Best practices for development environment and API dev?

Asked 16 years, 3 months ago Modified 8 years, 6 months ago

Viewed 892 times

-1

My current employer uses a 3rd party hosted CRM provider and we have a fairly sophisticated integration tier between the two systems. Amongst the capabilities of the CRM provider is for developers to author business logic in a Java like language and on events such as the user clicking a button or submitting a new account into the system, have validation and/or business logic fire off.

One of the capabilities that we make use of is for that business code running on the hosted provider to invoke web services that we host. The canonical example is a sales rep entering in a new sales lead and hitting a button to ping our systems to see if we can identify that new lead based on email address, company/first/last name, etc, and if so, return back an internal GUID that represents that individual. This all works for us fine, but we've run into a wall again and again in trying to setup a sane dev environment to work against.

So while our use case is a bit nuanced, this can generally apply to any development house that builds APIs for 3rd party consumption: **what are some best practices when designing a development pipeline and environment**

when you're building APIs to be consumed by the outside world?

At our office, all our devs are behind a firewall, so code in progress can't be hit by the outside world, in our case the CRM provider. We could poke holes in the firewall but that's less than ideal from a security surface area standpoint. Especially if the # of devs who need to be in a DMZ like area is high. We currently are trying a single dev machine in the DMZ and then remoting into it as needed to do dev work, but that's created a resource scarcity issue if multiple devs need the box, let alone they're making potentially conflicting changes (e.g. different branches).

We've considered just mocking/faking incoming requests by building fake clients for these services, but that's a pretty major overhead in building out feature sets (though it does by nature reinforce a testability of our APIs). This also doesn't obviate the fact that sometimes we really do need to diagnose/debug issues coming from the real client itself, not some faked request payload.

What have others done in these types of scenarios? In this day and age of mashups, there have to be a lot of folks out there w/ experiences of developing APIs--what's worked (and not worked so) well for the folks out there?

development-environment **pipeline** **api-design**

Share

Improve this question

Follow

edited May 31, 2016 at 16:41

Laurel

6,163 ● 14 ● 33 ● 59

asked Aug 27, 2008 at 1:38

bakasan

2,302 ● 2 ● 26 ● 33

2 Answers

Sorted by: Highest score (default)

1

In the occasions when this has been relevant to me (which, truth be told, is not often) we have tended to do a combination of hosting a dev copy of the solution in-house and mocking what we can't host.

I personally think that the more you can host on individual dev boxes the better-- if your dev's PCs are powerful enough to have the entire thing running plus whatever else they need to develop then they should be doing this. It allows them to have tonnes of flexability to develop without worrying about other people.

Share Improve this answer

Follow

answered Aug 27, 2008 at 3:55

SCdF

59.3k ● 24 ● 79 ● 114

1

For dev, it would make sense to use mock objects and write good unit tests that define the task at hand. It would help to ensure that the developers understand the



business requirements. The mock libraries are very sophisticated and help solve this problem.

Then perhaps a continuous build process that moves the code to the dev box in the DMZ. A robust QA process would make sense plus general UAT testing.

Also, for general debugging, you again need to have access the machine in the DMZ where you remote in.

This is probably an "ideal" situation, but you did ask for best practices :).

Share Improve this answer

Follow

answered Sep 30, 2008 at 16:03

Pat

299 ● 2 ● 7