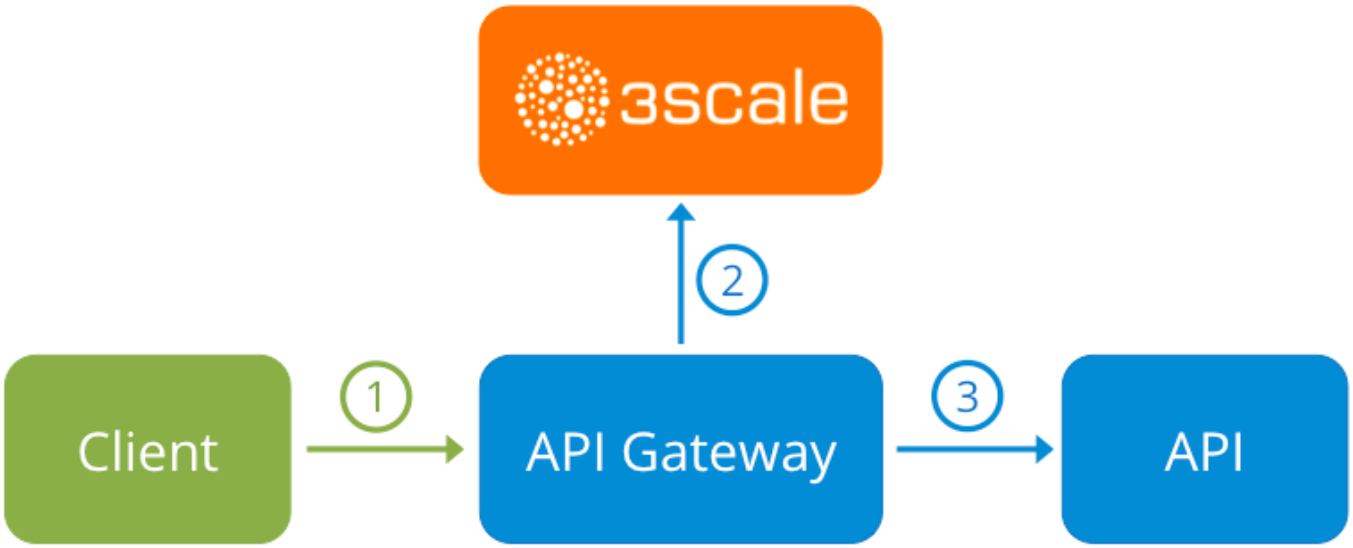
YAZZ.COM ENGLISH EDITION
YAZZ MAGAZINE



Inside this month's issue!

Getting started with Yazz

Your beginners guide to building apps on Yazz Pilot

The Yazz Component Directory

See some of the components you get out of the box with Yazz Pilot

Yazz's "Agile API Integration" solution

Remember that old mainframe that your boss keeps saying will be replaced by new tech every 5 years? Well, it hasn't been replaced yet, and probably will still be around for a long time to come!

Here at Yazz we are always going on about how business units should solve their own problems by building self service apps. But haven't we heard this before? IT vendors have been promising self service tools for decades now, and every time the promise is broken.

To give you an example of this, remember that old mainframe that your boss keeps saying will be replaced by new tech every 5 years? Well, it hasn't been replaced yet, and will probably still be around for a long time to come! What! Why will we still be using our decades old mainframe in the future? Well, the answer is that we will still need that mainframe because it runs business critical stuff and has years of business logic coded in, which takes a lot more time and money to replace than you thought. And why should you replace it when your customer does not care one bit if your business runs on Windows, Linux, Mac, Mainframe, or has millions of magical elves running around doing things manually by paper! Your customers just care about whether you can deliver the service that they pay you for!

However, enterprises still keep buying new tech, and because of this constant adoption of new technology most medium to large size enterprises have an expanding inventory of IT tools. And IT leaders have noticed that the pile of so called "legacy" systems keeps getting larger and larger each year! IT leaders also realise that you can't just replace one set of IT systems with another. 90% if the time you need to keep running the old systems in parallel with the new ones.

Yazz Pilot does not try to offer a "silver bullet" solution to fix this multiple systems problem. Instead, Yazz Pilot tries to embrace the chaos of multiple systems, and bring them together using "Agile API Integration". This means that instead of replacing existing systems with Yazz Pilot, it glues existing systems together using self service APIs and integration tools to deliver new business value.

"Yazz Pilot does not try to offer a silver bullet."

Robert Galbraith

Contents

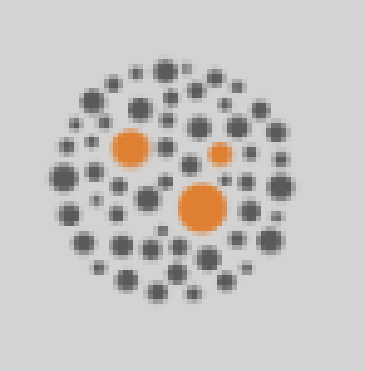
The architecture of Yazz Pilot

We describe the different components of Yazz Pilot and how it works internally

Test Yourself

Take this quick quiz to test your knowledge of Yazz Pilot

3Scale Component Review



Description	The 3Scale component allows you to call APIs on a 3Scale server
Price	Free
Supplier	Yazz
Pros	Allows you to access 3scale APIs without needing Postman or other tools
Cons	Only a limited subset of the 3Scale admin API available, the listing and calling of APIs
Features **2/5**	A very limited feature set which only allows you to browse already existing 3scale APIs and call an API
Ease of use **4/5**	The easy drag and drop interface is great, but you still need to understand how 3Scale works and get the 3Scale Admin API key to use this
Value for money **5/5**	You can't argue with free!
Overall **3/5**	This component may be a useful addition to any 3Scale developer's toolkit

Verdict

3/5

Sometimes it can be very tricky to develop a web app that connects to an API via 3Scale, requiring multiple developer tools. The 3Scale component simplifies this by giving access to the available 3Scale APIs via a single component.

Why 3Scale?

In the last few decades companies have collected huge amounts of data on their customers, products, and services. They have realised that this treasure trove of data has many internal uses, and also that it can be packaged as services and sold to external partners.

To expose this data from internal company systems the most common way is to expose the data via APIs. But since these APIs come from many systems located in different places there needs to be a way to find them all from one place. But to build this as a system is expensive, as it means that there needs to be a way to aggregate the data and control access to it. This is time consuming and costly to build (Just ask Amazon who spent billions of dollars doing this).

However, nowadays companies don't need to spend large sums of money to expose internal data and services. 3Scale was a pioneering company providing API aggregation as a product offering.

3Scale's product was so successful that in 2016 3Scale was acquired by Red Hat and subsequently open sourced. It is now owned by IBM and it's popularity has only grown with time.

What is 3Scale?

3Scale is a system to manage APIs for internal and external users. It allows:

- Ability to secure API services
- Ability to charge external users and partners for access to APIs
- Linking to companies internal and external LDAP and user directory services to automate access

A typical use case for 3Scale could be a Pension company where someone in customer support wishes to get certain information about a customer that is stored internally. Traditionally the customer service representative would need to contact the IT department to build a system to do this, which would take a long time and be very costly.

What is an API?
An API is an Application programming Interface. It is a way for one IT system to talk to another IT. An example could be a weather forecasting website which uses an API to get the weather:

http://weather.com/get-temperature/city/stockholm

Which could get the current weather as JSON:

```
{
   City: Stockholm
  Temp: 34
}
```

JSON
Javascript Object Notation (JSON) is an easy to use format for data, simpler to read than XML

However, if the Pension company exposes all its internal systems as APIs, then 3Scale can be used to access customer data via an API. This data can be made available to everyone by listing it in a central company directory.

Who uses 3Scale?

3Scale is mostly used by large enterprise customers since they have so many IT systems and APIs, and 3Scale excels at combining multiple APIs and packaging them up into products. Most smaller companies simply don't have enough APIs to require 3Scale.

How to get 3Scale

There are multiple ways to get 3Scale, the simplest being to sign up for a 3Scale account at https://www.3scale.net.

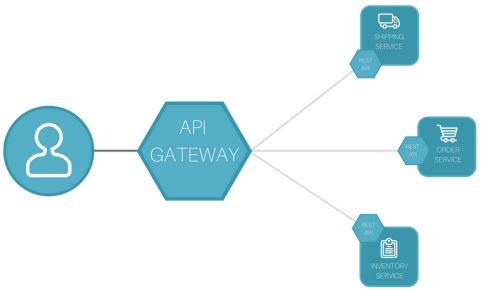
However, most companies will deploy 3Scale directly onto their own servers or in the cloud via a Kubernetes system such as Red Hat's OpenShift.

There is also a service called RHMI which bundles 3Scale and many other Red Hat offerings into a platform as a service.

We will not show how to install 3Scale, as installation can depend on many factors of how it is used, in the cloud or onsite. Also there are many configurations of 3Scale which complicates installation as well.

How 3Scale works

To enable many internal services to be exposed there is the concept of an API gateway.



3Scale has the following concepts:

- **API -** A set of HTTP API calls that will be offered as a service
- **Application Plan** - An application plan defines a set of rules or limits for accessing an API. Each API will need at least one Application Plan. In the case of BigCorp services all application plans will initially be "free", as APIs are provided for internal use, although in the future if BigCorp provides their APIs to outside external customers then BigCorp may also provide Paid for application plans

What is Postman?
Postman is a very popular tool testing APIs

OpenAPI Specification
(formerly Swagger Specification) is an API description format for REST APIs. An OpenAPI file allows you to describe your entire API

https://github.com/3scale
The open source repository for 3Scale

- **Application** - This links a 3Scale end user to an application plan
- **Audience** - End users and applications that will subscribe to APIs
- **ActiveDocs -** This is Red Hat's version of 3Scale documentation
- **Developer Portal** - This is the directory of all the 3Scale APIs available

How are 3Scale apps usually developed?

To develop something with an API which is already available in 3Scale, a develop usually has several steps:

- Go to the 3Scale Developer Portal and choose an API
- Request an API Key to use the API
- Test the API with the Developer Portal and a tool like Postman
- Create a web application using NodeJS, .Net, Spring Boot, or some other framework to access the API

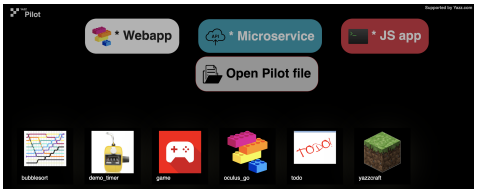
This can be quite a long process, especially the development step to connect to the API, and requires a developer to be constantly switching between Postman and their developer tool.

How does the Yazz 3Scale component help?

The Yazz 3Scale component greatly simplifies the development process of 3Scale apps by connecting the Yazz Pilot development tool directly to the 3Scale admin server. This allows the developer to easily select which APIs they wish to call and use drag and drop to build apps.

Try the 3Scale Component!

To try out the 3Scale component go to the Yazz Pilot editor:

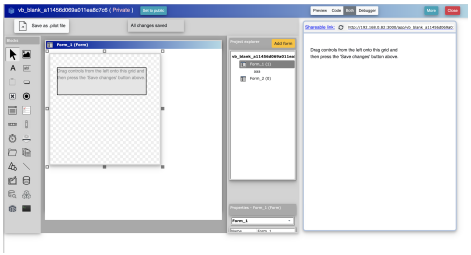


Then create a new app by pressing the **Webapp** button:

ActiveDocs
This is the interactive API documentation for 3Scale, based on OpenAPI

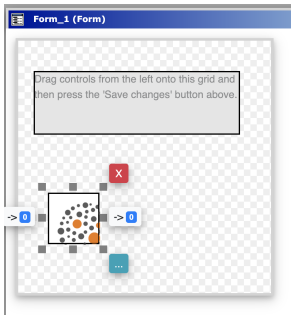
APICast
The 3Scale gateway. This is the component that is used to route API traffic

This should take you to the visual editor:



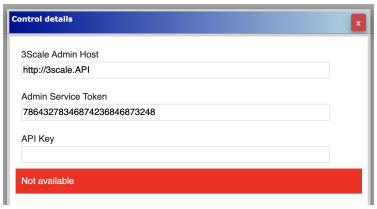
In the list of components on the left you should find the 3Scale component:

Drag it onto the Design Grid:



And then click on the green Details Button (...) at the bottom right:

: which should bring up the Component Details view for the 3Scale component :



If you look closer you will see that the red bar says **Not Available**. This is because the 3Scale component is failing to connect to the 3Scale admin server to get a list of available APIs:

Not available

Taking a closer look at the fields you can see that the the 3Scale component needs to know how to find the 3Scale Admin Server:

3Scale Admin Host

http://3scale.API

Admin Service Token

786432783468742368468732 48

API Key

The fields mean the following:

- **3Scale Admin Host** - The Domain Name of the the 3Scale Admin server
- **Admin Service Token** - This is a like a password that is used to access the admin host
- **API Key** - This is used by applications to call APIs and acts like a password

To fill in these fields you need to log in to 3Scale (or ask your system administrator) for the values.

