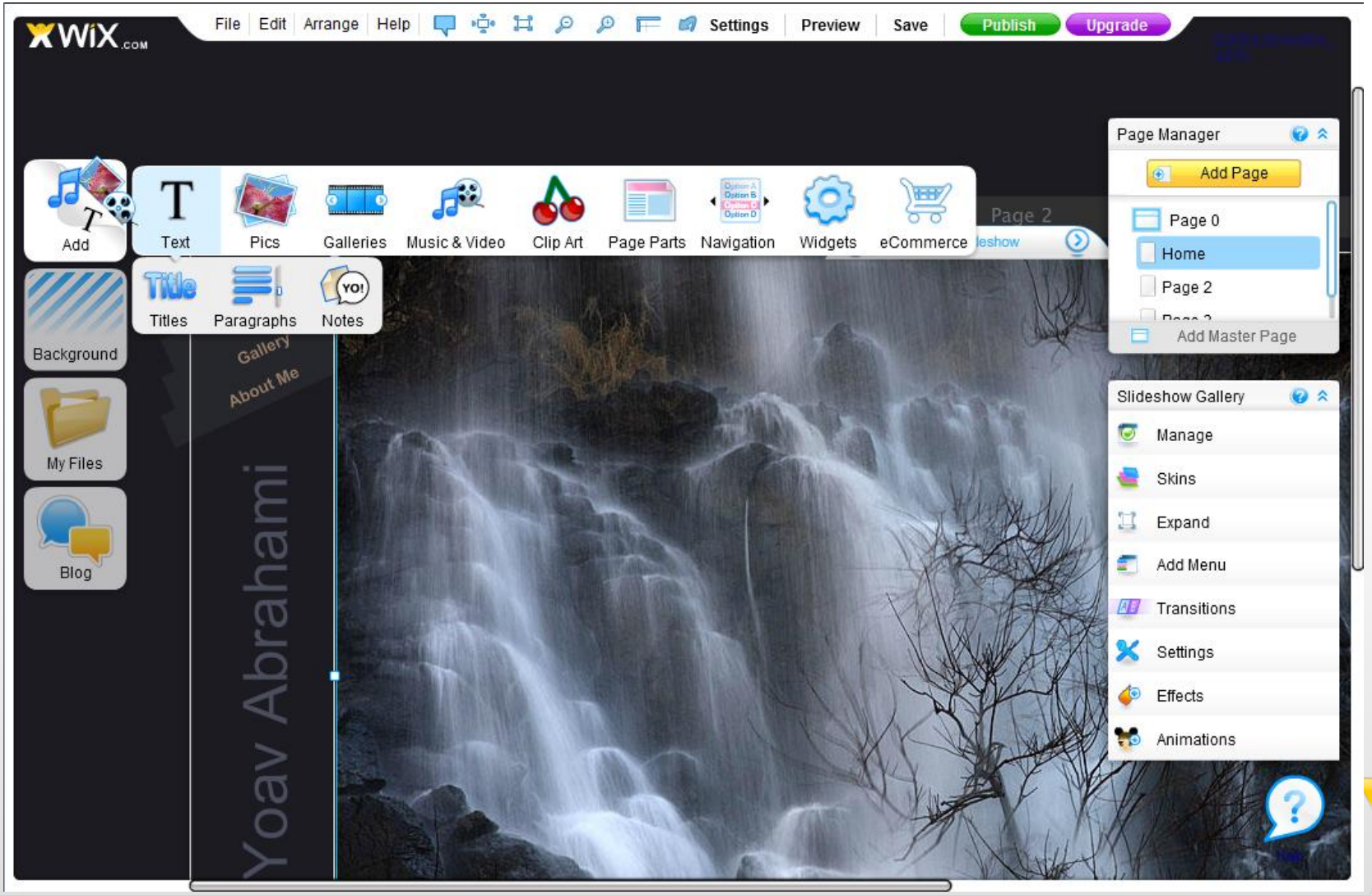
copenhagen goto; conference

Continuous Delivery at Wix

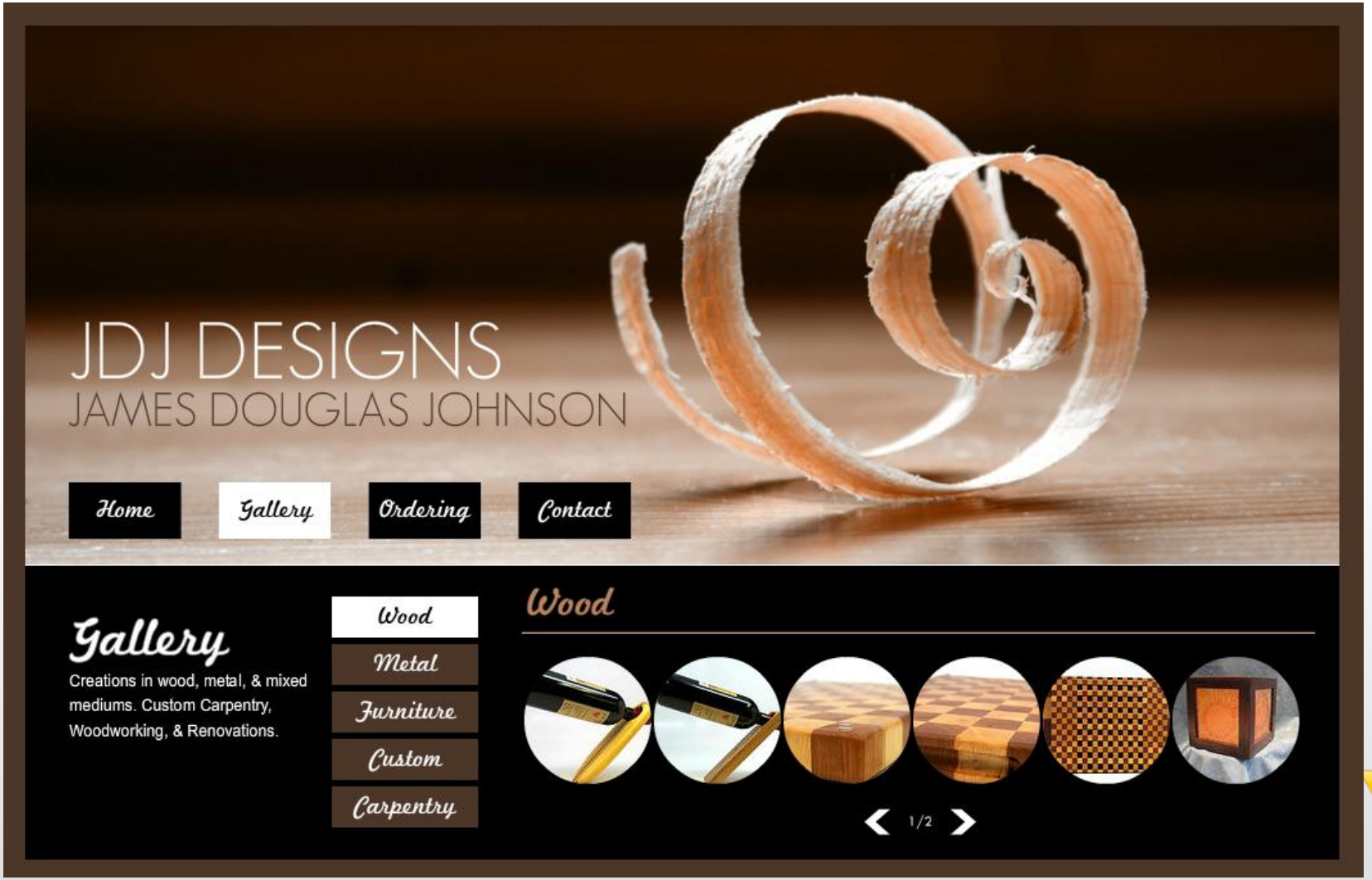
The motivations for CI/CD/TDD, implementation and impact

