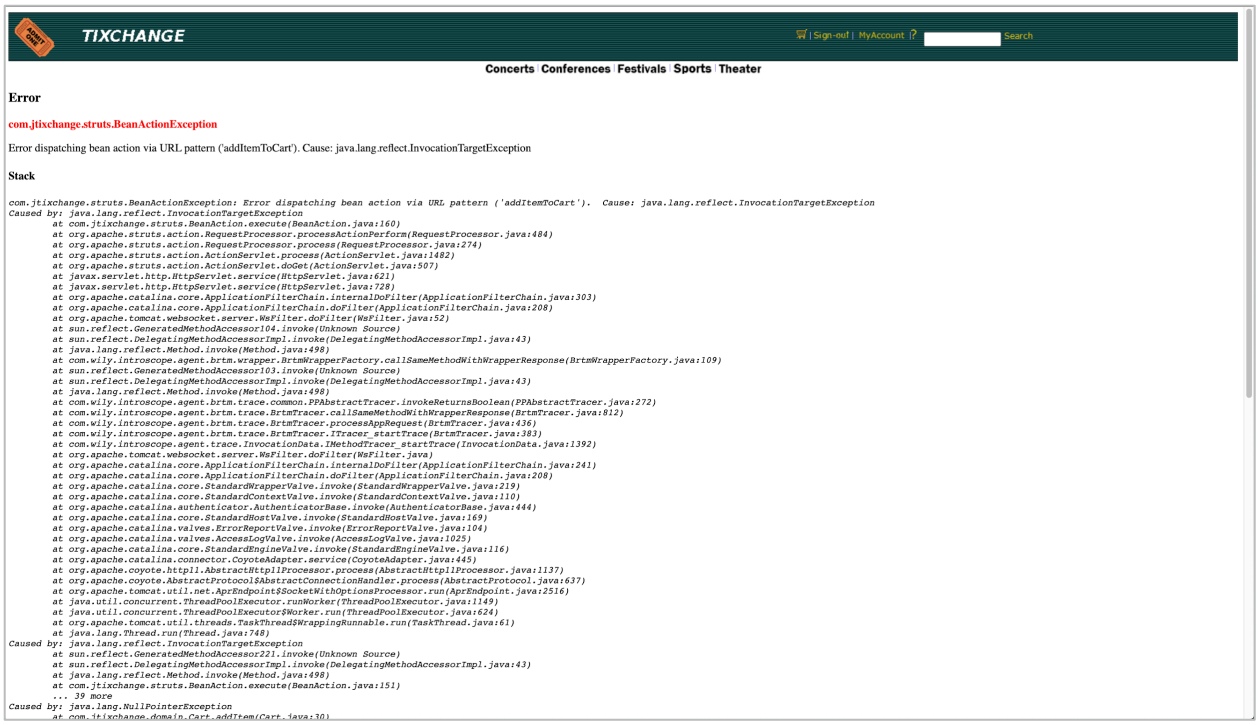
You want to attend the CA World 2011 and as before, you go ahead to purchase some day passes, so you click on the event link:



You click on the Item ID cell EST-58, as you want to see more information about the 5-day pass ticket. Sadly, you are presented with an error screen informing you that something wrong has happened.



Then you think you may as well buy the tickets anyway as you've made up your mind, so you go back and try to add the ticket into the shopping cart. Then this happens:



As a plain user, you understand that something wrong is going on and you kindly inform the site support staff that there is an error on that page. Sadly for the platform, you proceed to find another website to purchase the season pass and complete the purchase there.