RHMI
Red Hat managed Integration. Red Hat's Middleware offering available in the cloud

3Scale Application Plan
A set of services, eg: Free plan, Paid plan, Deluxe plan

URL
Used to identify where a web page or server can be found

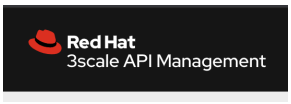
Domain Name
A human friendly way of writing an IP address. Ie:

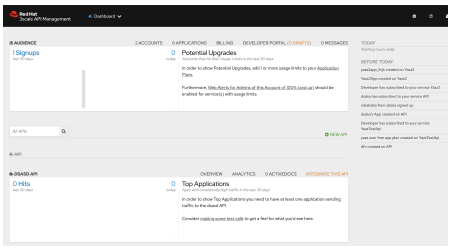
Apple.com

:instead of:

4.3.23.3211

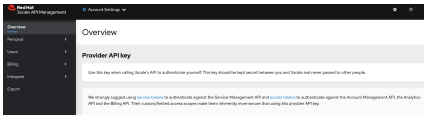
If you need to get this information yourself then log in to your 3Scale admin console in your browser:



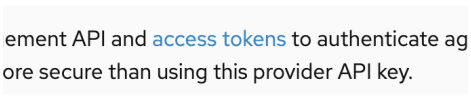


Then, at the top right click on the Settings icon:

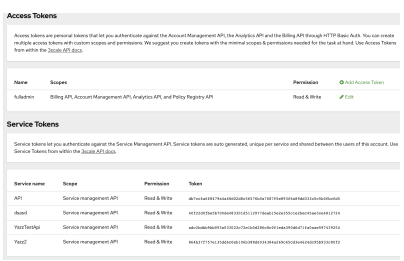
This will take you to the settings page:



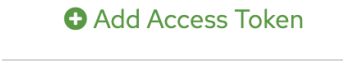
The settings page has an **Access Tokens** link. Click it:



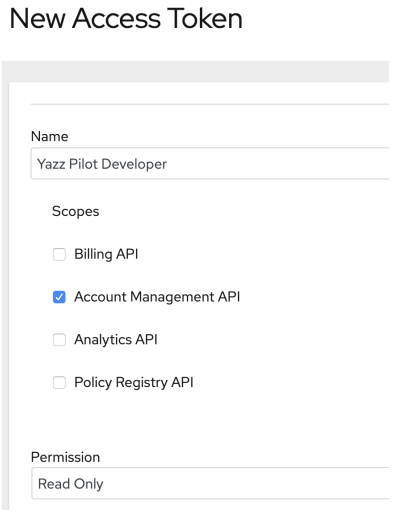
You should see a page that lets you manage your access tokens:



From here, click **Add Access Token** to create a new Access Token:

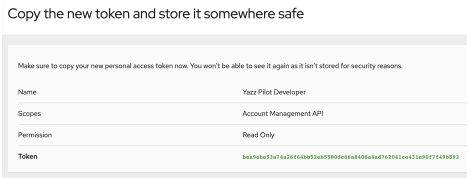


Then fill in the fields as shown below:



Then press the **Create Access Token** button at the bottom right:

And you should see the following screen:



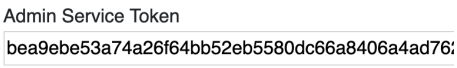
In the bottom right you will see the Access Token:

Copy the Access Token to the clipboard.

3Scale Application
A 3Scale application connects a subscriber to an Application Plan

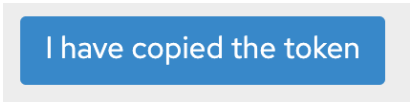
3Scale Account
An account is someone who has subscribed to a 3Scale service. This could be an end user, or it could be an application

And paste it into the 3Scale Component in Yazz Pilot:



It is probably a good idea to write the Access Token down in a safe place as you won't be able to access it again from 3Scale.

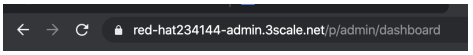
Then click this in 3Scale:



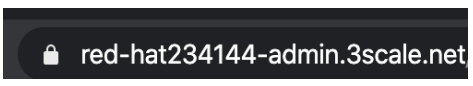
: This should close the access tokens page in the 3Scale admin console.

Just to give some context, later on in the Yazz Pilot editor this Access Token is used by the 3Scale Component to see the APIs that are available in the 3Scale system.

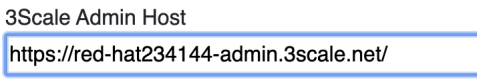
Next, we need to tell Yazz Pilot where the 3Scale admin console is located. So, in the web page where you have the 3Scale admin console open go to the address bar at the top of the browser:



And copy the first part of the URL to Yazz Pilot:



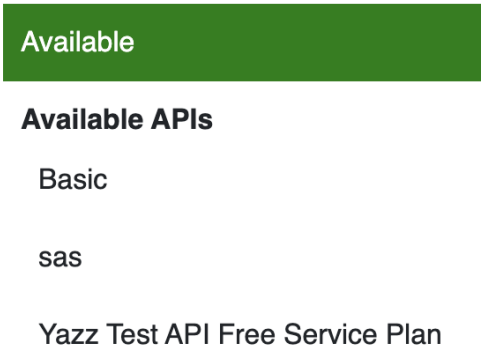
Paste the URL here in Yazz Pilot:



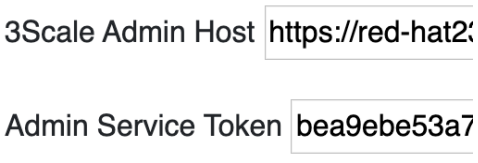
And then click anywhere on the Yazz Pilot form and you will see that the status changes from:

: to :

This means that Yazz Pilot has now connected to the 3Scale system. In Yazz Pilot you should now see a list of the available APIs:



If you don't see any APIs, or if the status still says **Not available** then go to the details pane of the 3Scale component in Yazz Pilot:

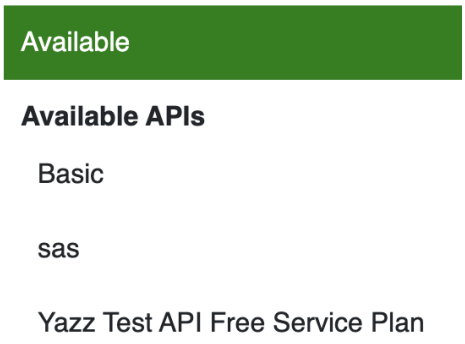


: and make sure that the URL and admin token were copied correctly from the 3Scale admin console.

3Scale Admin Host
This is the address of the 3Scale Admin server

3Scale Access Token
This is a token that grants access to the 3Scale admin server, which acts like a password

Now, assuming that Yazz is connected to 3Scale, the developer can now choose an API to use, which is this case will be **Yazz Test API Free Service Plan**:



At this point we need to pause, since we must understand that companies will want their APIs secure, so that their API are only accessible to certain users or accounts.

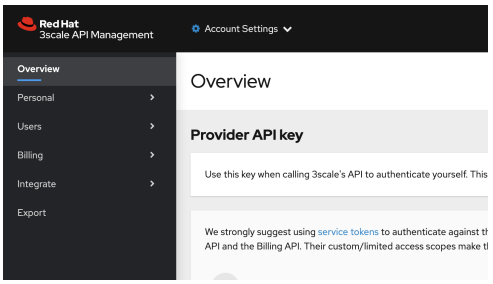
So how can we control access to a 3Scale API? Well, there are many ways such as using Keycloak, or connecting to OAUTH, but the simplest way is to use API Keys, which are randomly generated pieces of text (also known as API keys) which 3Scale generates, and any client calling the API in 3Scale will need to include the API key, to prove that it has access.

While this is not the most secure way of doing security, API Keys are the simplest, as they can be provided with every API call, something like this:

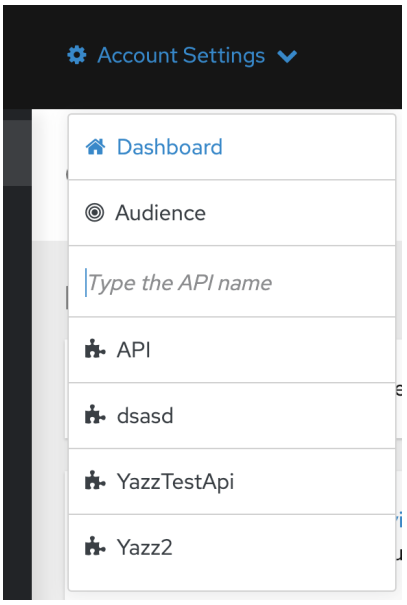
`https://yazz_test_api.3scale.com?user_key=`**`ADD_API_KEY_HERE`**

So how do we get the API Key for **Yazz Test API Free Service Plan**?

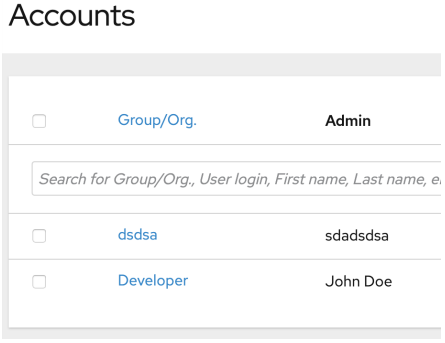
Go to the 3Scale Admin console:



At the top you will see a menu, which if opened looks something like this:



Select **Audience** at the top to see a list of who is subscribed to the API. You should see a list of **Accounts**:

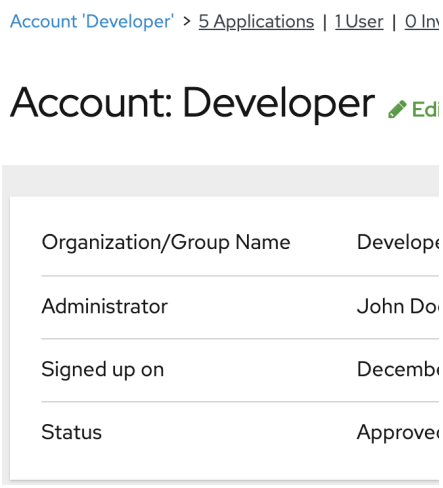


Audience
This is the term in 3Scale which represents end users and machines which need to subscribe to and call APIs

Developer Portal
This is the directory of all the 3Scale APIs available

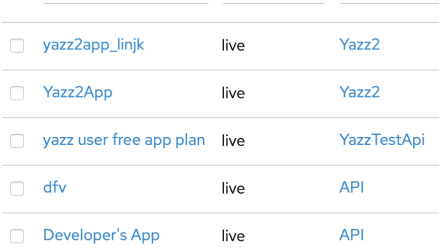
In this case there are only two accounts in the system. Select **Developer**:

This should take you to the details of the Developer account:



At the top it says **5 Applications**. This means that the developer has access to 5 APIs:

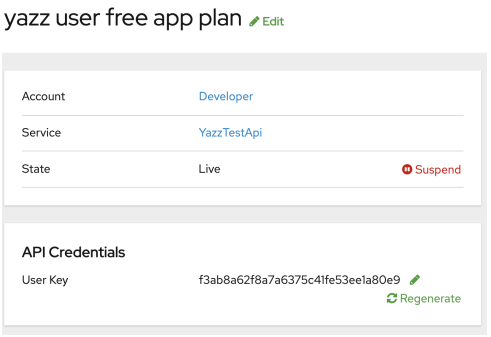
Click **5 Applications** to see the list of available APIs for the **Developer** account:



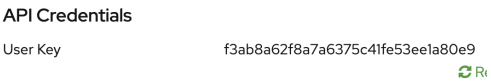
These are APIs that have already been set up in the 3Scale system.

Click **Yazz user free app plan**:

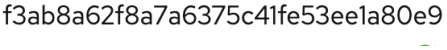
: and you will see a screen showing details of the API:



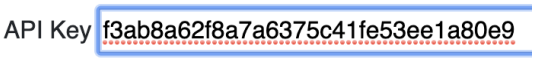
At the bottom is the **API Key** for the API:



Copy the **API Key**:



And paste it to the **API Key** field in the 3Scale Component in Yazz:



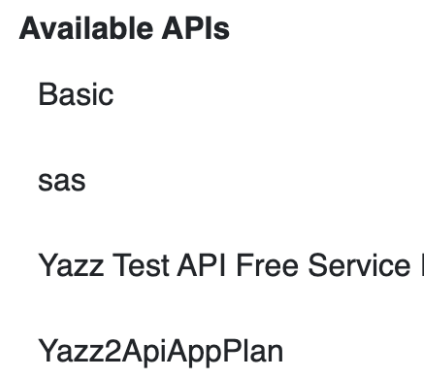
This means that the 3Scale component in Yazz Pilot now has the security token needed to call the **Yazz user free app plan** API from 3Scale.

Free App Plans
This is where companies allow their APIs to be used free of charge.

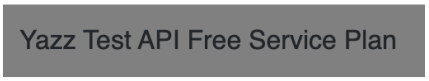
Usually free plans are used for internal users

Paid App Plans
This is where companies charge external companies to use their APIs

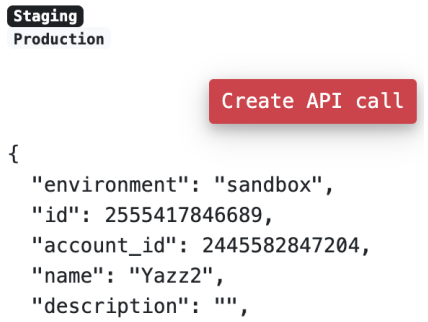
Next from the list of APIs in the 3Scale component:



Click on the **Yazz Test API Free Service Plan** API:



And the right hand side of the 3Scale Component should show the API details:



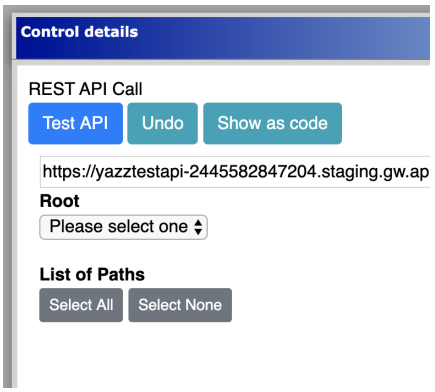
Click on **Create API call**:

Yazz should automatically add a **Rest API call** component to your application.

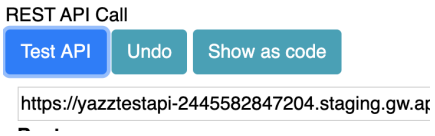
But what does this **Rest API call** component do?

The newly created **REST API call** is used to connect to the 3Scale API.