Yoav Abrahami

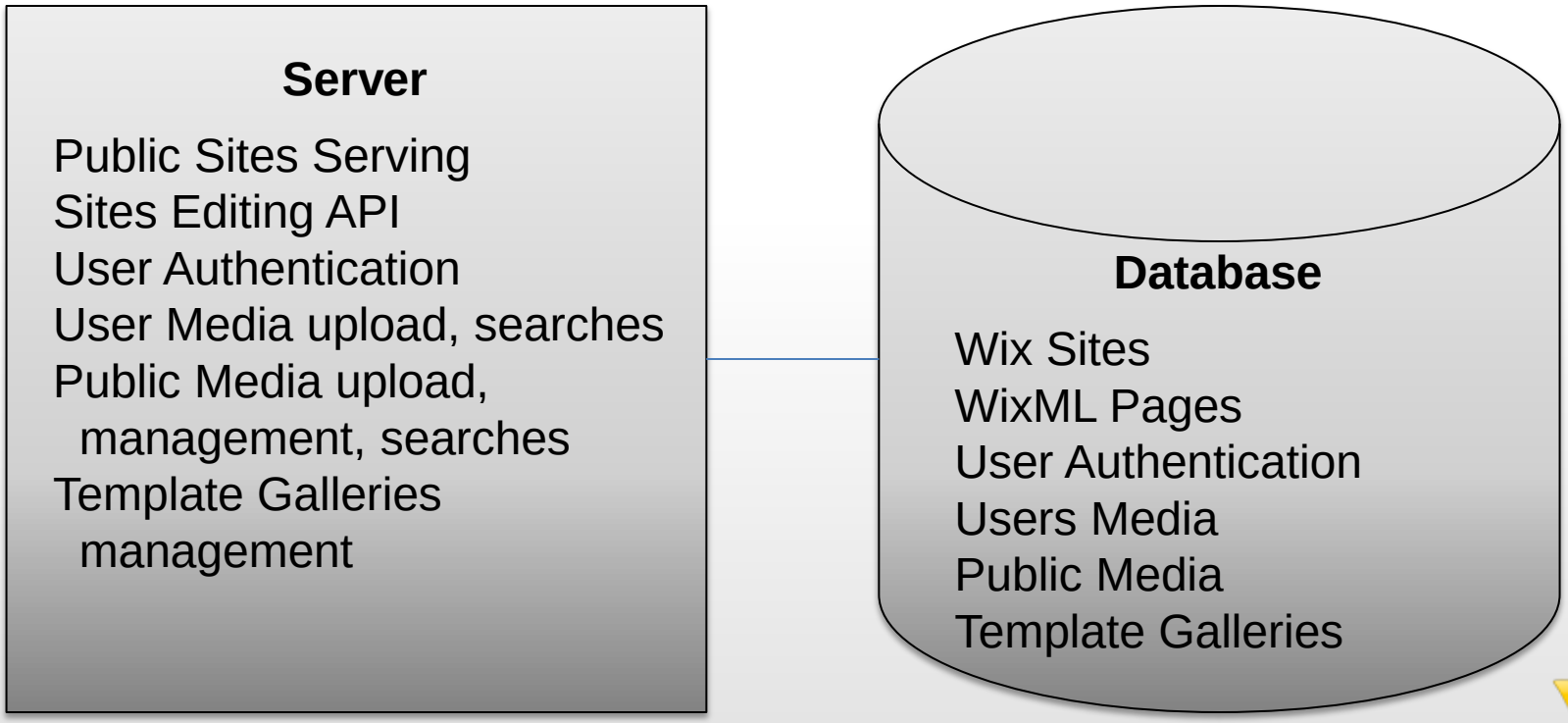
About Wix


File
Edit
Arrange
Help
Settings
Preview
Save
Publish
Upgrade
Add
Text
Pics
Galleries
Music & Video
Clip Art
Page Parts
Navigation
Widgets
eCommerce
Titles
Paragraphs
Notes
Background
My Files
Blog
Page Manager
Add Page
Page 0
Home
Page 2
Add Master Page
Slideshow Gallery
Manage
Skins
Expand
Add Menu
Transitions
Settings
Effects
Animations
Gallery
About Me
Yoav Abrahami

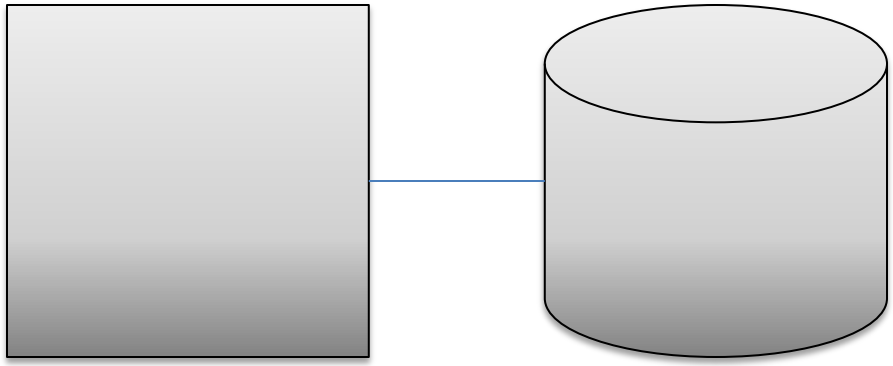
About Wix


JDJ DESIGNS
JAMES DOUGLAS JOHNSON
Home
Gallery
Ordering
Contact
Gallery
Creations in wood, metal, & mixed mediums. Custom Carpentry, Woodworking, & Renovations.
Wood
Metal
Furniture
Custom
Carpentry
Wood
1/2

Wix Initial Architecture

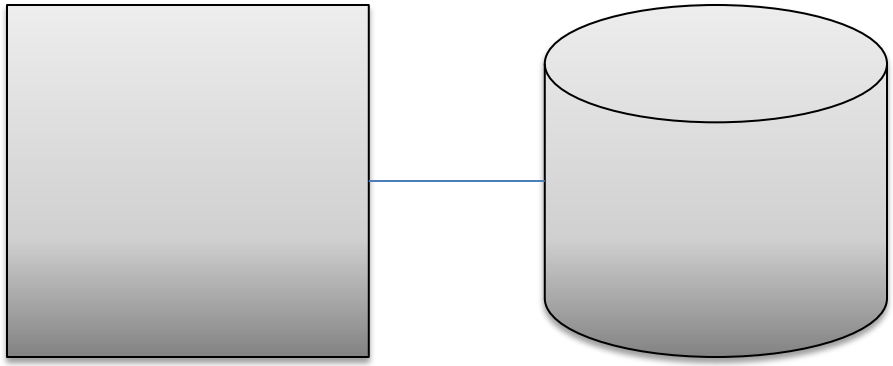


Wix Initial Architecture



- Tomcat, Hibernate, Custom web framework
  - Everything generated from HBM files
  - Built for fast development
  - Statefull login (tomcat session), EHCache, File uploads
  - Not considering performance, scalability, fast feature rollout, testing
  - It reflected the fact that we didn't really know what is our business
  - Yoav A. - "it is great for a first stage start-up, but you will have to replace it within 2 years"
  - Nadav A, after two years - "you were right, however, you failed to mention how hard it's gonna be"