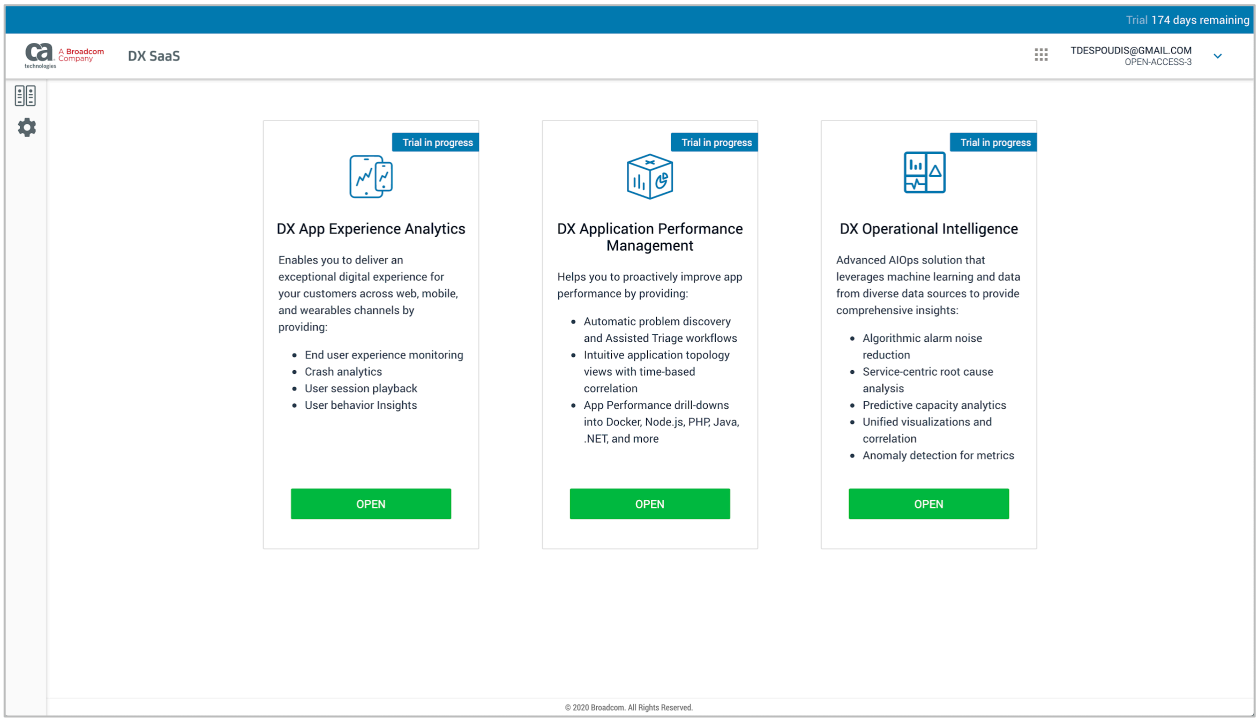
As you have guessed, this is a typical reaction for that situation, and for things like that to propagate into production is not only embarrassing but costly. Is there a way to prevent that from happening in the first place? Or is there a way to at least be more proactive and diagnose the problem before any complaints?

Investigating the issue with DX APM

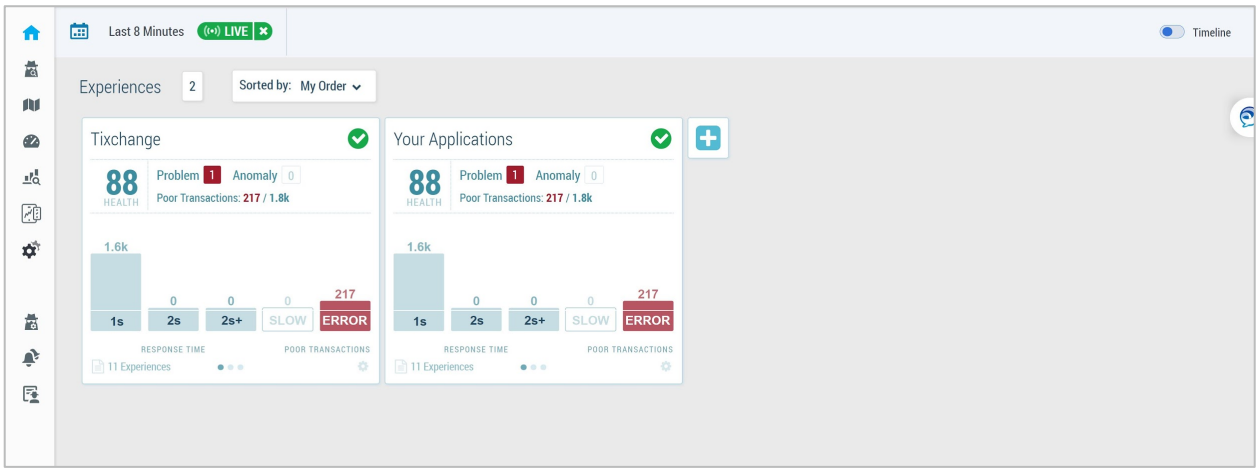
In this section, we're going to see how to proactively diagnose problems in production with DX APM.