So after clicking **Create API call** you should be automatically taken to the Component Details view for the REST Api component that was created:

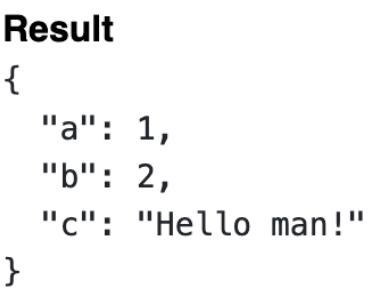


At the top you can see the URL that the REST API will call:



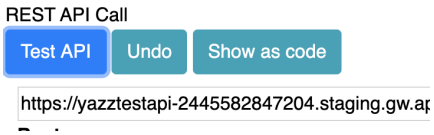
This is the URL of the 3Scale API that Yazz will call. Test it by pressing **Test API**:

And you will see a JSON result:



Alternatives to 3Scale
There are alternatives to 3Scale like WS02 and APIGee

Note, that you can also test this URL in a web browser, by copying the URL shown in Yazz Pilot:



And pasting it into the address bar of a web browser:

In the web browser you should see something like:

```
{"a":1,"b":2,"c":"Hello man!"}
```

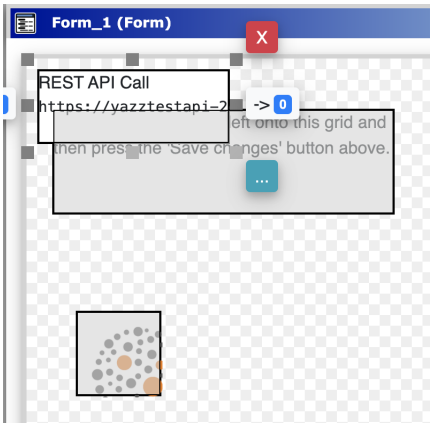
Note that the result from the web browser is the same as the result from the API call in Yazz Pilot.

Now that you have an API from 3Scale in your Yazz app, you can access it programmatically.

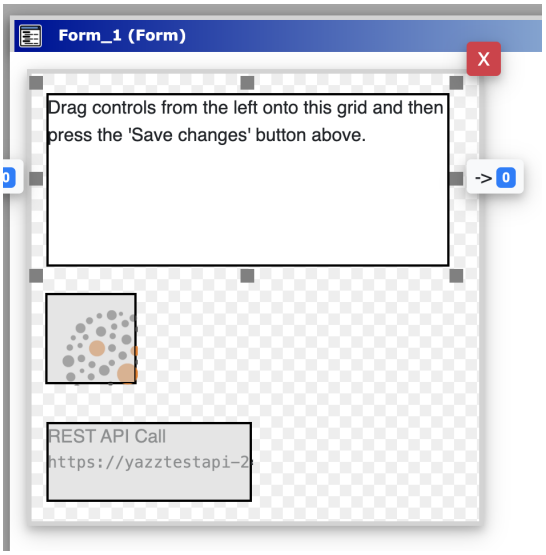
In the Yazz editor close the detail screen by pressing the Red button at the top left corner:

The details screen should close and take you back to the Design Grid.

In the Design Grid you will notice the newly created **REST API Call** component in the top left:



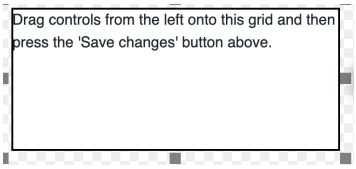
Clean up the form a bit by moving the components around:



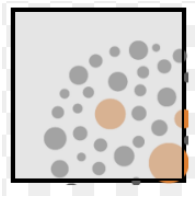
So now you can more clearly see the different components.

REST API Call Component
This is a component in Yazz Pilot which is used to call an API

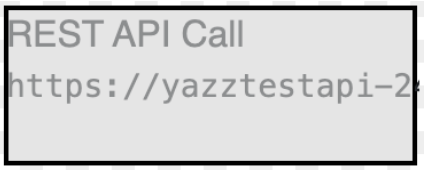
First is the Label component at the top (called **aaa**) which was automatically created when we made the new web app in Yazz.



In the middle you should see the 3Scale component we added to browse the 3Scale APIs.

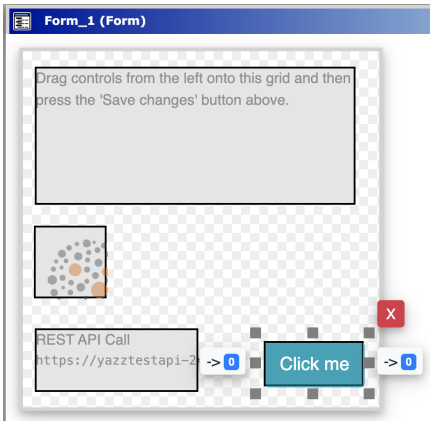


And finally at the bottom you see the REST Api component that was automatically created by the 3Scale component. This is the actual component that will do the work of calling the API:



Next, add some interactivity. In the list of components on the left find the Button component:

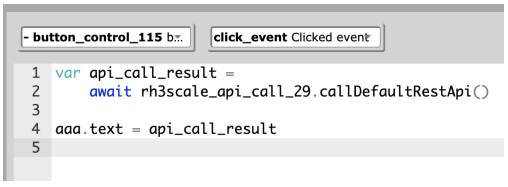
Drag the **Button** component to the Design Grid to make a new button:



Next, go to the properties for the button component, and find the **Clicked event**:

Click the three dots (...) next to the **Clicked event** to edit the event:

Then enter the following code into the editor

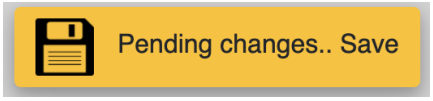


This code means that when a button is clicked, call the 3Scale API that we defined earlier, and then display the result.

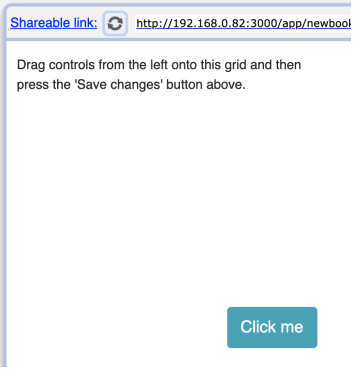
Showing text in Yazz
The label component in Yazz can be used to show text, as shown by the code:

aaa.text = "SOME TEXT"

Next, save the app by pressing Save:



: and then you will see the app displayed in the Preview window:



Press the **Click me** button and you should see the result:

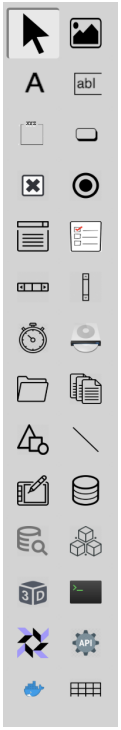
Well done! You have made your first 3Scale app.

Conclusion

In this review we discussed 3Scale and showed you how to use the Yazz Pilot 3Scale Component to build a simple end user application.

API Keys
A simple way to perform 3Scale security by passing a security key with every request

The Yazz Component Directory



Yazz Pilot lets you build web applications using a drag and drop UI, and has a set of pre-built components that can be used.

The components are shown on the left hand side of the UI editor. Most of them are common building blocks for building web apps like buttons, text input fields, and database components.

Fun fact:
We didn't invent the concept of pre-built components, or the component toolbar. In fact we copied the idea from a discontinued Microsoft product called Visual basic (shown on right).

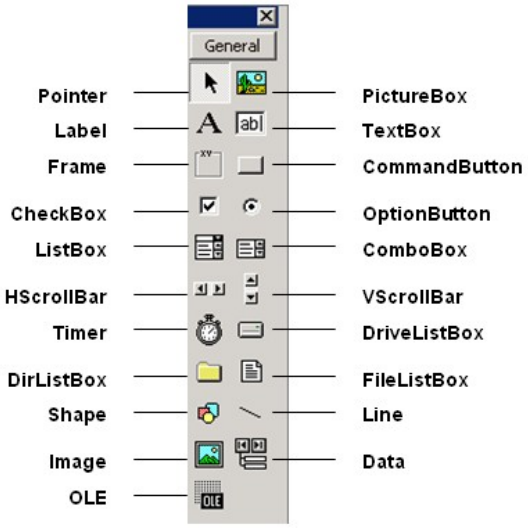
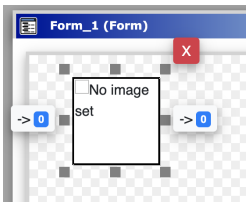


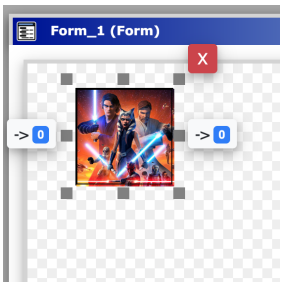
Image component

The image component is used to add an image to your application.



In the Design Grid you should select a file to upload an image:

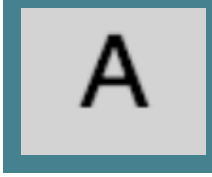
The image will be previewed in the Design Grid:



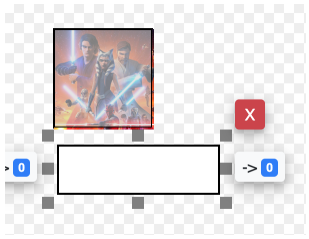
If you Save the change then you can see the final image in the app preview:

MAY THE FORCE BE WITH YOU!

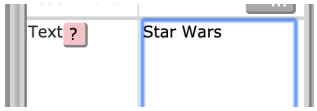
Label component



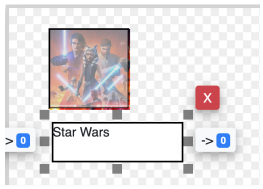
The label component adds text to your application.



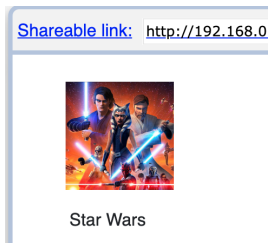
Once you have added a label component to your application then you should enter some text in the Property Inspector for the **Text** field:



Here you can see the text in the Design Grid:

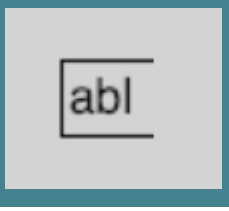


If you update the application then you can see the result:

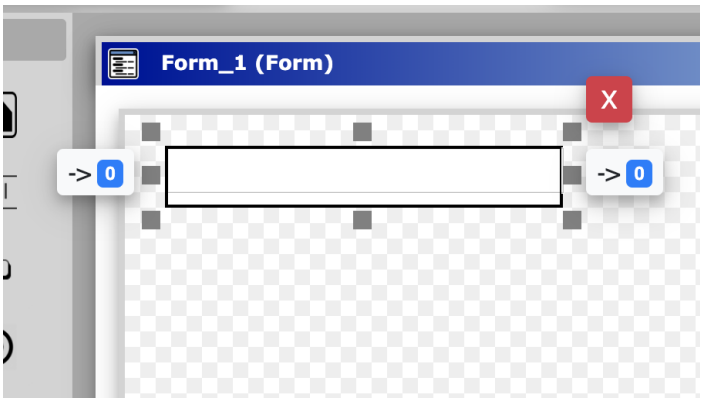


Well done! You have now added some text to your application.

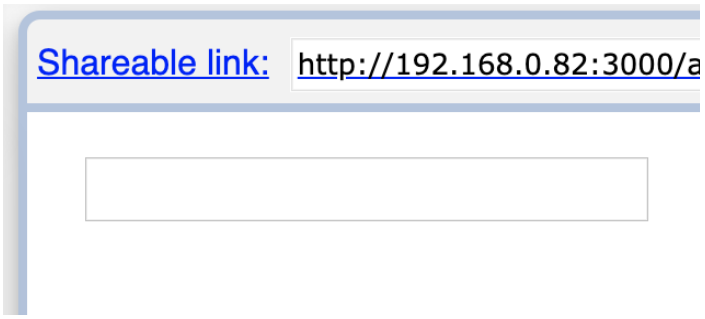
Input component



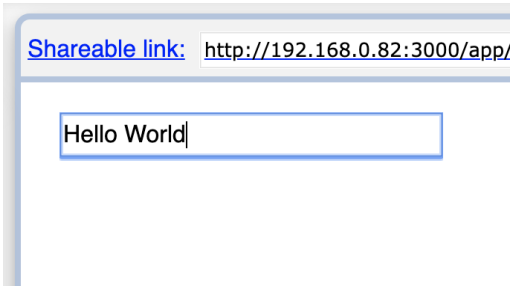
The input component is used to get user input.



In the App Preview the input component should look something like this:

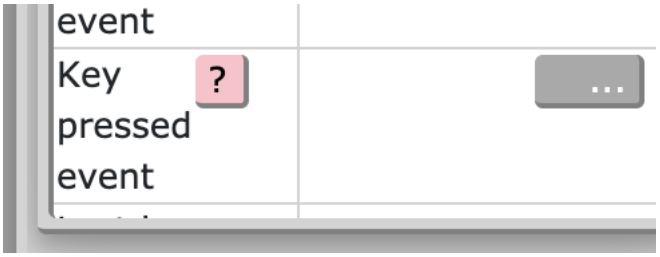


Type in some text to see that it works:

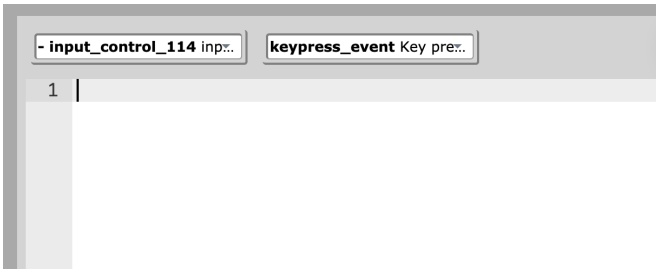


Next we want something to happen when we press the Enter key.

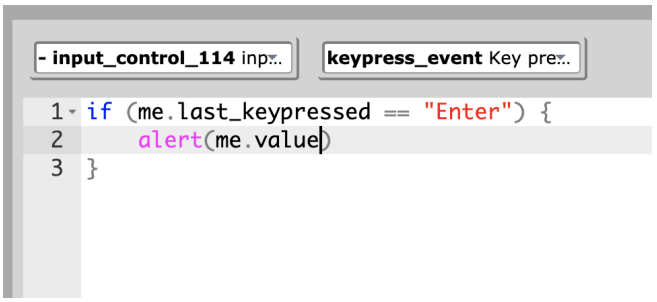
Go back to the Property Inspector for the Input component, and you should see this:



Click on the 3 dots (...) and you should see an event editor:

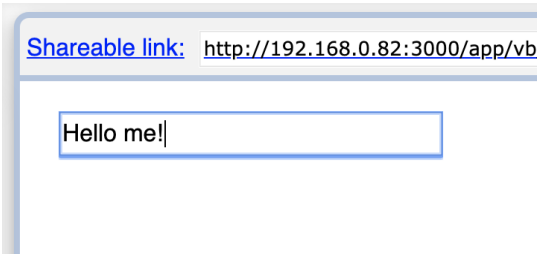


Enter the following code:



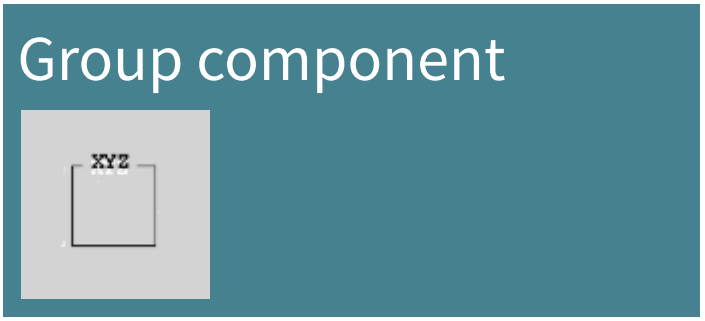
This says that whenever the user presses the Enter key then show a message box displaying the text that the user typed in.