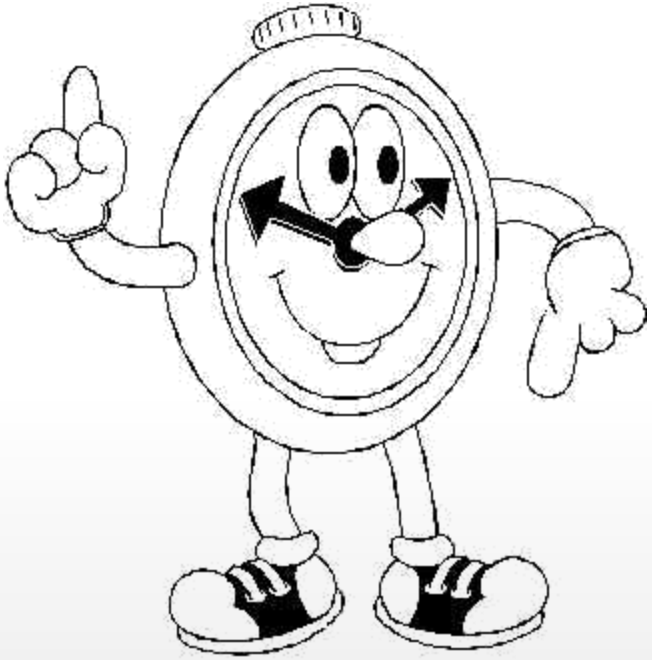
Wix Initial Architecture



What we have learned

- Don’t worry about ‘building it right from the start’ – you won’t
- You are going to replace stuff you are building in the initial stages of a startup or any project
- Be ready to do it
- Get it up to customers as fast as you can. Get feedback. Evolve.
- Our mistake was not planning for gradual re-write
- Build for gradual re-write as you learn the problems and find the right solutions

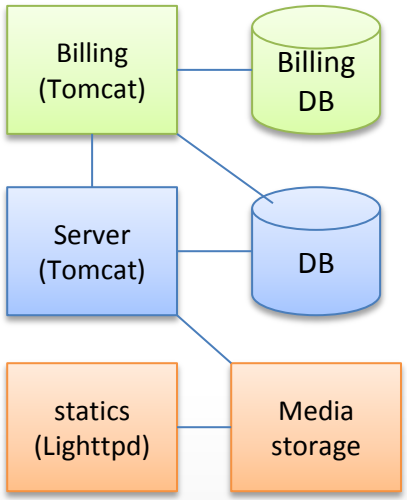
Two years passed



- We learned what our business is – building websites
- We started selling premium websites

Two years passed

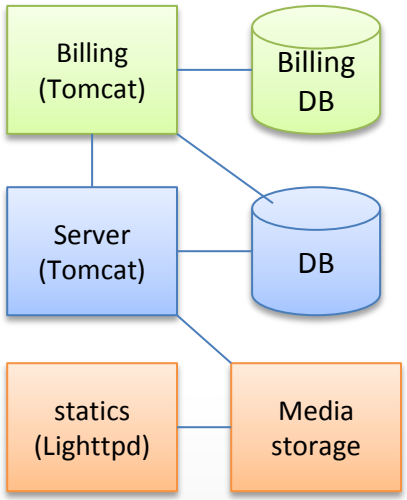
- Our architecture evolved
  - We added a separate Billing segment
  - We moved static file storage and HTTP serving to a separate instance



- But we started seeing problems
  - Updates to our server imposed complete wix downtime
  - Our static storage reached 500 GByte of small files, the limit of Bash scripts
  - The codebase became large and entangled. Feature rollout became harder over time, requiring longer and longer manual regression
    - Strange full-table scans queries generated by Hibernate, which we still have no idea what code is responsible for…
  - Statefull user sessions required a lot of memory and a statefull load balancer

Two years passed

- Our architecture evolved
  - We added a separate Billing segment
  - We moved static file storage and HTTP serving to a separate instance



- But we started seeing problems
  - Updates to our server imposed complete wix downtime
  - Our static storage reached 500 GByte of small files, the limit of Bash scripts
  - **The codebase became large and entangled. Feature rollout became harder over time, requiring longer and longer manual regression**
    - **Strange full-table scans queries generated by Hibernate, which we still have no idea what code is responsible for…**
  - Statefull user sessions required a lot of memory and a statefull load balancer

Motivations for CI/CD/TDD

- We were working traditional waterfall
- With fear of change
  - It is working, why touch it?
  - Uploading a release means downtime and bugs!
- With low product quality
  - Want to risk fixing this bug? Who knows what may break?
- With slow development velocity
  - From “I have a great new product idea” to “it is working” takes too match time
- With tradition enterprise development lifecycle
  - Three months of a “VERSION” development and QA
  - Six months of crisis mode cleaning bugs and stabilizing system

Wix's CI/CD/TDD model

- Abandon "VERSION" paradigm – move feature centric life
- Make small and frequent release as soon as possible
  - Today we release about 10 times a day, gaining velocity
- Empower the developer
  - The developer is responsible from product idea to 10,000 active users
  - Remove every obstacle in the developer's path
  - Big cultural change from waterfall – affects the whole company
- Automate everything – CI/CD/TDD
  - CI – Continuous Integration
  - CD – Continuous Delivery / Deployment
  - TDD – Test Driven Development
- Measure Everything
  - A/B test every new feature
  - Monitor real KPIs (business, not CPU)

Test Driven Development

- TDD workflow
  - Definition: ~~First write a test-case, then write the code for the test to pass and then refactor the code~~
  - My Definition: write the code and tests at the same time. During development, run only tests! (don't write Main(), deploy to app server, etc).
- Code vs Testing Code
  - Developers invest in refactoring the production code to have high quality.
  - ~~But the test code is just that something we ^$@&*@# ~*@ must live with.~~
  - Test code is as important as production code. We invest in modeling it, refactoring it and building the tools to make it clear and maintainable.

Test Driven Development

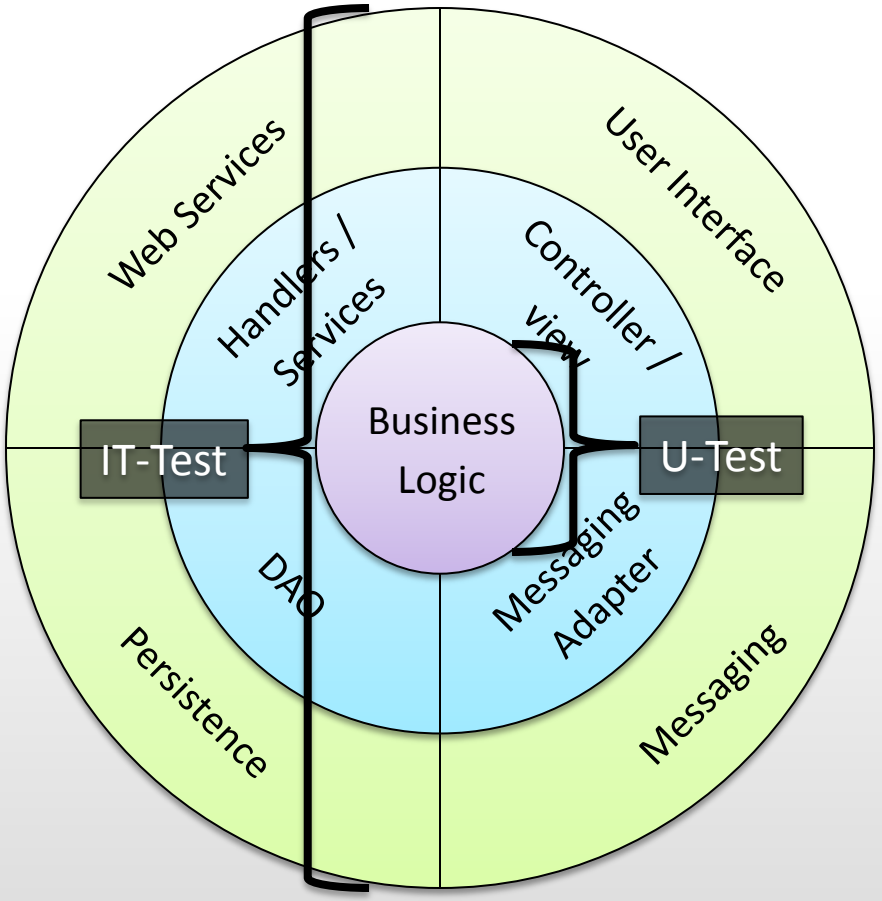
- What people think is the impact on development
  - ~~TDD slows down development~~
  - With TDD we write more code (product + test code).
- Actual impact on development
  - We development faster
  - Removes fear of change
  - Easier to enter some-else's project
  - Do we really need QA? (Yes, they code tests)
  - 10-30% slower, 45-90% less bugs
  - Considerably faster time to fix bugs
- Current Test Count (U-Tests + IT-Tests) – over **6500**