In the previous section, we saw how a problem that propagated in production can result in a negative experience and loss of revenue. Next, we are going to switch to the role of an SRE that monitors the application behavior for issues, and can go into the code itself and make changes when needed.

First, we log in to DX APM:

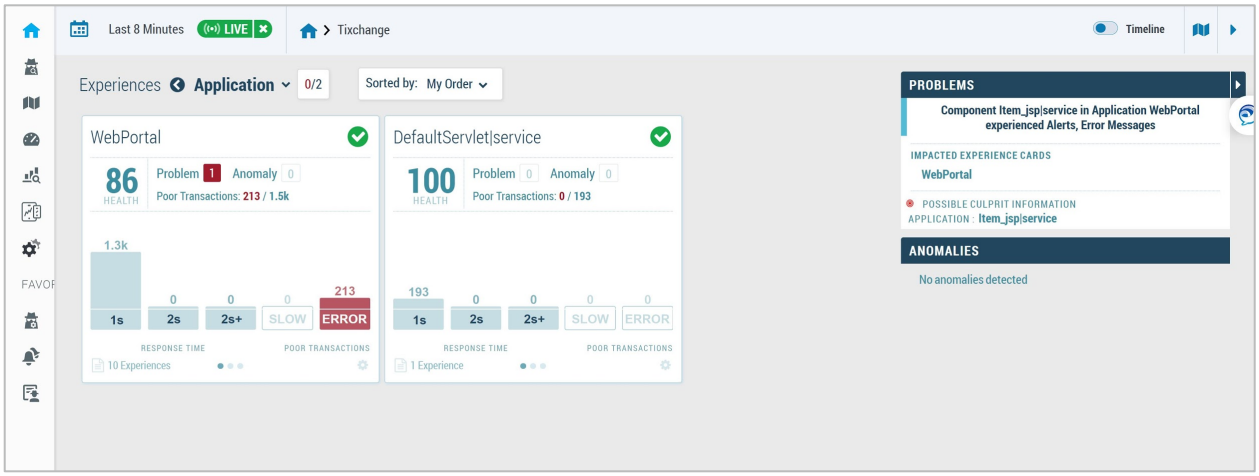


Upon logging in to DX APM, you will be presented with the **Experience View** screen which shows a list of enrolled applications that the platform monitors.



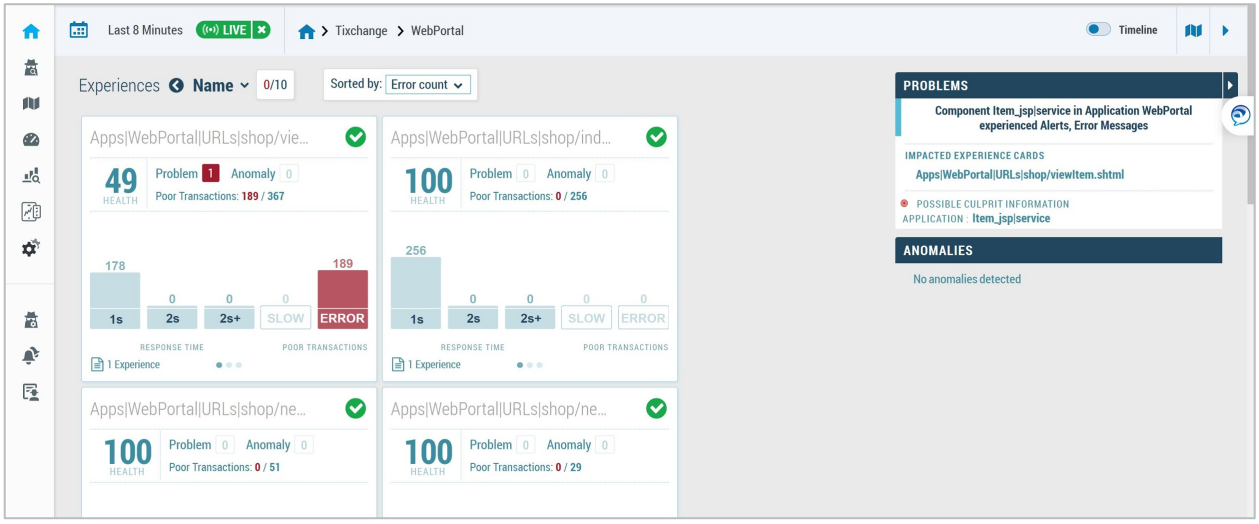
This view shows a summary view of the end-user transaction health for the monitored application on the cluster. By default, the monitoring windows are for the last 24 hours, but you can configure that in history or with live updates.