Try it by saving and previewing the app:



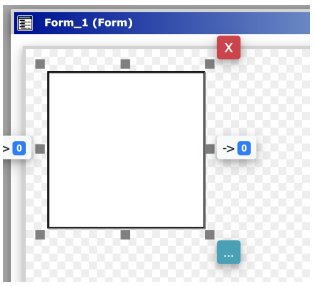
Enter some text into the input field and press the Enter key. A popup similar to the following should show:

You have now covered the basics of the input component.

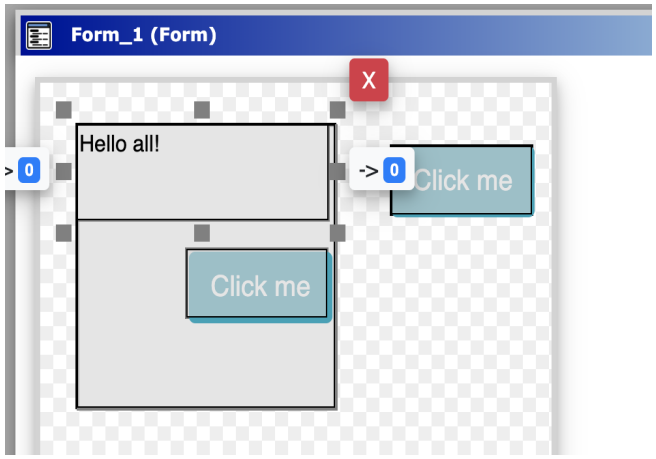


The group component is used to group several components together as one visual unit.

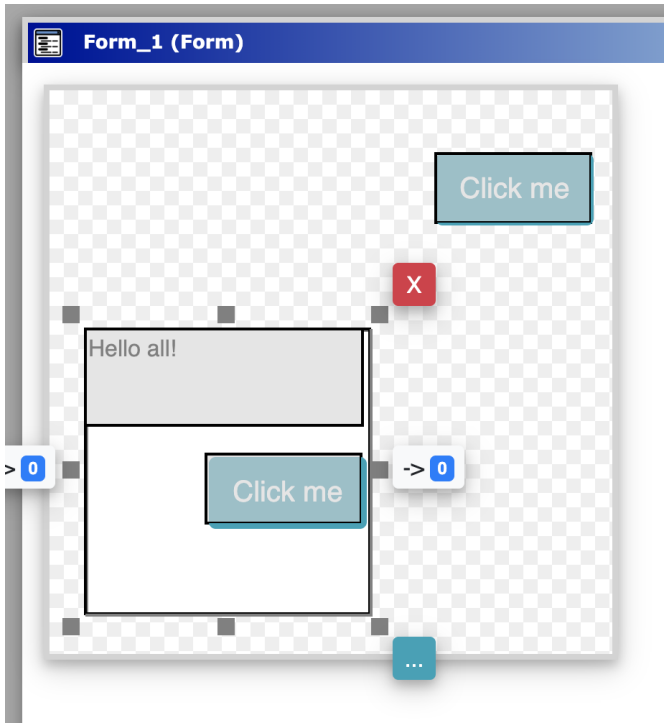
As an example, place a group component onto the Design Grid:



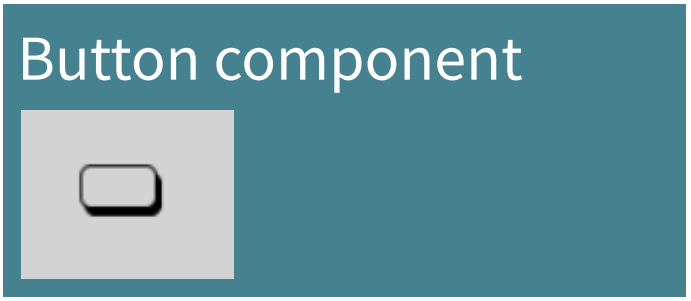
Then add some other components onto the Design Grid, placing some of them some inside the group component boundaries, and some outside the group component:



Next move the group component inside the Design Grid and you should see that all the components inside the group component move as one unit:

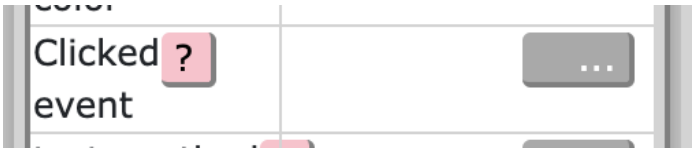


So a group component allows you to group related visual elements, for example when creating an input form.

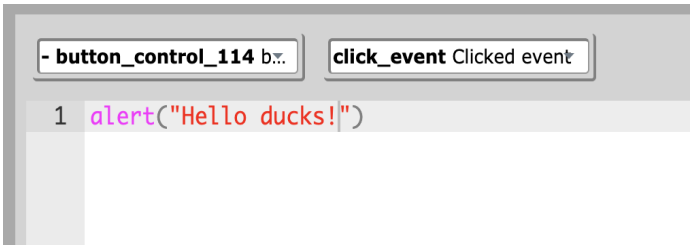


The Button component is used to display a clickable button.

Once a button has been added to the Design Grid then click on the three dots next to the click event in the property inspector:

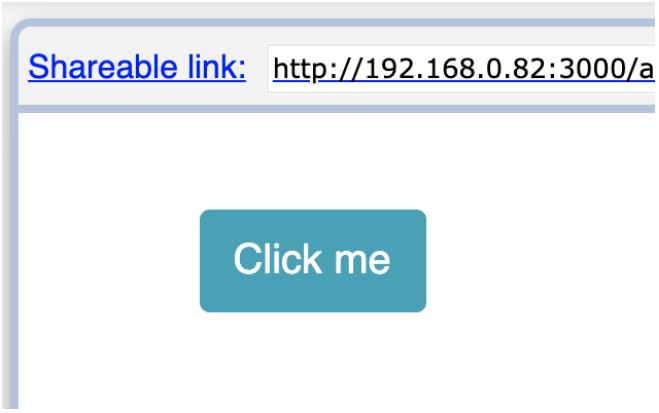


Then add the following code:



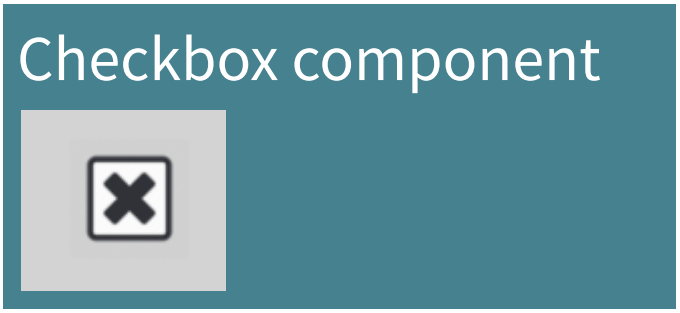
This says that when the user presses the button show the text **Hello ducks!** in a popup message box.

Try it by going to the app preview and pressing the button:



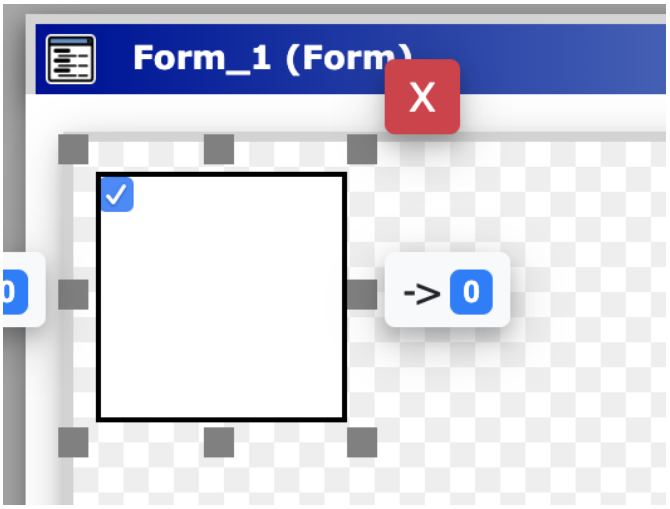
You will see the following message

That illustrates the usage of the Button component. Of course in most cases you will do more than just show a message box. Usually the Button component will be used to get information from fields in a form and perform some action.

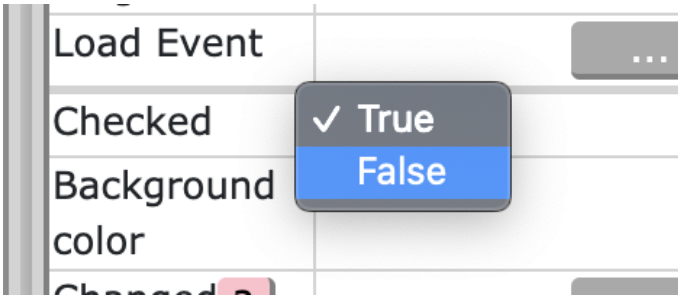


The checkbox component is used to display a checkbox.

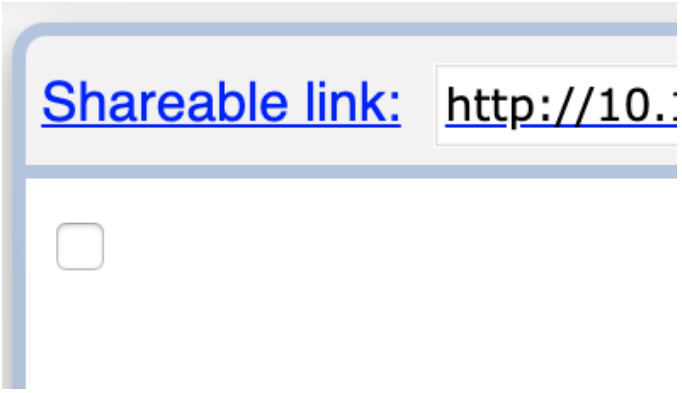
In the Design Grid it looks like this:



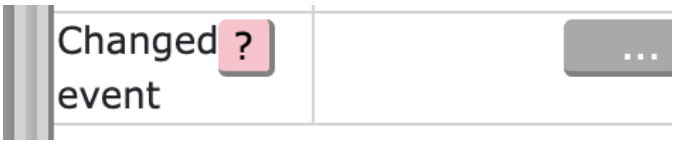
You can set it to **unchecked** by setting a property:



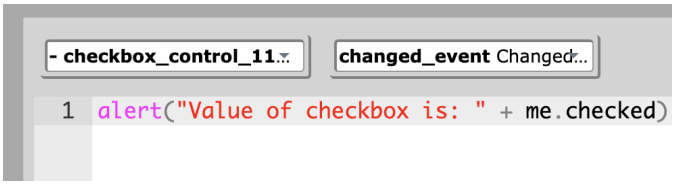
: which will look like this in an application :



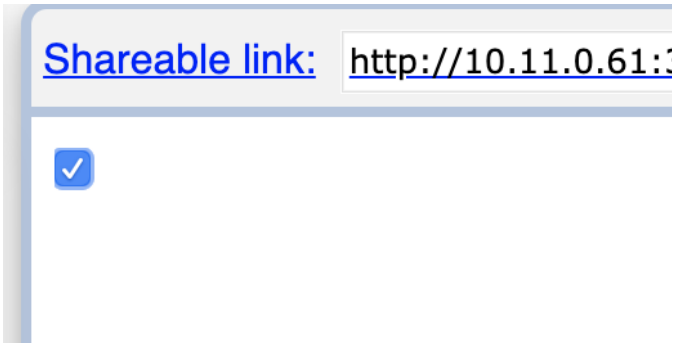
You can see the value of the Checkbox component as it changes by adding a change listener to the checkbox:



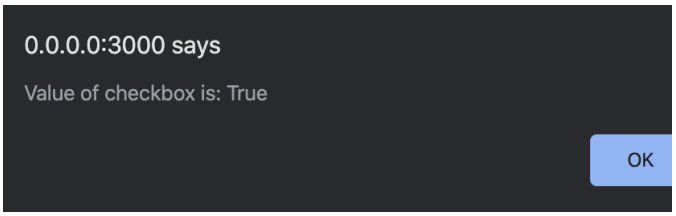
Click the 3 dots to edit the event, then enter:



Now whenever you change the value in the checkbox you will see a popup with the new value:



Try it:



Well done! You now have a checkbox that you can use in your applications.

Conclusion

We have shown you some of the components that are available in Yazz Pilot, but this is just a few of them. We will write more about the other components in an upcoming article.

Getting Started with Yazz Pilot

Step 1

Installation options for Yazz Pilot

There are many ways to install Yazz Pilot:

- Directly from source with NodeJS and Git
- Using Docker
- Using Ubuntu Snap
- Using Kubernetes
- Using Red Hat OpenShift deployment templates

We think that Docker will be the simplest choice for this tutorial.

Step 2

Choosing Docker

Since we will use Docker to install Yazz we should explain why. So, what is Docker?

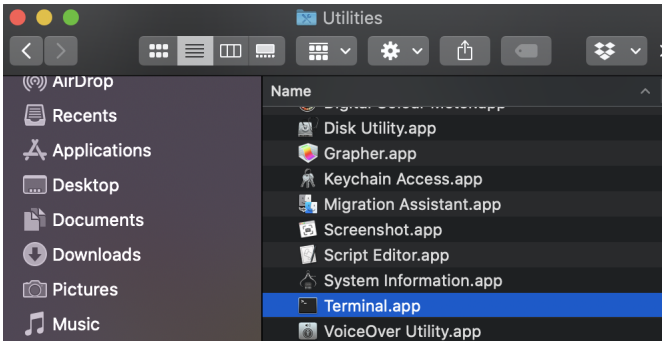
Docker allows you to run applications which are packaged up into what is known as **"containers"**, which are self contained binaries that can package an application and all it's dependencies. All containers are also cross platform and will run on Windows, Mac, and Linux.
This means that we have simpler deployments and application development as we avoid the **"It works on my machine!"** types of development problems.