TDD @ Wix

- Server side – Java, C - Automated U-Tests and IT-Tests
  - U-Tests – mockito, Hamcrest, JUnit, Wix enhancements (logging, builders, etc).
  - IT-Tests – full embedded mode support, including embedded MySQl, embedded Jetty, embedded MongoDB, etc.
  - All tests run on every code check-in
- Client side – JS - Automated U-Tests and working on Automating GUI-Tests
  - U-Tests – Jasmine, Testacle – distributed parallel U-Test runner integrated into IDE and Maven
  - GUI Tests
    - Working on Selenium, with embedded RC and external grid
    - Still a large manual effort
  - U-Tests run on every code check-in
  - Lint (custom profile) run on every code check-in

TDD @ Wix

- U-Tests
  - Test the business logic of the application
  - No Dependencies
- IT-Tests
  - Test the integration with different libraries (inbound or outbound)
  - Tests if we use the library correctly
- Learning Test
  - Tests used to learn how to use a certain library



TDD @ Wix

- U-Test example (as complex as it gets)
  - Setup: Custom Junit Runner and mocking
  - White Box test

```
@Test
public void testRenderingNoDebug() {
    when(scriptSource.getScriptsList(DebugMode.nodebug))
        .thenReturn(ImmutableMultimap.<String, Url>builder()
            .putAll("core", new Url(CORE1_JS), new Url(CORE2_JS))
            .putAll("main", new Url(MAIN1_JS), new Url(MAIN2_JS))
            .build());
    when(scriptSource.getScriptsList(DebugMode.nodebug))
        .thenReturn(ImmutableMultimap.<String, Url>builder()
            .putAll("core", new Url(CORE3_JS))
            .putAll("main", new Url(MAIN3_JS))
            .build());

    Renderable renderable = scriptsRenderer.renderScripts(DebugMode.nodebug);
    assertThat(renderable.toString(), allOf(
                not(containsString("<script type=\"text/javascript\" src=\"" +
                    CORE1_JS + "\"></script>")),
                not(containsString("<script type=\"text/javascript\" src=\"" +
                    CORE2_JS + "\"></script>")),
                not(containsString("<script type=\"text/javascript\" src=\"" +
                    MAIN1_JS + "\"></script>")),
                not(containsString("<script type=\"text/javascript\" src=\"" +
                    MAIN2_JS + "\"></script>")),
                containsString("<script type=\"text/javascript\" src=\"" +
                    MAIN3_JS + "\"></script>"),
                containsString("<script type=\"text/javascript\" src=\"" +
                    MAIN3_JS + "\"></script>"),
                not(containsString("${"))));
}
```

TDD @ Wix

- IT-Test example (as complex as it gets)
  - Setup: embedded MySQL, migrations, embedded Jetty, testDao
  - Black Box test - Test over HTTP (Json RPC in this case) to DB.

```
@Test
public void renderWebHtmlUsingRpcPositive() throws IOException {
    Document document = buildSampleDocument();
    testWebSiteDao.saveOrUpdate(defaultSite_1()
                .withDocumentJson(siteDigester.serializeDocument(document))
                .withWixDataJson("{}")
                .build());

    Route route1 = defaultRoute("www", "/")
                .withIdInApp(siteId_1.getId())
                .withApplicationType(ApplicationType.Flash)
                .build();
    Route route2 = defaultRoute("m", "/")
                .withIdInApp(siteId_2.getId())
                .withApplicationType(ApplicationType.HtmlMobile)
                .build();

    RenderResponse render = remoteWebHtmlRemoteRenderer.render(defaultRequest()
                .withMetaSite(defaultMetaSite(metaSiteId, route1, route2))
                .withRoute(route1)
                .withPath("/")
                .build());

    assertThat(render.getHeadContent().render(), containsString(FAVICON_JPG));
        assertThat(render.getBodyContent().render(), allOf(containsString(PAGE_DATA_ID_1),
                containsString(PAGE_1),
                containsString(PAGE_2)));
}
```

Guidelines for successful TDD

- Tests should run on project checkout to a random computer.
  - No dependencies on anything installed
- Tests that cannot be debugged on a developer machine will never consistently run for any period of time
- Tests should run fast
- Tests have to be readable
  - They are the project spec
- Fixture is evil!

CI/CD @ Wix – Release Process

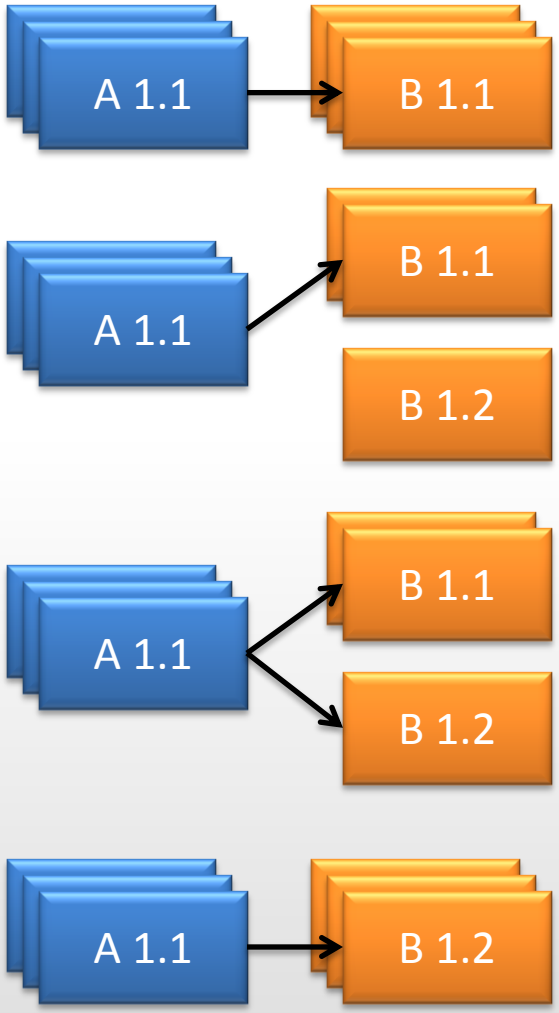
- During development (on developers machine)
  - Maven (Snapshot), one Trunk
  - Compile → Unit Tests → Embedded ITs
- On code check-in
  - TeamCity, Maven (Snapshot), Artifactory
  - Compile → Unit Tests → Embedded ITs → Dev Repo
- Mark as RC
  - Lifecycle, TeamCity, Maven (release), Artifactory
  - Compile → Unit Tests → Embedded ITs → RC Repo
- Staging (when needed)
  - Chix, Chef, Sous-Chef, Artifactory, New Relic, App-Info
  - Deploy - Staging → Self Tests → Monitor → QA Tests
- Deploy to production
  - Chef, Artifactory, Sous-Chef, New Relic, App-Info
  - Deploy - Production → Self Tests → Monitor → A/B Test

How does it works – CD Practices

- Backwards and Forwards compatible
  - Each component has to function with latest, next or prior version of other components (including DBs)
- Gradual Deployment & Self-Test
  - Deploy new version to one server and perform self-test. If it passes, continue deployment to other servers.
- Feature Toggle
  - Open a new feature by feature toggle configuration
- A/B Testing
  - Open a new feature to a percent of your users. Is it better?
- Exception Classification
  - What exceptions are real errors? What do you care about?
- Small Development Iterations
  - Release frequent – small pieces of functionality

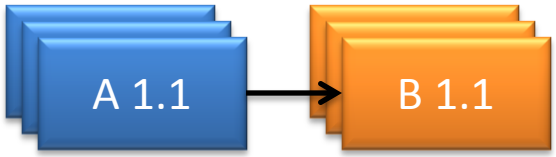
Gradual Deployment

- Assume two components
- We shutdown one and install on it the new version. It is not active yet
- Do self test
- Activate the new server it is passes self test
- Continue deploying the other servers, a few at a time, checking each one with self test