We can see that the Tixchange application shows some errors. From this view, we can investigate those errors by clicking on the Card and checking the next screen.



Here we can see the WebPortal application that hosts the Tixchange project. On the right-hand side you can see a panel with the overview of the Problems and Anomalies detected here. This is useful as it gives you insights about where the errors originate and which components are affected.

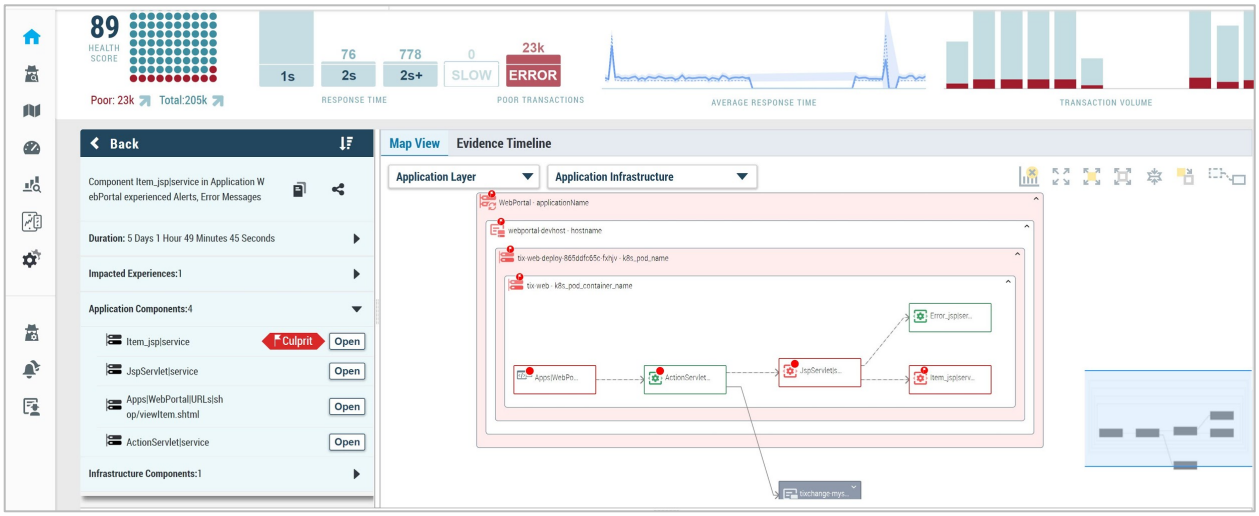
Click on the WebPortal card to see the detailed view:



This screen shows the related paths and endpoints that serve particular business services. We can see the individual health and performance bar charts, and we have the option to sort them by parameters. By looking at the dashboard above we can quickly conclude that the **viewItem** endpoint is experiencing major problems.

A problem is something that can be categorized as a high-severity effect that has an impact on the end-user. An anomaly is an effect that deviates from the normal behavior and requires some attention.

To proceed forward, we need to click on the Problem link that shows the detailed report of the errors:



This is the Analysis Notebook view, which is very helpful as it depicts several components including:

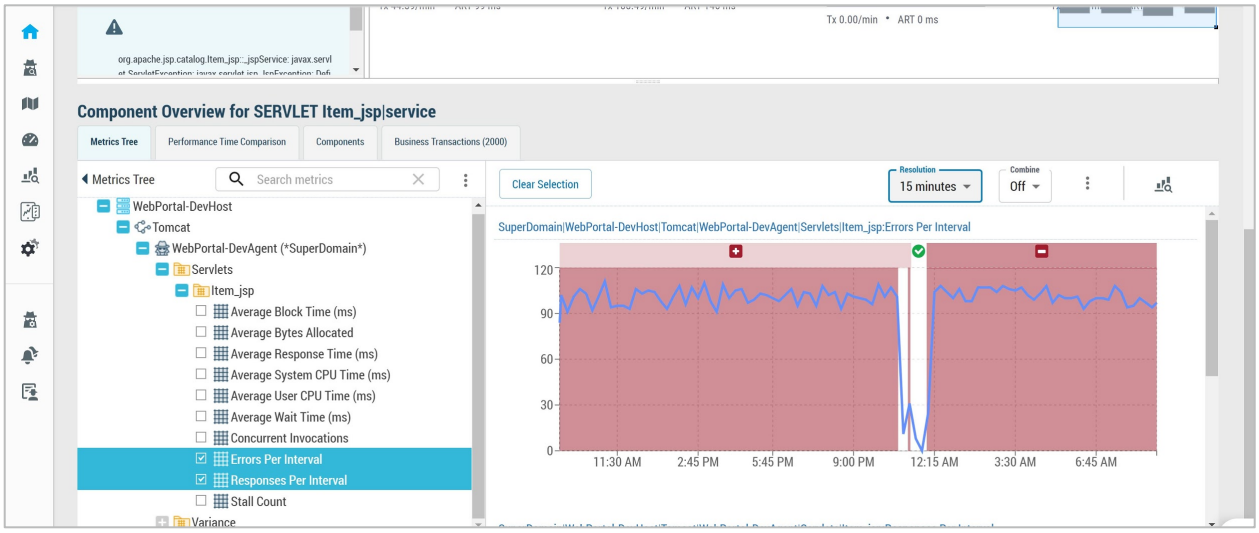
- The Network Map view of the Microservice components,
- The Evidence Timeline that shows the type of errors throughout the time period selected,
- A panel with flags pointing to the detected culprit that causes the errors.
- Showing Application, Infrastructure and network correlation with the root cause either application, network or infrastructure component.

The panel on the left-hand side already informs us that the culprit is one Application Component. This is very useful as it narrows down considerably the investigation surface.

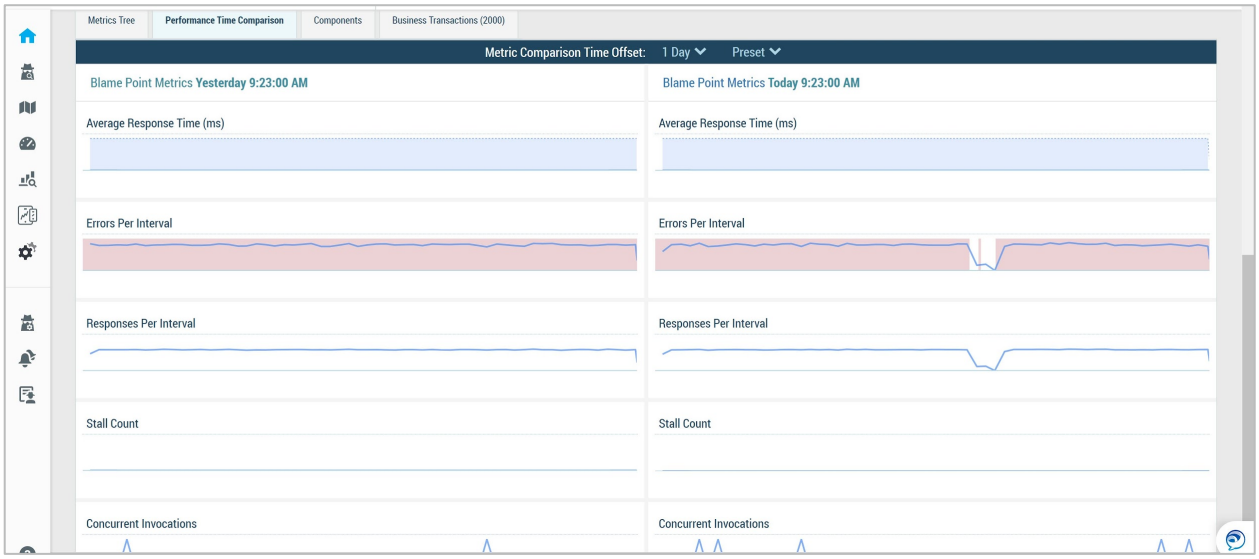
Click on the Culprit flag and then scroll down into the panel with the title:

Component Overview for SERVLET Item_jsp|service

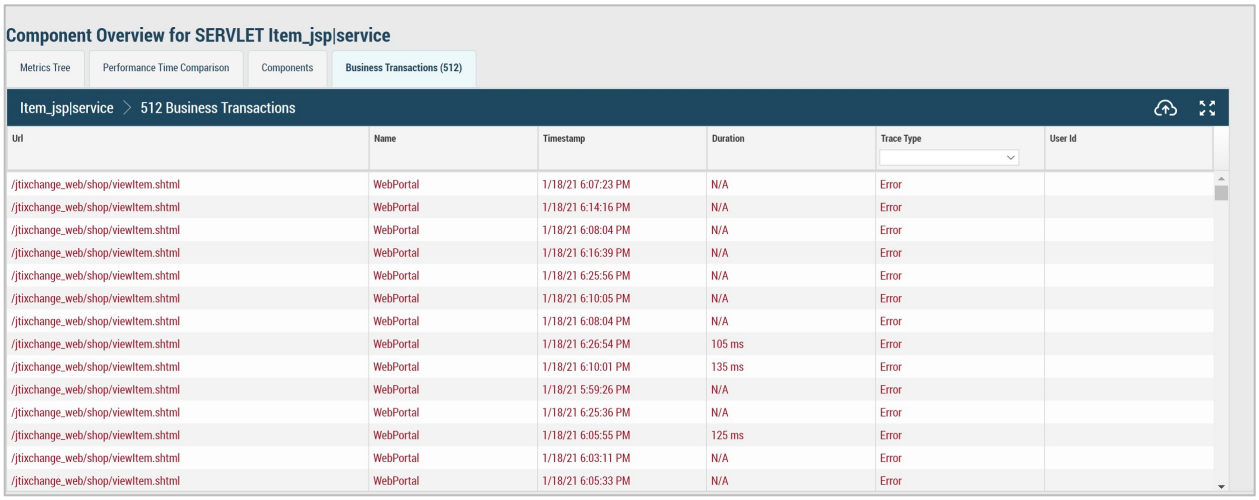
There you can see the metrics view of the Items_jsp servlet and you can check each one of the boxes to see a graph of the metrics. You are mostly interested to see the combined Errors Per Interval and the Responses by interval. You can select both boxes and use the dropdown **Combine** option on the right to merge them all in one graph:



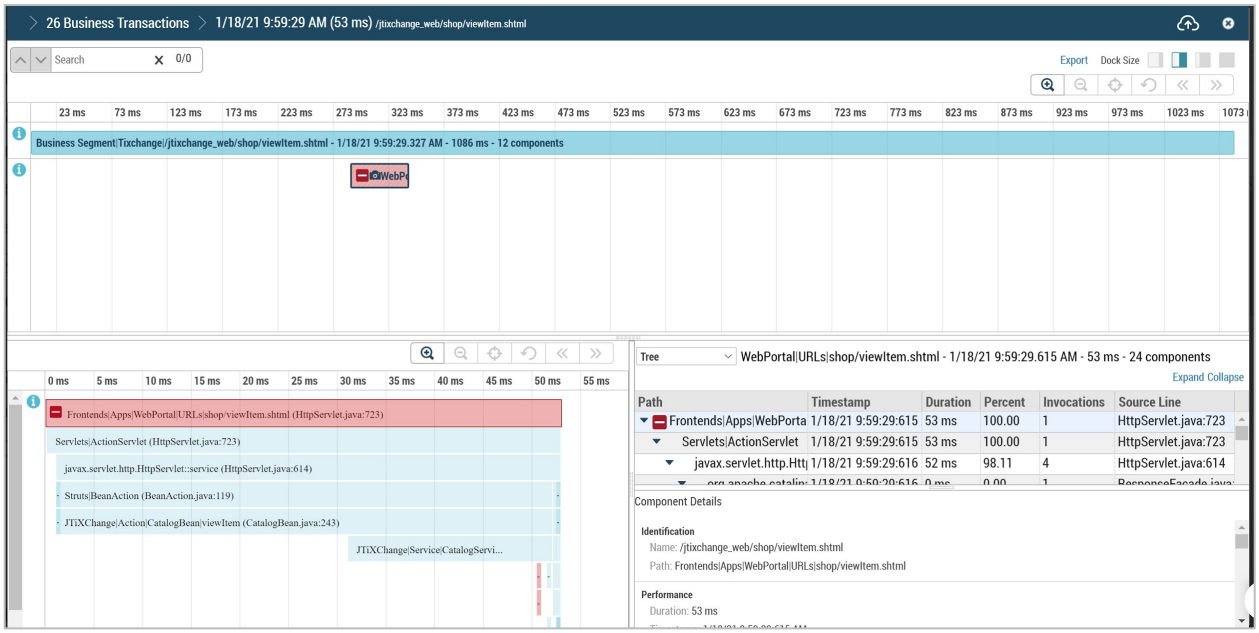
You can see that all responses resulted in errors, and that's really bad. If you click on the **Performance Time comparison** tab, you can inspect the whole metrics in one view; and when you hover on top of one graph, you can see the parallel values on each dimension:



Clicking on the last tab – Business Transactions – can reveal more information about the failing requests. Actually, this tab exposes a more drilled-down view of the request-response cycle. DX APM automatically collects deep dive diagnostics of application transactions.

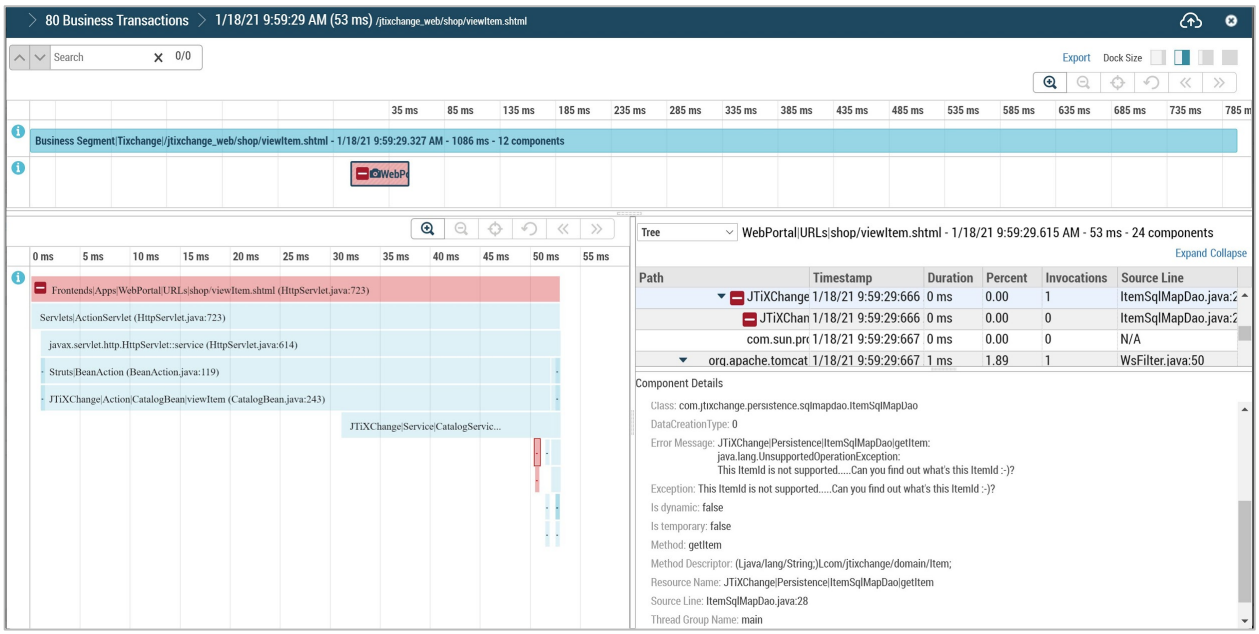


As you can see there is a list of erroneous transactions. By clicking in one of them reveals the source of the problem. (It may take a few seconds to load the details, though).



You will actually see the correlation of the request-response timeline and the reported stack trace. There are some lightning icons (, ⚡) that are marked as potential culprits. By clicking on each of the lightning boxes and inspecting the stack trace on the right-hand side we can easily conclude that the **ItemSqlMapDao** object tries to find a product item in the database and somehow fails to find it, so it throws an error might be due to the item being over.

Note that you may see errors propagated and impacted end users experience **viewItems.html** shown with lightning icons ( ) .In addition, by inspecting the Component Details section on the bottom right corner, we can infer that **ItemSqlMapDao** class **getItem** method at line 28 causes the error.



With this information, we can now pass it to the developers; or, as an SRE with an error budget, you can fix it yourself with some proper documentation.

Next Steps

Learn more about DX APM at www.broadcom.com/apm or check out our documentation page for more in-depth information on how to configure and operate the solution.