Step 3

Using the command line

To use Docker you will also need to be familiar with the command line. Most operating systems will give you access to the command line. On the Mac the command line should look something like this:

To start the command line on Mac which is known as **"Terminal"** you need to go to the **/Applications/Utilities** directory:

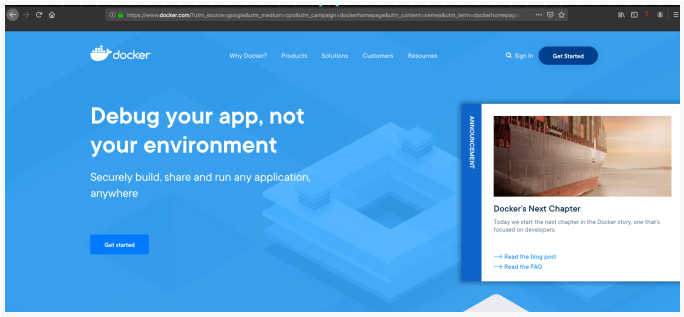


For Linux users, we assume that you are already familiar with the command line, but for Windows users we recommend the **Dos** prompt or **Powershell**.

Step 4

Installing Docker

To install Docker go to **docker.com** for instructions:



Once you have installed Docker then you can confirm that it works by entering **"docker"** at the command line, something like this:

```
$ docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/User
  -c, --context string     Name of the context to use to connect to the da
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"e
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/U
      --tlscert string     Path to TLS certificate file (default "/Users/f
      --tlskey string      Path to TLS key file (default "/Users/fquraish/
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit
```

We recommend that you play around with Docker to understand the basic commands and concepts first.

Note that Docker only works on **Windows 10 Professional** as **Windows 10 Home edition** does not support **Hyper-V**, which is needed to run Docker.

Step 5

Installing Yazz Pilot

Run Yazz using Docker by running the following at the command line:

```
docker run -p 80:3000 -d zubairq/pilot
```

Once we have started Yazz it should output something like:

```
Yazz Pilot started on:

http://0.0.0.0:3000
```

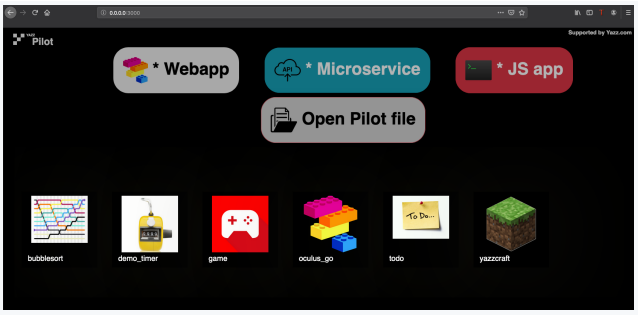
Then, open a web browser and go to the URL:

```
0.0.0.0:3000
```

What is 0.0.0.0?

0.0.0.0 means "all IP addresses on the local machine"

You should then see something like:

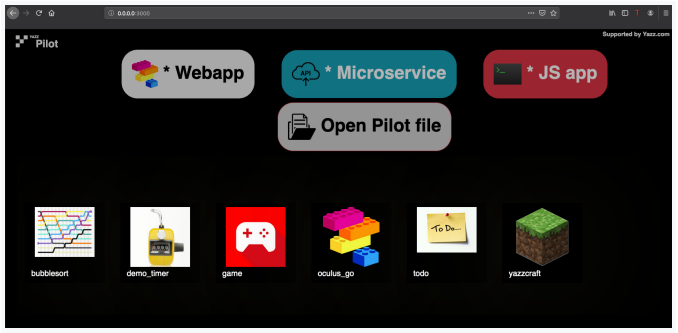


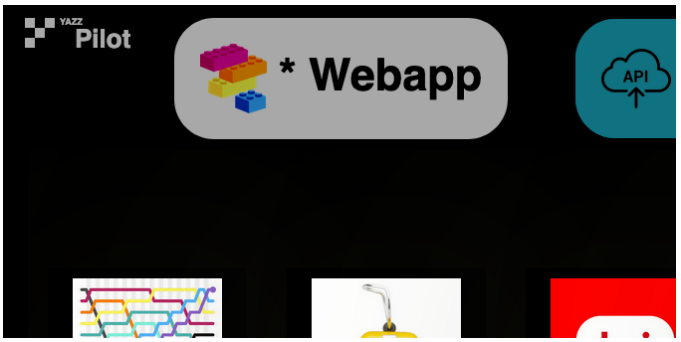
: which is the home page for Yazz Pilot.

Step 6

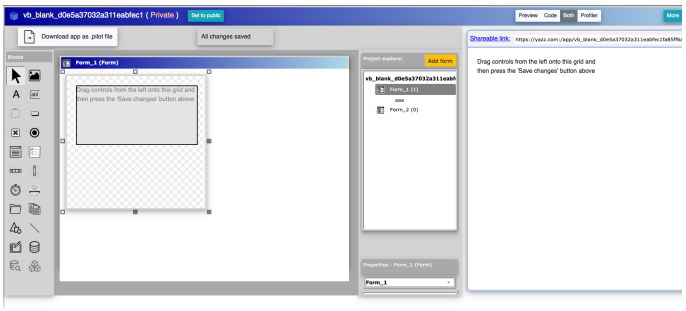
Building your first app

From the home page click the **"Webapp"** button loacted at the top left of the screen:





Once you click on it then you should see the Yazz Pilot UI editor, which should look something like this:

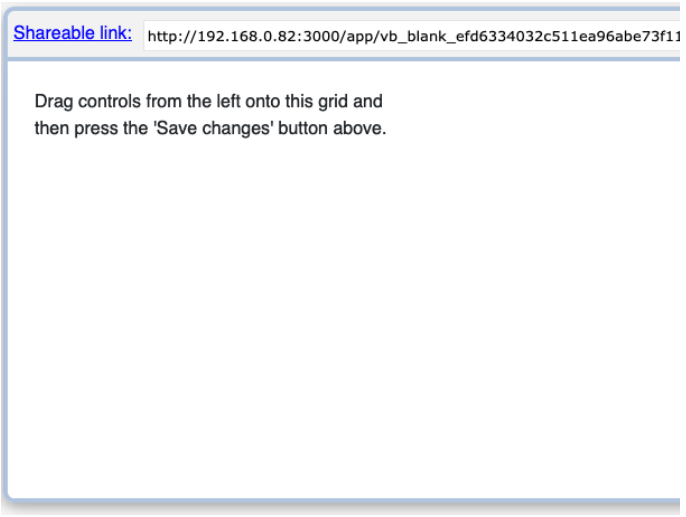


So, now you have actually created a new app from the default template. You haven't actually customised anything yet, but it is still a Yazz Pilot app!

Step 7

Sharing your first app

You can see a preview of the app that you have just created by looking to the right hand side of the editor:



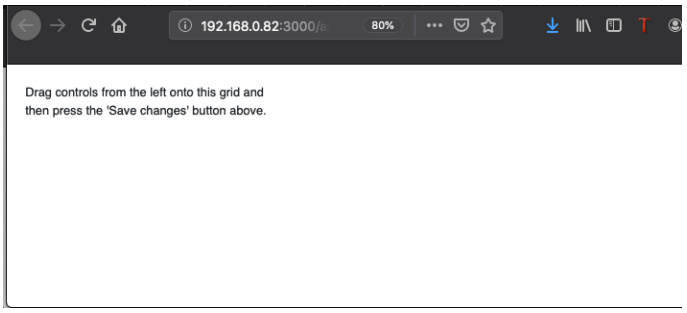
This is the app that your end users will see. At the top notice that there is a URL bar. This shows the URL link that you users will use to access your app.

This URL can be shared with your colleagues so that they can access any apps that you build

Click on **"Shareable Link:"** and you should see your app open in a new browser tab.

Try it!

Clicking on **"Shareable link:"** should bring up:

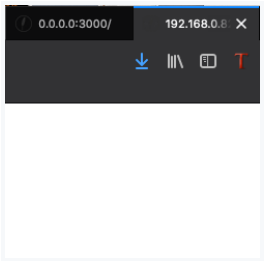


Great, you just opened your first app in a web browser!

Step 8

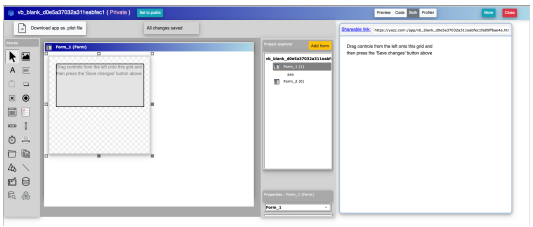
Switching back to the editor

At the top of your browser window you should now have two tabs open, which may look something like this:



One tab is for viewing the app you made, and the other tab is the UI editor.

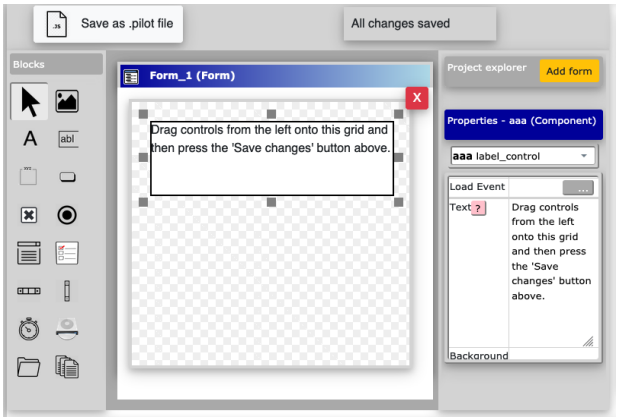
If you press the small **"X"** on the right hand tab then the web application you opened should be closed, and you should be taken back to the main editor:



Step 9

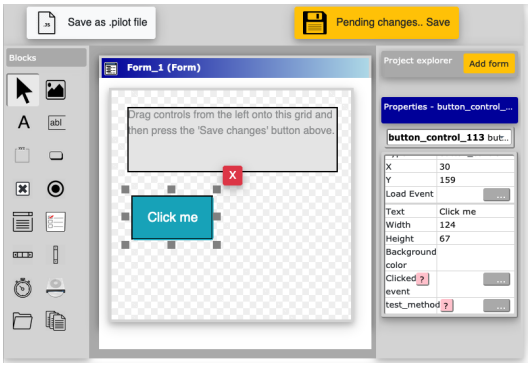
Customizing the app

On the left of the screen you will now see the main UI editor:



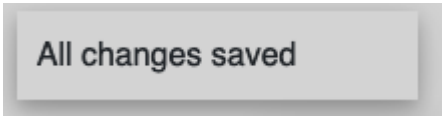
On the left is a list of components that can be added to your app. Add a Button component by moving the mouse to the Button icon:

: and dragging the Button icon to the Design Grid:

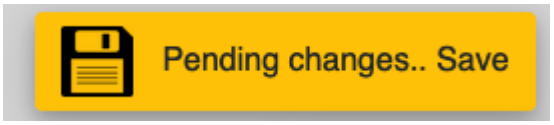


: and you will see the Button placed on the Design Grid.

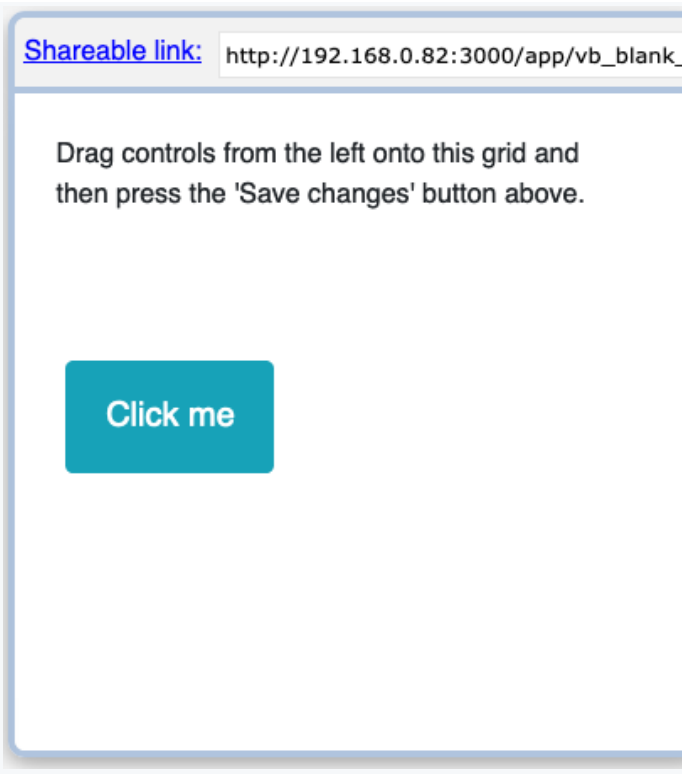
Notice that after you have added the Button to the Design Grid that the button above the Grid Editor changes from:



: to :



Click the **"Pending changes... Save"** button, and you will see that the Application Preview updates to reflect the changes:

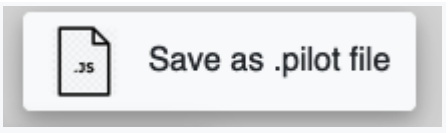


Congratulations, you have now customized your first Yazz Pilot app!