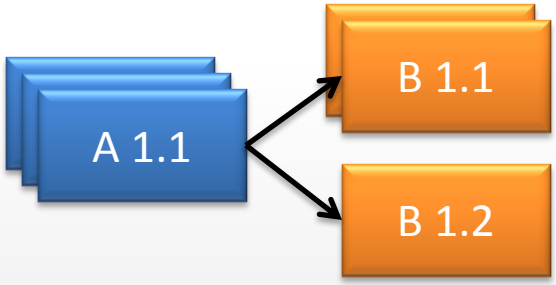


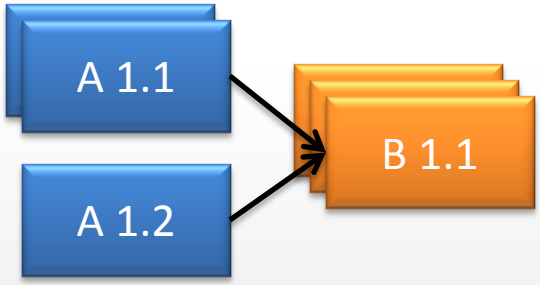
Backward and Forward compatible

- Assume two components

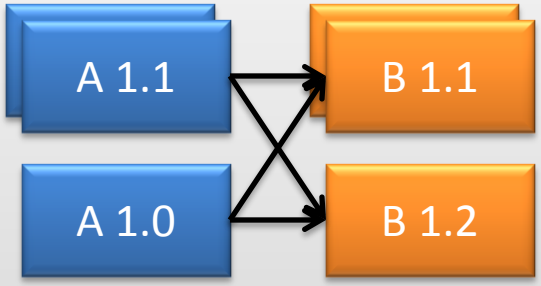


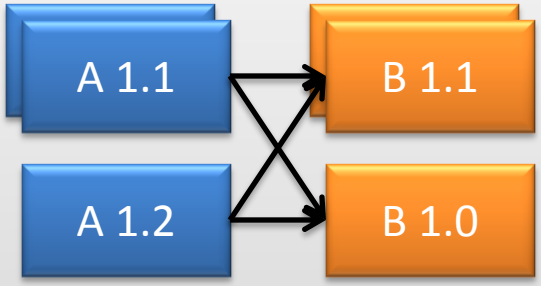
- We release a new version of one





- Now Rollback the other…





Feature Toggle

- Everyone develops on the Trunk
  - It's the developer responsibility not to break anything (requires TDD, CI)
- Every piece of code can get to production at anytime
  - Release of something done by another developer
- What about
  - Incomplete features?
  - Not tested / validated features?
- Feature Toggle to the rescue
  - Unused new code can go to production – no harm done
  - Used new code goes with a guard – use new or old code by feature toggle
- Feature toggle by
  - Static configuration on the server
  - User – open a feature to selected users
  - Any other rule you need

A/B Test

- When we open a new feature
  - It may be production with Feature Toggle
  - It may be a new deployment
- We open the new feature to a certain % of users
  - Define KPIs to check if the new feature is better or worse
  - If it is better, we keep it
  - If worse, we check why and improve
  - If we find flaws, the impact is just for % of our users (kind of Feature Toggle)
- An interesting site effect on product
- How many times did you have the conversion “what is better”?
  - Put the menu on top / on the side
  - If checkout getting inconsistent – do an error or do a best effort (e.g. Amazon)?
- Well, how about building both and A/B Testing?

Exception Classification

- Every application has errors.
  - Some are important, some not so
    - Login failure vs “table not found in DB”
- We classify exception by
  - Business – errors caused by user behavior
  - System – errors preventing our service
  - Level – Fatal, Error, Warning or Recoverable
- The errors are tracked on app-info and monitoring

Usage Summary	
Total Calls	3735039
Total Successfull calls	3726285
Throughput	866.9 rpm
Average Response Time	8.4 mSec
Error Rate	0.00 %
System Errors	5270/0/14/39
Business Errors	3256/0/175/0

c.w.d.s.DocumentService.getDataNode	Rolling Last	2,911 0	3.3 mSec -	29.2 mSec -	0/0/0/0 -	0/0/0/0 -
c.w.h.e.s.HtmlEditorService.cloneDocument	Rolling Last	1,985 0	657.8 mSec -	23,286.4 mSec -	0/0/7/14 -	27/0/0/0 -
c.w.e.e.w.MobileDocumentController.clone2	Rolling Last	1,985 0	658.2 mSec -	23,286.8 mSec -	0/0/7/14 -	27/0/0/0 -

Small Development Iterations

- No Waterfall
- No Scrum
- No Iterations
- No large documents
- Build something small
- When it is ready, deploy it
  - Measure it
  - Then fix it
  - Again
  - And again, until Dev, Product and Customers are happy
- Then start changing it
  - Again, as a small change

Changes the company DNA

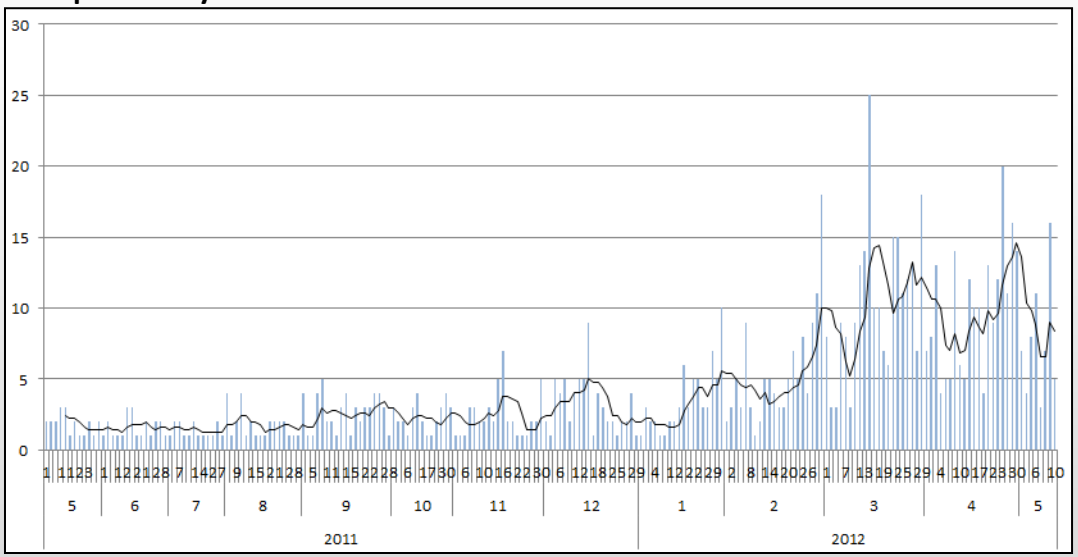
- Changes Product
  - No longer 2 months specification cycles
  - Instead, ask what is the minimal useful feature set that we can open?
  - How can we deploy it within a week?
  - Work closely with developers to answer those questions
  - Think small, fast, agile and about validating ideas with real users
  - Decision making using A/B Testing and measurements
- Changes Operations
  - Don’t do deployments – it’s the developer responsibility
  - DevOps – mixes developers and system responsibilities
  - Responsible for Wix Runtime env
  - Can initiate rollback
  - Build the CD infra, guide developers, DRP, attacks, etc.