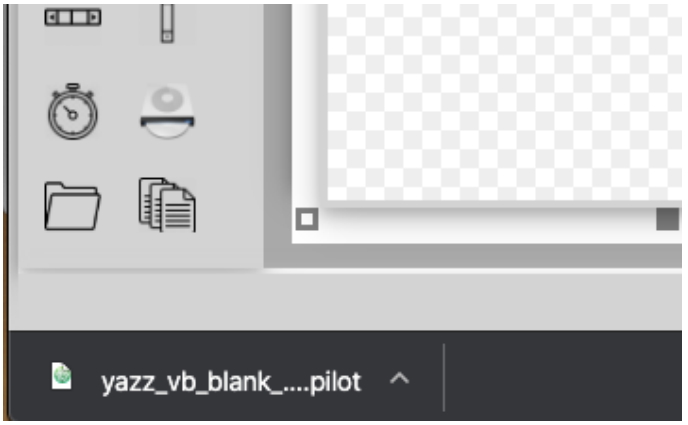
Step 10

Saving the app to your Computer

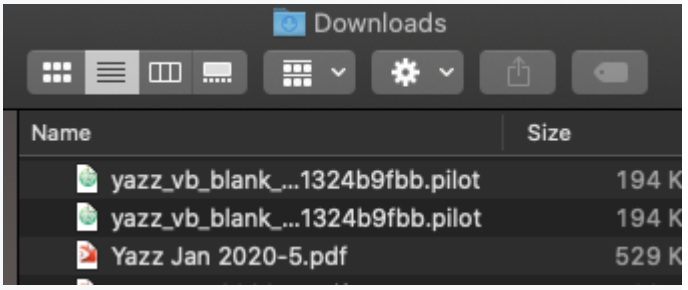
If you want to be able to save the Application to your computer then click on the **"Save as .pilot file"** button at the top of the editor:



You will then see at the bottom of the editor:



: that the file has been saved to your computer. Usually the file will be stored in the **"Downloads"** folder of your machine:



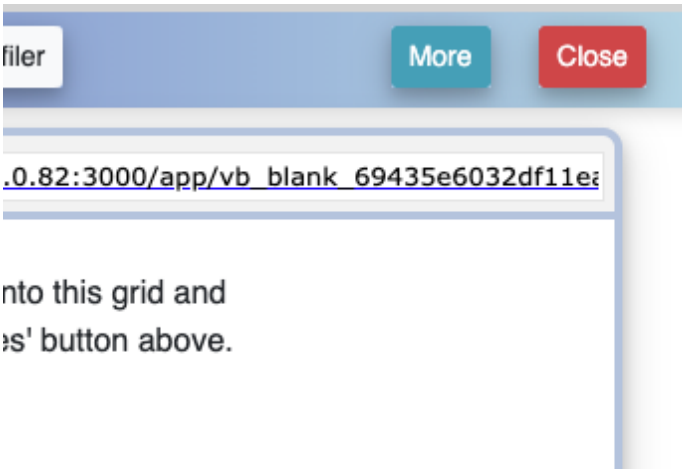
You now have a saved copy of your application.

Note that you can also open the **.pilot** file in your favorite editor as it is just a text file!

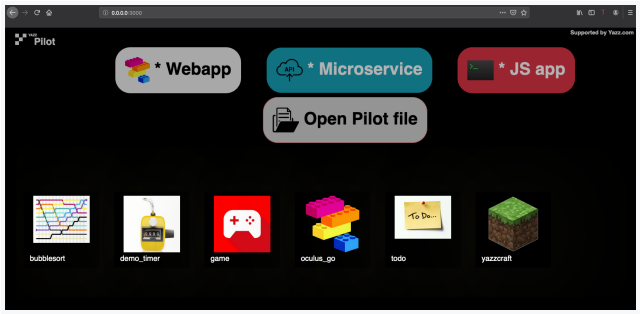
Step 11

Opening a saved application

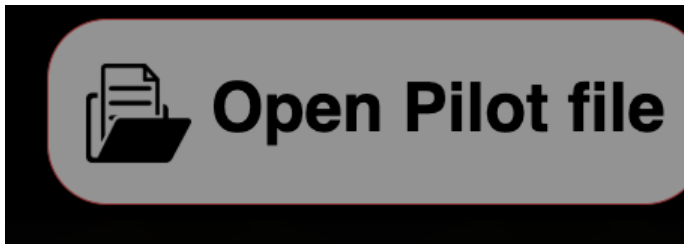
If you wish to open a saved application, press the **"Close"** button to go back to the Yazz Pilot home page:



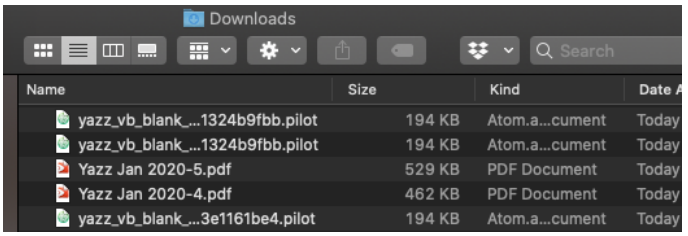
: which takes you back to here:



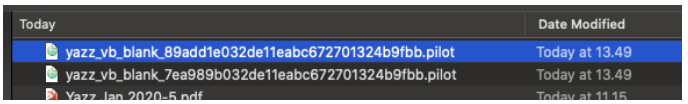
Then, to open the saved application press the button **"Open Pilot File"** to try to open a Yazz Pilot app from the computer's file system:



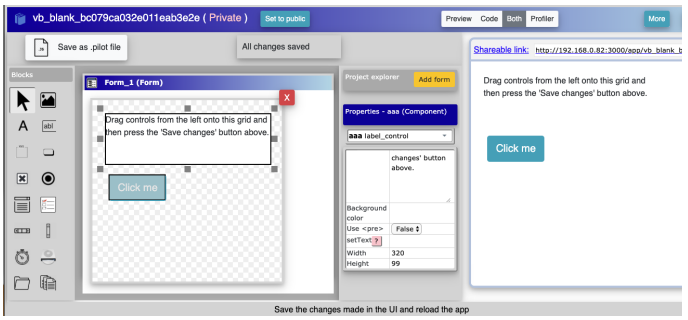
You then get a list of files to choose from:



Choose the app you wish to load (with a file extension of .pilot):



And your application will load:



Congratulations, you have now created your first Yazz Pilot application, customized, saved, and opened it again.

Open Source Automation for the Enterprise

The problem

Employees who work in large enterprises need new IT Solutions to help automate tasks on a daily basis. To get these IT solutions they usually turn to their organisation's local IT Department.

However, because the process takes so long to get any solution from the IT Department, employees usually take matters into their own hands, and end up building a solution themselves using Excel, or other desktop tools. This usually works as a quick fix, but soon things can get messy when their document based solutions starts to be used by other people too.

Previous solutions

This lack of being able to deliver IT solutions fast is not a new problem, and has been written about since the 1970s. Companies have already tried many ways to solve this, with solutions usually falling into one of two main categories:

- Better IT tooling
- Improving IT Methodologies

IT Tooling - Every few years a new batch of tools comes up to solve this. In the 1990s it was Visual Basic, Powerbuilder, Delphi, SQL Windows, Microsoft Access, and a whole bunch of other tools.

However, in time, because of technology and platform changes these tools always got superseded, and the world reverts back to Excel and document based tools.

The same thing happened in the 2010s, and tools like Zapier, Airtable, IFTTT, Retool, Notion, Coda and many others repeated the cycle, trying to use more specialised tools to allow self service IT. Companies also decided to build alot of things using Java, Javascript, Python, and other low level programming tools, but this was very costly to build and maintain.

Methodologies - Every few years a new batch of methodologies comes up. In the past it was Waterfall, Agile, Kanban, ITIL, and PRINCE amongst others

Today, a process known as DevOps seems to have taken hold, and a whole industry has spawned up around DevOps methodologies

The current crisis

Since companies have been aware of the issues with delivering IT systems for so long, then why is this such a big deal now?

Well, over time companies accumulate many IT systems, and if you take any medium to large scale organisation and look at the hundreds of IT systems that they have, you will see many complex links and dependencies. This means that it is impossible to make changes to IT systems rapidly to keep up with a rapidly changing market and requirements.

Companies that fix this complexity issue will survive and thrive, but companies that don't risk ending up getting bogged down by the huge cost and complexity of IT.

Too many choices

So we have a crisis, but we also have many solutions, so there is hope, right?

Well, right now in the 2020s we are at the point where the CIO or CTO of organisation has so many choices about which IT tools and methodologies to choose from that they have another problem of too much choice!

This is a problem because if a CIO/CTO bets on the wrong tool or methodology then their business will suffer in 2 - 5 years time, when the tool they choose gets bought by another company, puts up a paywall, shuts down, or when a more mature platform comes along rendering their IT systems obsolete.

This is where firms like Gartner or IDC come in to try to provide a roadmap to CIOs, CTOs, and business leaders.

What does the future hold?

Here at Yazz we think that the Gartners and IDCs will be even more relevant in the future, since there will be even more tools within Enterprises.

So Yazz's position is not to try to create more tools, but to be "Glue", that helps people to build systems by connecting the rest of the IT tools together, tools like Salesforce, Oracle, Slack, Excel, Word, Outlook, and databases.

We also believe that the industry was in this same situation before, in the 1990s, where there was too much choice of IT tools, and the market could not sustain them all, except for a few tools which connected the other systems together, like Visual Basic, Excel, and Word, which all came to dominate the landscape. With the exception of Visual Basic, those same tools from the 1990s still dominate today, nearly 30 years later!

What are the options?

As described earlier the technology options are:

- Continue using low level languages like Java, Javascript, C, C++
- Use one of the many commercial offerings such as Outsystems, Mendix, Airtable
- Use an Open Source offering which protects a company from some forms of lock-in commercial risk

As for methodologies the options are:

- Waterfall
- Continue having IT as a separate department
- Have a DevOps view of the business which means that IT is just another business unit, on par with marketing or sales

The Yazz solution

We talked earlier about two categories of solutions, technology solutions and methodology solutions.

Yazz is a technology solution, and wants to fill the void left by Visual Basic, and allow systems to be glued together, by using modern web based, containerised and open source technology that will work onsite AND in the cloud.

The Yazz solution is also document based so it can be used from one person's PC, all the way to a multi-user setup using servers.

Conclusion

As a business you have many options. We have tried to outline some of the most popular solutions, and what the Yazz Pilot solution to the problem is.

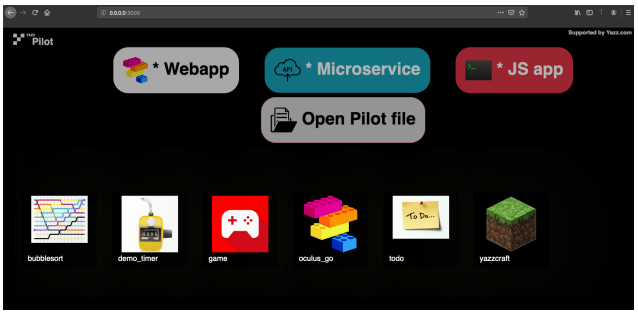
Of course, you must always realise that there is no one size fits all solution, and try to research the market thoroughly before you choose any solution.

Writing code in Yazz Pilot

Step 1

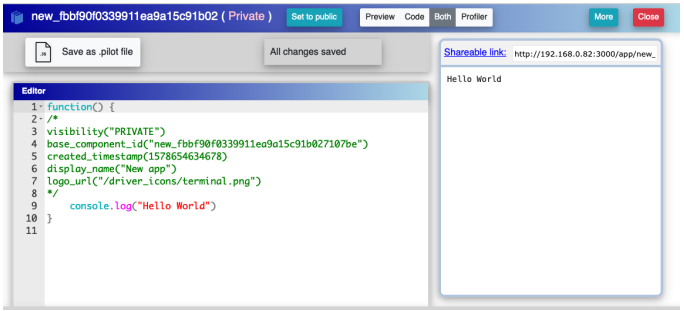
Writing Hello World

In this tutorial we will show you how to make a simple text based app in Yazz Pilot. First go to the Yazz home page



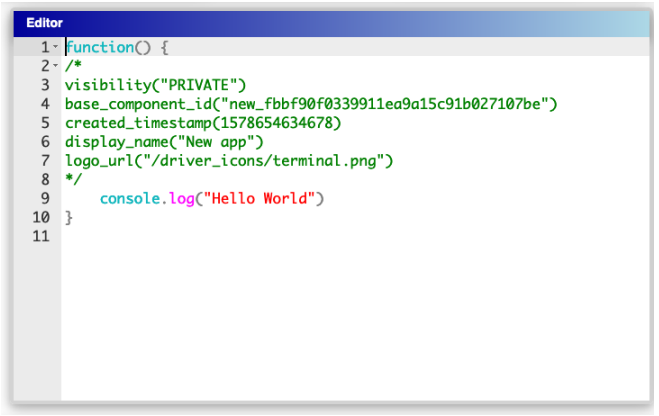
Then click on the button **"JS app"**:

This should bring up code editor as shown below:



On the right there will be an Application Preview which will show the app that was just created

And on the left you can see the code that was used to create this application:



The following code is where the logic for outputting **"Hello World"** was performed:

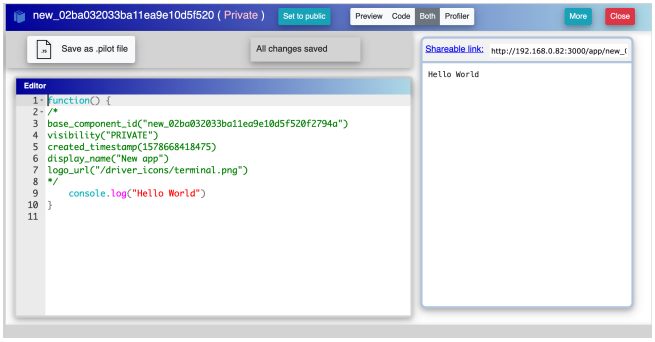
```
console.log("Hello World")
```

That's it. You don't need to do anything else. You have just created your first Yazz Pilot application!

Step 2

Understanding Hello World

Let's try to understand how the Hello World app was built



For the Hello World app the only code that was relevant was the following line:

```
console.log("Hello World")
```

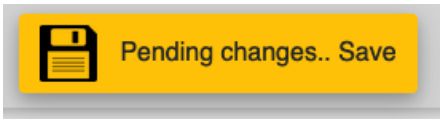
This line caused the Application Preview to show the text **"Hello World"**, like this:

Now, let's change the application. Change the code from **"Hello World"** to **"Hello Yazz":**

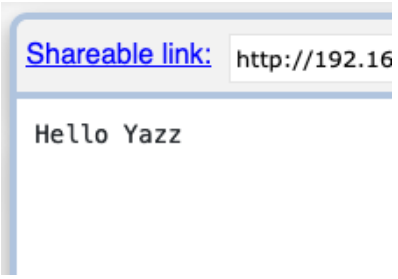
```
display_name("New app")
logo_url("/driver_icons/terminal.png")
*/
    console.log("Hello Yazz")
}
```

When you change the code you will see that the button which says **"All changes Saved"** changes to **"Pending changes... Save"**.

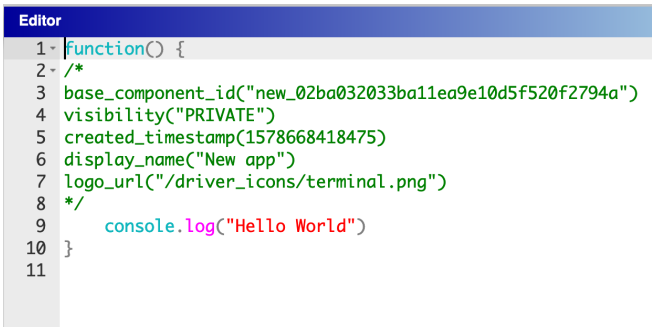
Then press the **Save** button:



Once pressed, the app will be updated, and the Application Preview will change to reflect the updates:



Let's go back to code now. Apart from the code to output a message, you may have noticed a lot of weird green text. What was that?



Let's take a close look

```
/*
base_component_id("new_02ba032033ba11ea9e10d5f520f2794a")
visibility("PRIVATE")
created_timestamp(1578668418475)
display_name("New app")
logo_url("/driver_icons/terminal.png")
*/
```

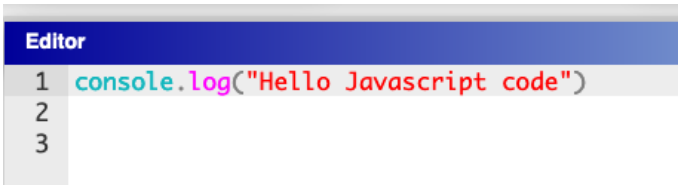
The text in green is code that was added by the Yazz Pilot system as information about the code:

- **base_component_id** - The ID of this component
- **visibility** - Who can view this code
- **created_timestamp** - When this code was created
- **display_name** - Display name in the Application Catalog
- **logo_url** - The app icon to show in the app catalog

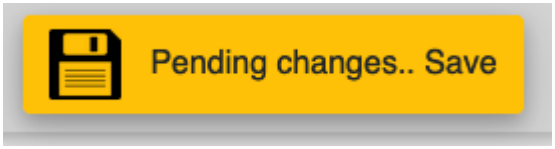
Above the green text was also the following line:

This is a Javascript function definition, as every app in Yazz is represented as a Javascript function.

We can demonstrate how all Yazz apps are Javascript functions. Delete all the text in the editor and replace it with the following line:



Then press **Save**



When you press **Save** you will see that the code is updated with an enclosing function statement and some green code inserted by Yazz:

The App Preview also changes to show that a new app has been created from that one line of code!

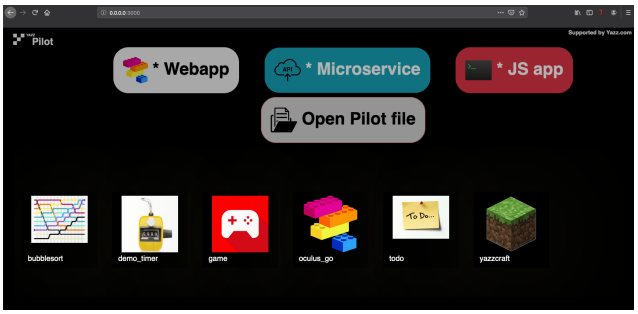
This is just the basics of writing an app in Yazz Pilot, and is the simplest way to get started by writing just a few lines of Javascript!

Step 3

Using the debugger

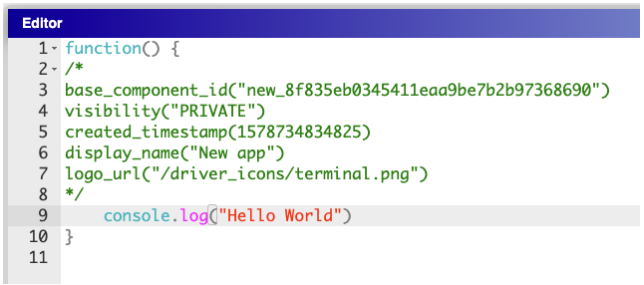
Often when making an app you will run into problems, and will need to see what the program is doing when it runs. You can use the built in debugger to step through your program as it runs.

Go to the Yazz Pilot home page:



Then click **"JS App"** to make a new text app:

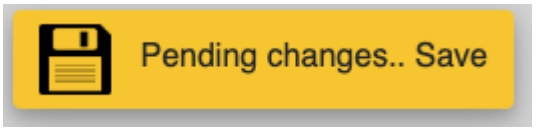
This will create a new text application. Here is the code in the editor:



And here is the App Preview:

In the editor add some code to the app to print some numbers and the word **"Done":**

Then press **Save:**

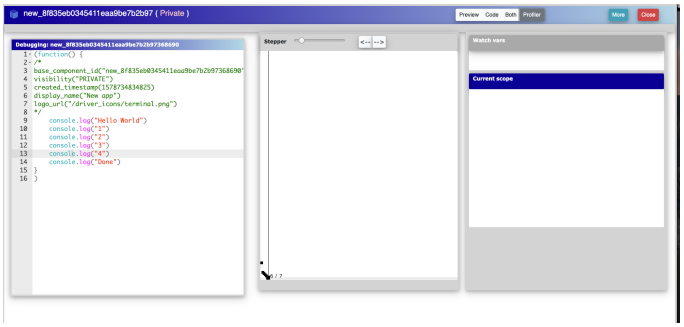


You will see that the App Preview updates to reflect the changes:

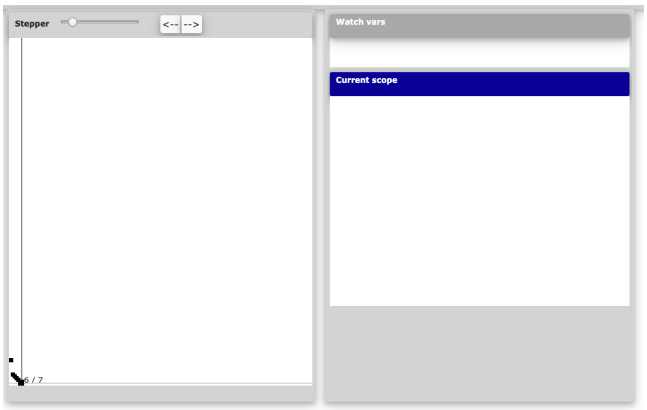
Next, we want to step through the program in the debugger. Above the App Preview there is a menu:

Click on **"Profiler"** in the menu:

And you should be taken to the debugger view:



In the middle you should see something like this, quite empty right now!

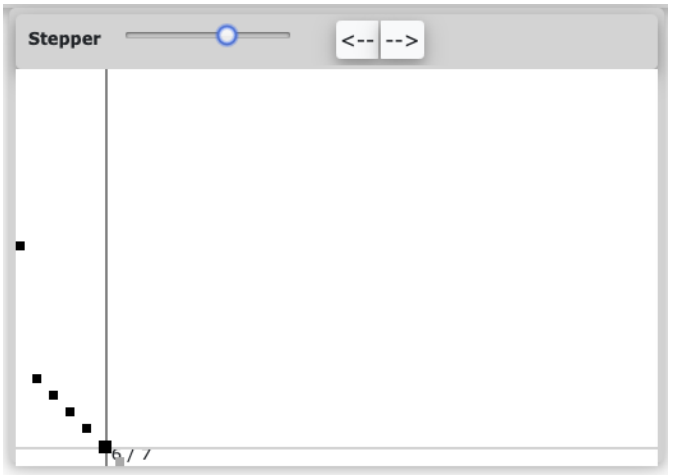


Move the cursor over the blocks:

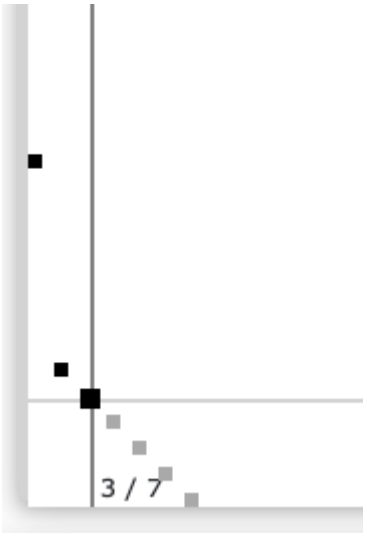
Drag the **"Stepper"** to the right to zoom in:

Until you get about half way:

And you will see that the window below is zoomed. This shows the code steps as the program is run:



If you look at the code windows now you will see that the currently executed line is highlighted



: and you will see that the code lines are highlighted:

Well done, you have learnt the basics of the Yazz debugger!

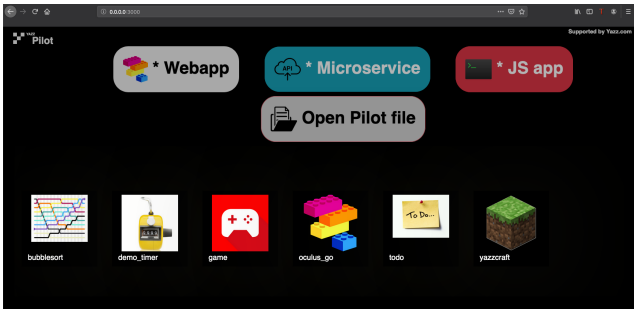
The debugger allows you to see which lines of code are executed so that you can track down problems and see what your Yazz apps are doing internally.

Step 4

Writing events

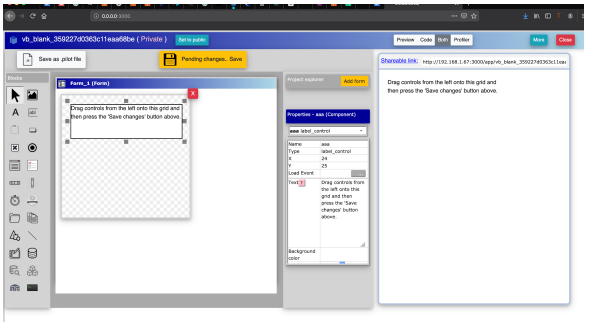
When writing web based apps in Yazz Pilot we showed earlier how you can drag and drop components onto the Design Grid to visually build apps. However, when you want to add interactivity to your apps then you will want to respond to things such as Button clicks so that your can perform some business logic or action. We will now show you how to write code to respond to these events.

First go to the Yazz home page:

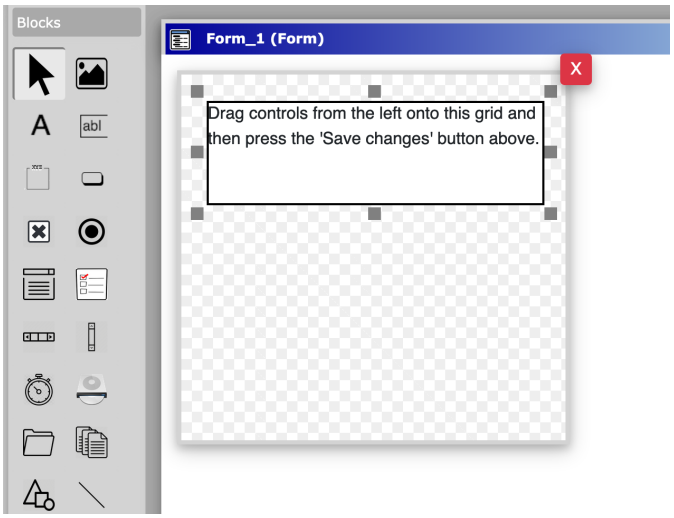


Then press the **"Webapp"** button to make a new webapp:

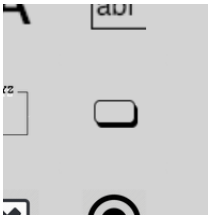
You should now see the UI editor



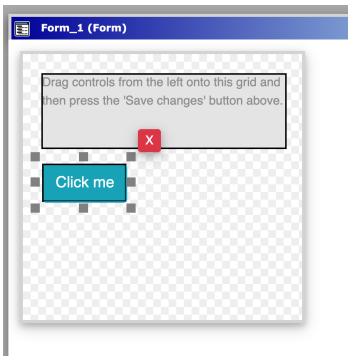
In the editor, go to the list of components on the left of the Design Grid:



On the left you will see components that can be added to the design grid. You will see the **Button** component:

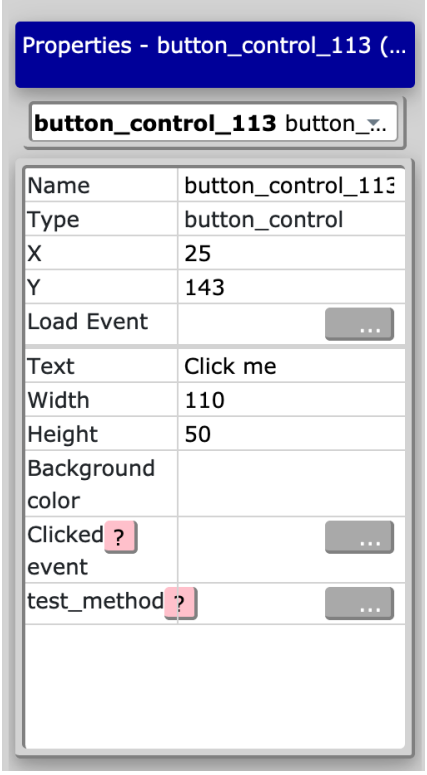


You can add the **Button** component to the design grid by clicking on the Button and then dragging and dropping it to the **Design Grid**:

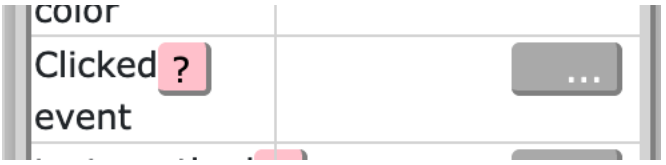


Once you have dragged the Button to the Design Grid you will see it displayed.