Changes the company DNA

- Changes Developers
  - Responsible for building a product / feature
  - Responsible from a product idea (with product) to development, testing (with QA developers) to deployment (with operations) to rollback (with monitoring and BI)
  - DevOps – work closely with operations to enable deployment and rollback, fully automated
  - Work closely with product to find the best simple minimal product to build and validate
- Changes Architects
  - No longer making all the designs
  - Instead, guides developers, works with system and dev, governs important designs (if we make a mistake, we can probably fix it fast)

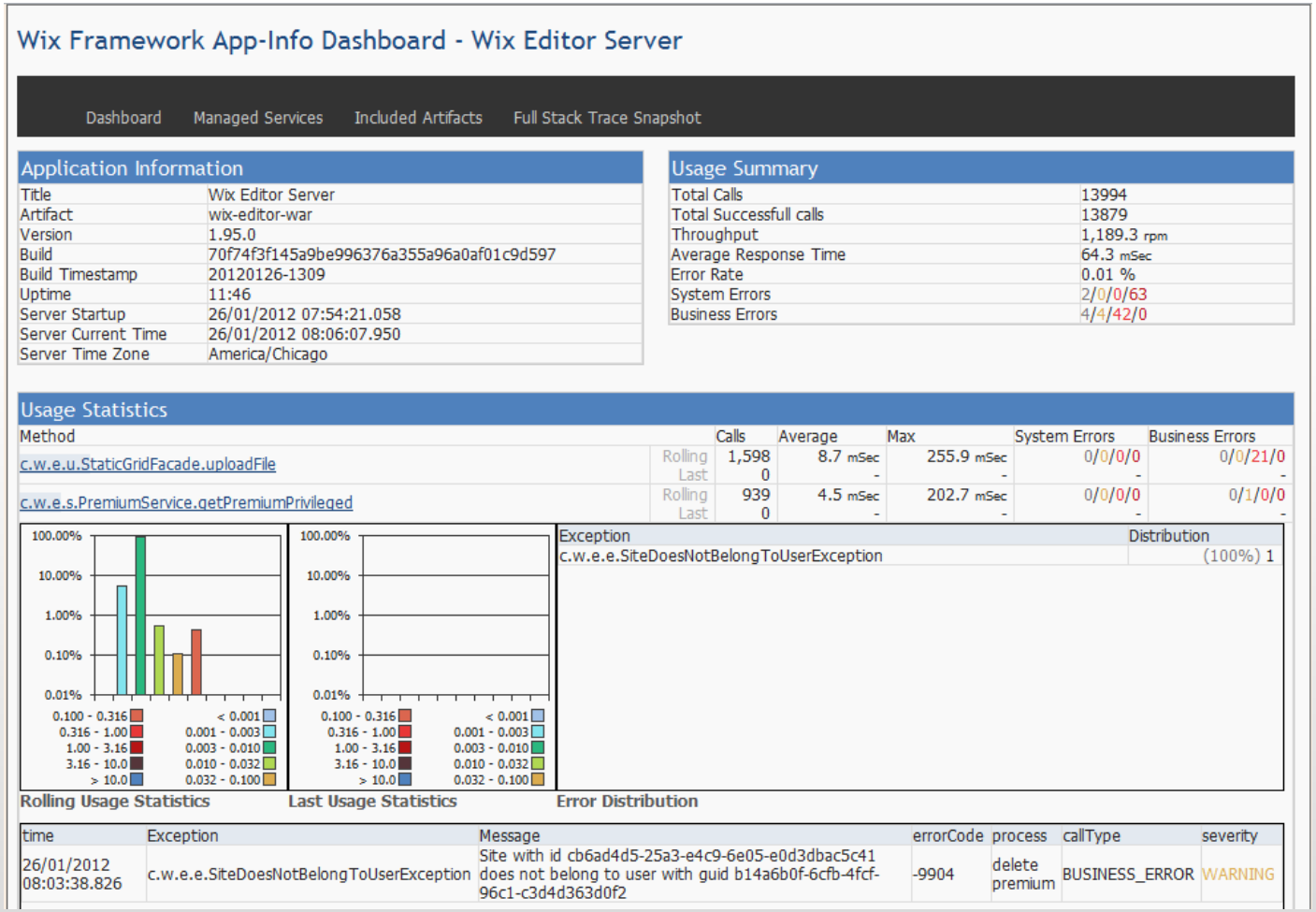
Where are we today?

- We have re-written our flash editor product as an HTML 5 editor
  - In just 4 months
- We are introducing Wix 3rd party applications (developers API)
  - In just 6 weeks
- We are easily replacing significant parts of our infrastructure
- And we are doing 9.5 releases a day!
  - Number of Releases per day



Tools - App-info

- Application Dashboard
  - Application Information, usage and errors on every server



Wix Framework App-Info Dashboard - Wix Editor Server

Dashboard | Managed Services | Included Artifacts | Full Stack Trace Snapshot

Application Information

Title	Wix Editor Server
Artifact	wix-editor-war
Version	1.95.0
Build	70f74f3f145a9be996376a355a96a0af01c9d597
Build Timestamp	20120126-1309
Uptime	11:46
Server Startup	26/01/2012 07:54:21.058
Server Current Time	26/01/2012 08:06:07.950
Server Time Zone	America/Chicago

Usage Summary

Total Calls	13994
Total Successfull calls	13879
Throughput	1,189.3 rpm
Average Response Time	64.3 mSec
Error Rate	0.01 %
System Errors	2/0/0/63
Business Errors	4/4/42/0

Usage Statistics

Method		Calls	Average	Max	System Errors	Business Errors
c.w.e.u.StaticGridFacade.uploadFile	Rolling	1,598	8.7 mSec	255.9 mSec	0/0/0/0	0/0/21/0
	Last	0	-	-	-	-
c.w.e.s.PremiumService.getPremiumPrivileged	Rolling	939	4.5 mSec	202.7 mSec	0/0/0/0	0/1/0/0
	Last	0	-	-	-	-

Rolling Usage Statistics | Last Usage Statistics

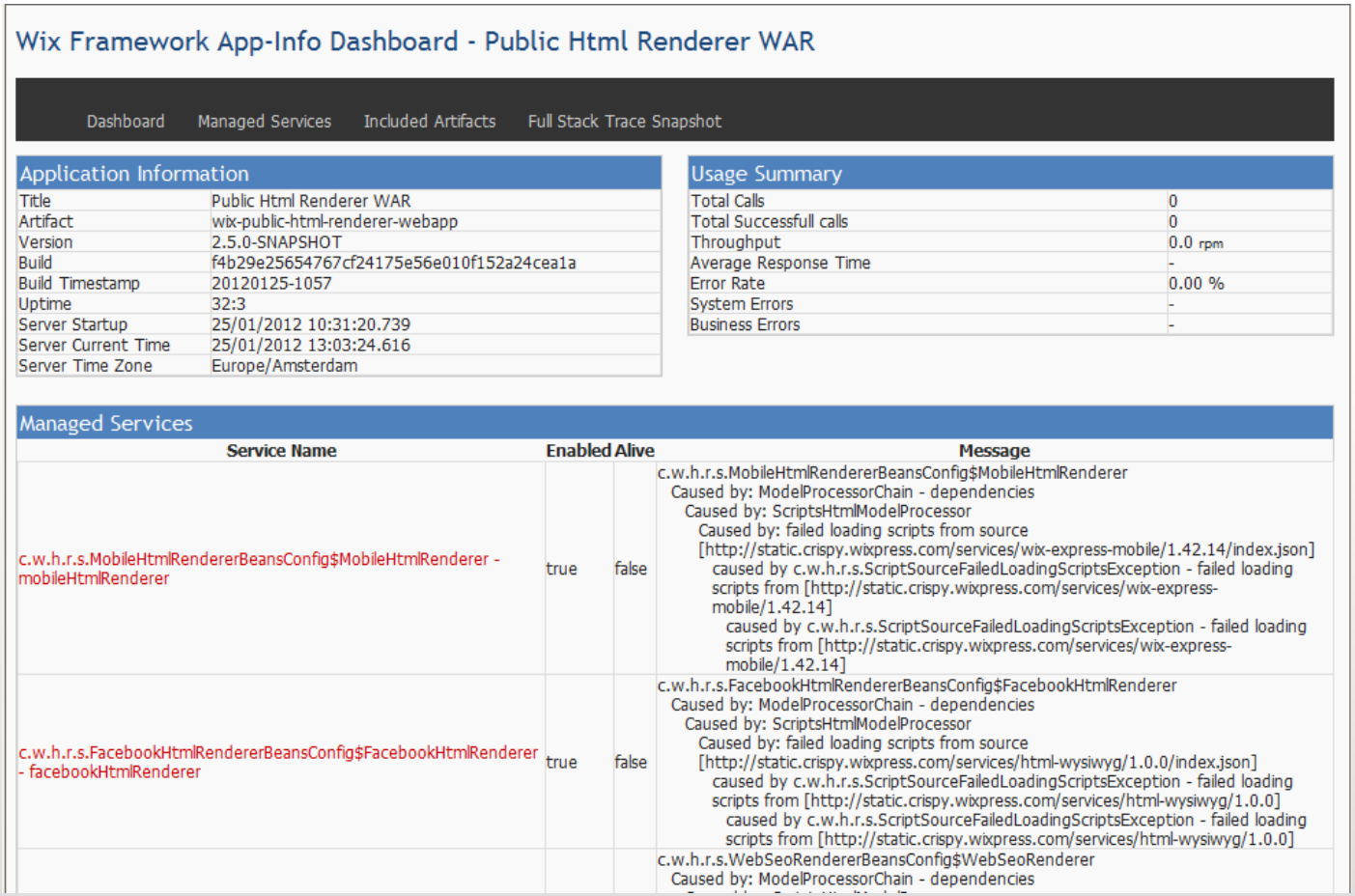
Exception	Distribution
c.w.e.e.SiteDoesNotBelongToUserException	(100%) 1

Error Distribution

time	Exception	Message	errorCode	process	callType	severity
26/01/2012 08:03:38.826	c.w.e.e.SiteDoesNotBelongToUserException	Site with id cb6ad4d5-25a3-e4c9-6e05-e0d3dbac5c41 does not belong to user with guid b14a6b0f-6cfb-4fcf-96c1-c3d4d363d0f2	-9904	delete premium	BUSINESS_ERROR	WARNING

Tools - App-info

- Self-Test –
  - Can my application function?



Wix Framework App-Info Dashboard - Public Html Renderer WAR

Dashboard | Managed Services | Included Artifacts | Full Stack Trace Snapshot

Application Information

Title	Public Html Renderer WAR
Artifact	wix-public-html-renderer-webapp
Version	2.5.0-SNAPSHOT
Build	f4b29e25654767cf24175e56e010f152a24cea1a
Build Timestamp	20120125-1057
Uptime	32:3
Server Startup	25/01/2012 10:31:20.739
Server Current Time	25/01/2012 13:03:24.616
Server Time Zone	Europe/Amsterdam

Usage Summary

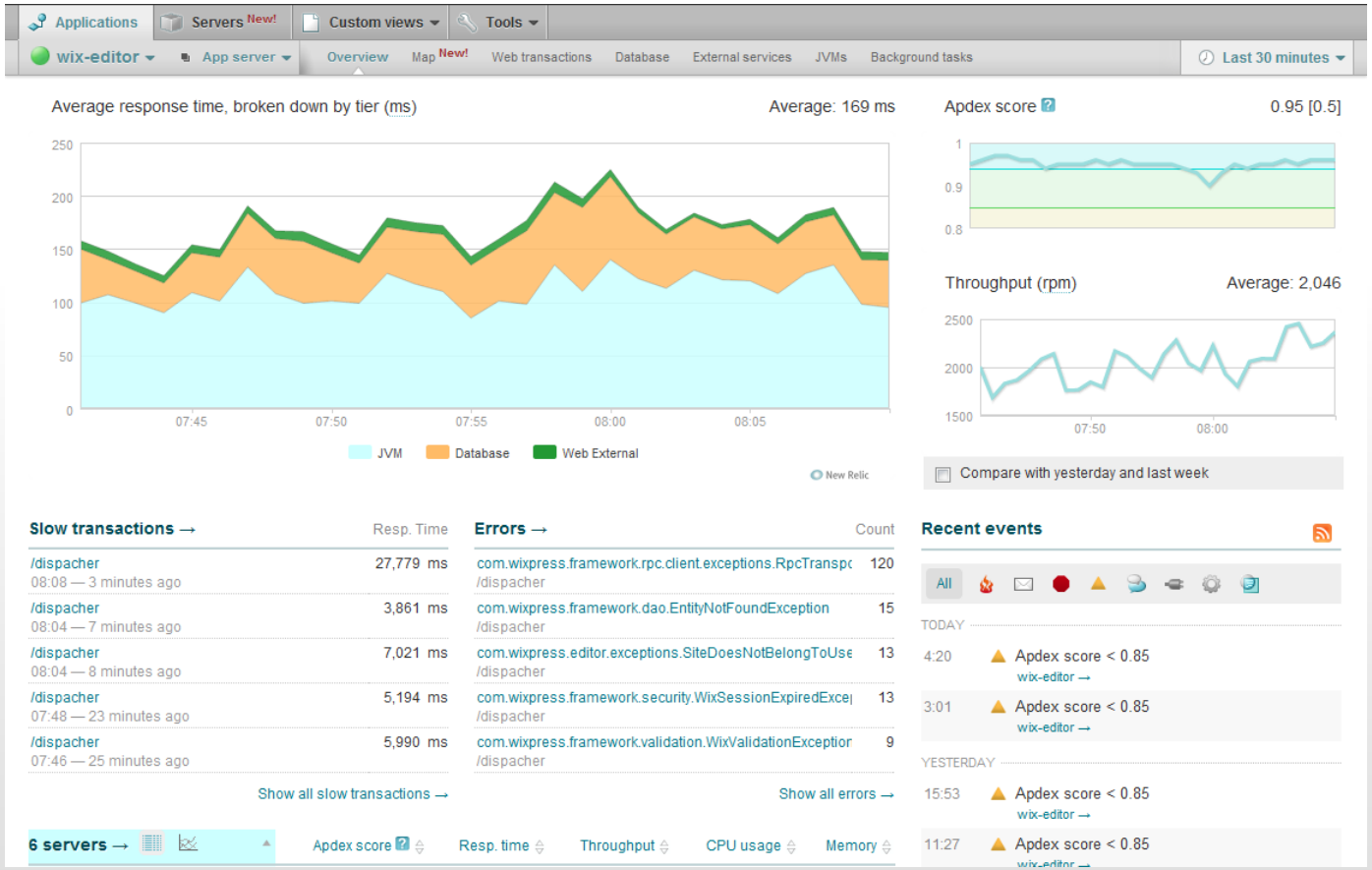
Total Calls	0
Total Successfull calls	0
Throughput	0.0 rpm
Average Response Time	-
Error Rate	0.00 %
System Errors	-
Business Errors	-

Managed Services

Service Name	Enabled	Alive	Message
c.w.h.r.s.MobileHtmlRendererBeansConfig$MobileHtmlRenderer - mobileHtmlRenderer	true	false	c.w.h.r.s.MobileHtmlRendererBeansConfig$MobileHtmlRenderer Caused by: ModelProcessorChain - dependencies Caused by: ScriptsHtmlModelProcessor Caused by: failed loading scripts from source [http://static.crispy.wixpress.com/services/wix-express-mobile/1.42.14/index.json] caused by c.w.h.r.s.ScriptSourceFailedLoadingScriptsException - failed loading scripts from [http://static.crispy.wixpress.com/services/wix-express-mobile/1.42.14] caused by c.w.h.r.s.ScriptSourceFailedLoadingScriptsException - failed loading scripts from [http://static.crispy.wixpress.com/services/wix-express-mobile/1.42.14]
c.w.h.r.s.FacebookHtmlRendererBeansConfig$FacebookHtmlRenderer - facebookHtmlRenderer	true	false	c.w.h.r.s.FacebookHtmlRendererBeansConfig$FacebookHtmlRenderer Caused by: ModelProcessorChain - dependencies Caused by: ScriptsHtmlModelProcessor Caused by: failed loading scripts from source [http://static.crispy.wixpress.com/services/html-wysiwyg/1.0.0/index.json] caused by c.w.h.r.s.ScriptSourceFailedLoadingScriptsException - failed loading scripts from [http://static.crispy.wixpress.com/services/html-wysiwyg/1.0.0] caused by c.w.h.r.s.ScriptSourceFailedLoadingScriptsException - failed loading scripts from [http://static.crispy.wixpress.com/services/html-wysiwyg/1.0.0]
			c.w.h.r.s.WebSeoRendererBeansConfig$WebSeoRenderer Caused by: ModelProcessorChain - dependencies

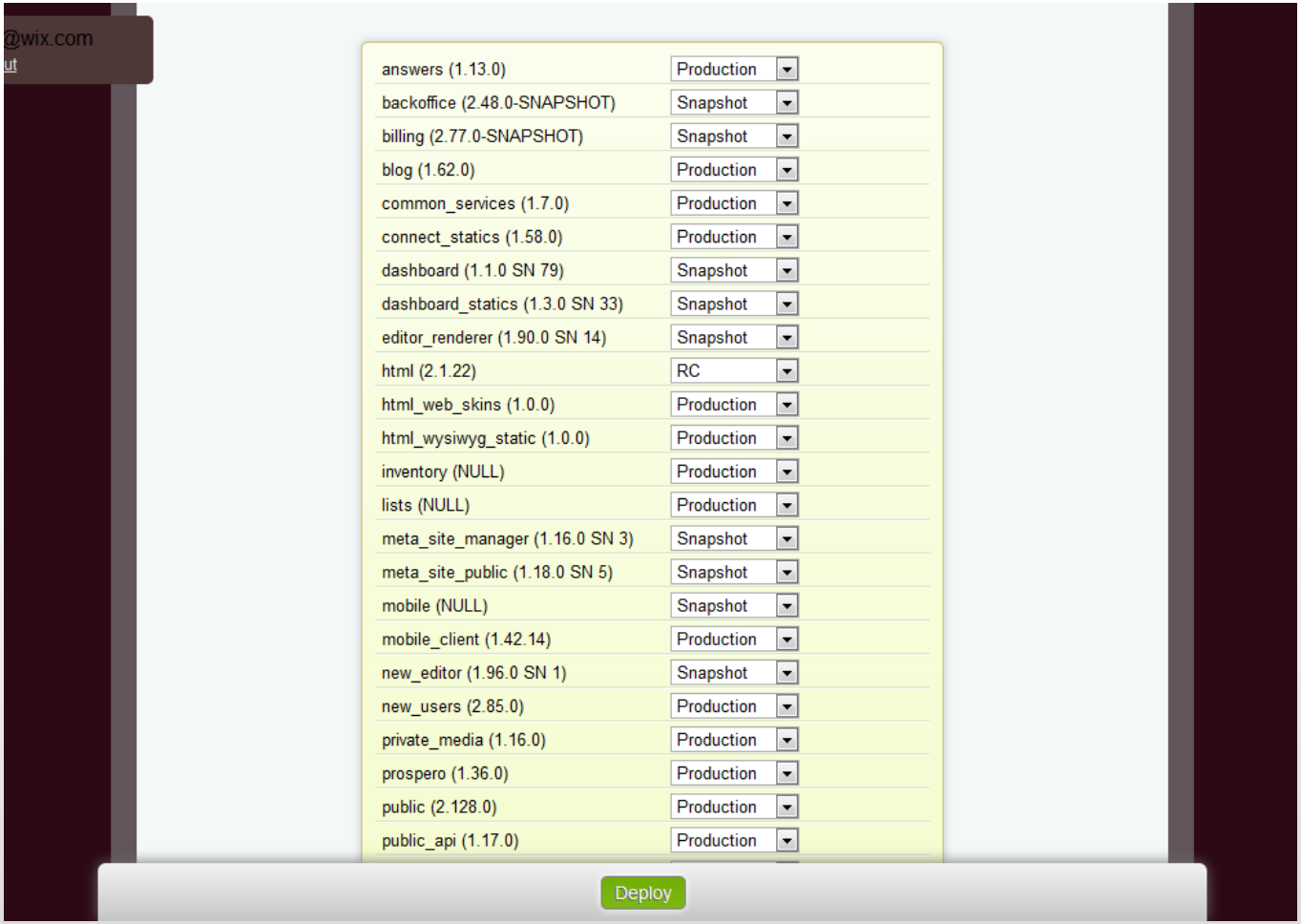
Tools - New Relic

- External Monitoring of applications



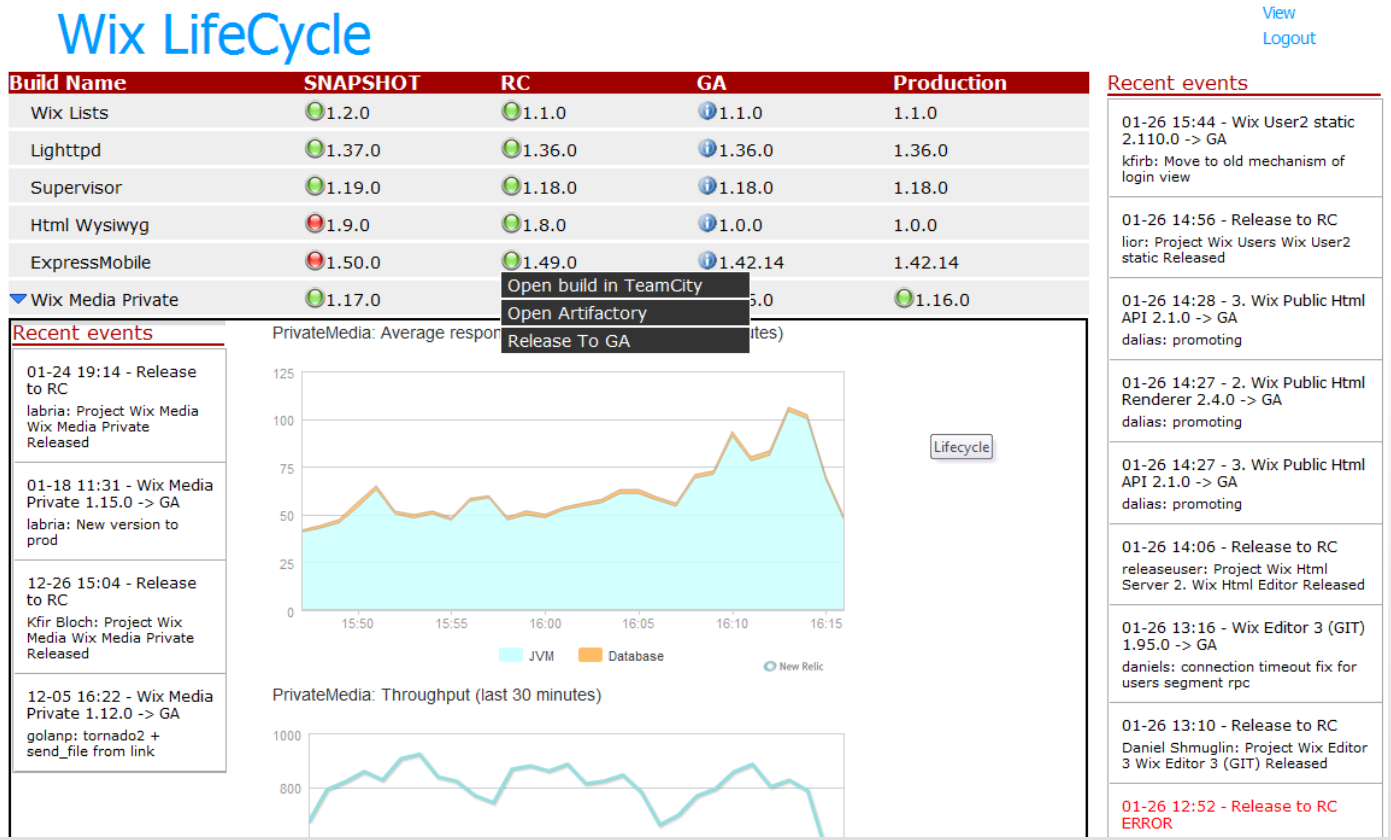
Tools - Chix

- Staging Environments manager
  - Self-service deployment to staging environments



Tools - Lifecycle

- Centralized Dashboard
  - Release RC, GA, Production (with Artifactory, TeamCity)
  - Build status, production status (with chef, sous-chef)




Q&A