Next, click on the Button in the **Design grid** to select it, and then examine the Button properties on the right hand side of the screen:



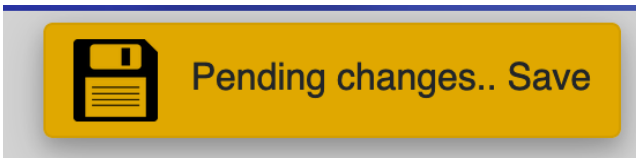
Find **"Clicked event"** and then click the 3 dots on the right hand side:



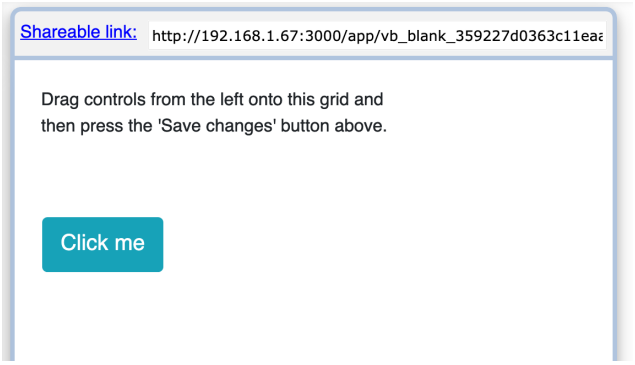
This will bring up the code editor, where you can define what will happen when the button is clicked:

We will just add one line of Javascript to show an alert. Enter the following code:

And then save the changes by pressing **Save**:

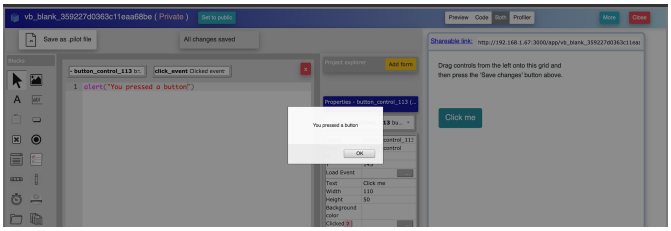


The **App Preview** window on the right should change and you should see something like this:

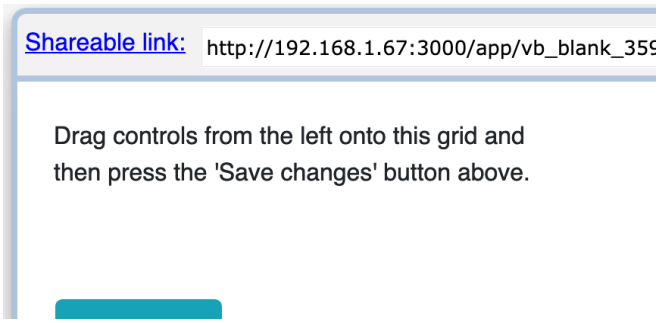


Press the **"Click Me"** button in the App Preview:

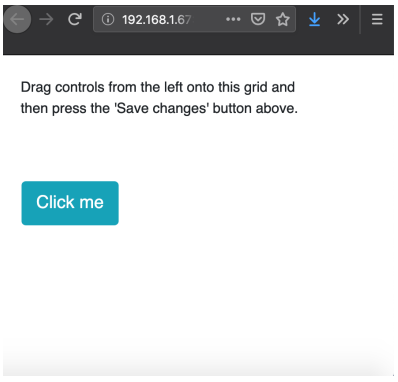
And you should see a message box pop up:



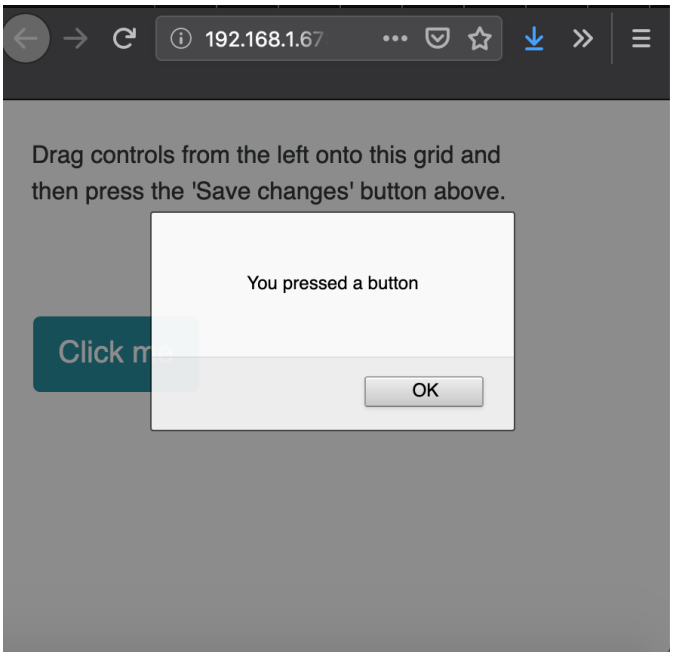
You don't have to use the App Preview. You can also see how your application will loo to end users. Click **"Shareable Link"** from just above the App Preview:



The App should now load in a new browser tab:



Press the **"Click me"** button again:



And the message **"You pressed a button"** should pop up.

Well done! You have now created your first action in response to an event!

The architecture of Yazz Pilot

Introduction

As you know by now Yazz Pilot is a tool for building internal web applications using reusable Javascript components.

However, it would be useful to know how it works. Yazz Pilot may look very simple at first glance, since apps can be as simple as:

```
console.log("A simple text based app");
```

But don't let this simplicity fool you, since there is a big vision, and a lot of thought has gone into how Yazz has been built.

Vision

The vision for Yazz Pilot is:

- ❏ **Short term** - to provide an interactive tool to build internal applications
- ❏ **Medium term** - to build an app store so that people can collaborate over the internet and build components and applications which can be reused by others. This should provide a way for companies to buy and sell components
- ❏ **Long term -** to have Yazz Pilot systems use AI to provide embedded and autonomous systems where the system can learn and make human guided decisions

Current Features

Yazz Pilot has the following features:

- ❏ Can build locally hosted webapps
- ❏ Can run in Docker or Kubernetes
- ❏ Can run as a Snap Package
- ❏ Can run under NodeJS
- ❏ Badly behaving apps which crash are detected and restarted
- ❏ Can work offline from the internet
- ❏ App code written in Javascript
- ❏ All apps can be emailed as self contained HTML files (including database apps)
- ❏ Components are reusable
- ❏ All apps have a built in SQL Database

Specification

Yazz Pilot has the following specifications:

- ❏ Made using Javascript
- ❏ 6 NodeJS worker threads
- ❏ Uses IPC communication between NodeJS threads
- ❏ Uses Sockets for web browser to server communication
- ❏ Uses cross platform NPM packages only
- ❏ 6 worker threads run all server side code
- ❏ Uses VueJS for UI
- ❏ Uses AFrame for VR / AR
- ❏ Sqlite embedded database used for runtime data

License and code

The main GitHub repo can be found at:

https://github.com/zubairq/pilot

: and the license is:

MIT license

Usage of the MIT license means that the code can be copied or redistributed without attribution

Architecture

Yazz consists of a main NodeJS webapp written using the Express framework. This express app talks to a scheduler, a child process, and 6 worker nodes which run app code. Internally all state is stored using a built in Sqlite database.

Scheduler

Whenever a request comes in to run some code the request goes through the scheduler. The scheduler checks the worker process pool and decides which NodeJS child process will execute the code.

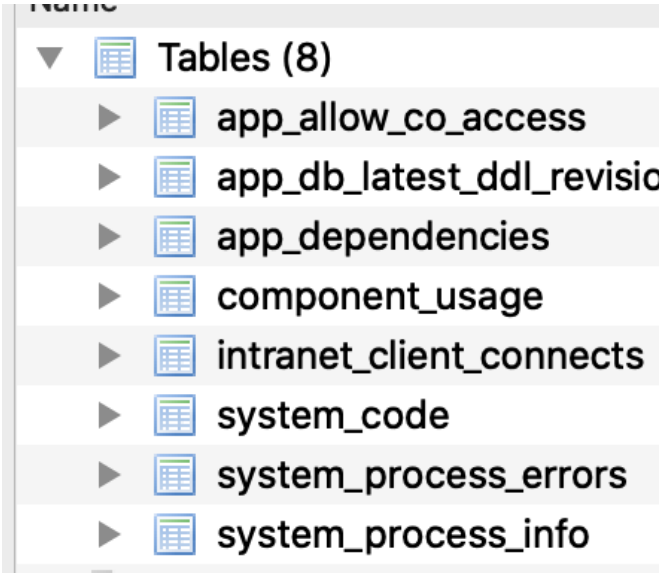
If a process is killed or crashes for some reason then the scheduler restarts it.

Worker nodes

All worker nodes are child NodeJS processes. Each worker node processes the code for one action at a time. Once an action has completed processing then a worker node can process another action.

Sqlite

Sqlite is used to store all internal state for Yazz Pilot. It consists of the following tables:



app_allow_co_access - Unused. The idea is related to collaborative editing

app_db_latest_ddl_revisions - Ruby on rails style database migrations list for Yazz apps

app_dependencies - Describes JS library dependencies do different apps have

component_usage - Shows which components depend on each other

intranet_client_connects - Unused. This is a relic from when Yazz was called VisiFile to discover other nodes on the network

system_code - Stores the app code and hashes

system_process_errors - Shows errors in NodeJS processes

system_process_info - Shows the state of the running nodes in Yazz

Components

Yazz Pilot is modular and built the following using component types:

- ❏ Server side
- ❏ UI components
- ❏ Microservices
- ❏ Custom components
- ❏ Editors

Every component follows the same pattern, defined as a **.pilot** file, which is a valid Javascript function. Eg:

```
function( ) {
    console.log("Hello world")
}
```

: is a valid Yazz Pilot component.

Server side components

To create a server side component in Yazz use the **only_run_on_server** tag when defining the function:

```
function (args) {
   /*
   description("REST API Call server side function")
   base_component_id("rest_call_service")
   only_run_on_server(true)
   */

         console.log("Running on the server");
         ...
```

UI apps

To create a UI app in Yazz define the app using VueJS:

```
async function(args) {
   /*
   base_component_id("todo")
   is_app(true)
   display_name("Todo App")
   visibility("PUBLIC")
   description("This will create a demo todo app")
   logo_url("https://imgur.com/logo.jpg")
   */
         Vue.component("todo", {
               template: `<div>
                              Todo List<br>
                              <li v-for='item in
items'>
                                    <button
                                    ...
```

Microservices

Microservices are written in a similar way using the **rest_api** tag:

```
function new_microservice(args) {
   /*
   base_component_id("new_microservice")
   display_name("New microservice")
   rest_api("change_this_url")
   logo_url("/driver_icons/rest.png")
   */
         return {a: 1, b: 2, c: "Hello Pilot!"}
}
```

UI Custom Components

A UI Custom Component in Yazz is a special type of component, which is used only within the visual drag and drop editor. It also contains properties, actions, and event definitions:

```
function(args) {
   /*
   is_app(true)
   component_type("VB")
   display_name("Horizontal scrollbar component")
   description("This will return the horizontal scrollbar")
   base_component_id("horiz_scroll_component")
   properties(
      [
         {
            Id: "text",
            Name: "Text",
            Type: "String"
          }
          ,
          {
             id:  "background_color",
             name:   "Background color",
             type:   "String"
          }
          ...
```

Editor Components

Even the code editors in Yazz are components.

Yazz Pilot currently has a text based editor and a UI drag and drop editor.

Non Functional Requirements

- Must be able to be started from a single docker command at the command line
- Must be able to be accessed in under 0.5 seconds

Text apps

Very simple text apps can be made with Yazz Pilot by opening up the editor and entering:

```
console.log("A simple text based app");
```

This makes it very easy to get started, as you don't need to define any UI or graphical code.

Conclusion

We have described the basics of the Yazz Pilot Architecture.

Test Yourself

Question 1

Yazz Pilot uses Basic as the scripting language

True	*False*

Yazz Pilot uses Basic as the scripting language	
True	*False*
This is incorrect as even though the concept of Yazz Pilot is basic on the Visual Basic programming language, it uses Javascript instead as the scripting language.	This is correct. Yazz pilot uses Javascript, not Basic. The reason for this is that Yazz Pilot runs in the browser, and Javascript is the native scripting language for the browser.

Question 2

Yazz Pilot has a built in copy of which database?

Sqlite	*Postgres*

Yazz Pilot has a built in copy of which database?	
Sqlite	*Postgres*
This is correct as Yazz Pilot uses Sqlite itself as a database and every app built with Yazz has it's own Sqlite database as well, including offline browser based Yazz apps.	This is incorrect, as Postgres would make the installation of Yazz Pilot a lot more complicated. However, a Yazz Pilot app can connect to an external Postgres database via a Postgres connector Component

Question 3

Which platforms does Yazz Pilot run on?

Windows, Mac and Linux only	*Windows, Mac, Linux, Raspberry PI, Docker, Kubernetes, Snap, NodeJS*

Which platforms does Yazz Pilot run on?	
Windows, Mac and Linux only	*Windows, Mac, Linux, Raspberry PI, Docker, Kubernetes, Snap, NodeJS*
This is incorrect as while Yazz Pilot can run on Windows, Mac and Linux, it can also run anywhere that containers or NodeJS can run	This is correct, as Yazz Pilot can almost anywhere via containerised platforms or NodeJS

Question 4

Which system is Yazz Pilot inspired by?

Eclipse CHE	*Visual basic 6*

Which system is Yazz Pilot inspired by?	
Eclipse CHE	*Visual basic 6*
This is incorrect as although Eclipse CHE also is web based and runs in a container, Eclipse CHE is a fully featured editor for low level languages such as Java	This is correct, as Yazz Pilot is based on a Windows app from the 1990s which had a strong market for reusable components

Question 5

Which of the following can be entered as a valid Yazz Pilot program?

console.log("Hello World")	*function( ) {* * Output "hello world"* *}*

Which of the following can be entered as a valid Yazz Pilot program?	
console.log("Hello World")	*function( ) {* * Output "hello world"* *}*

This is correct as this simple one liner in Javascript will be converted to the following by Yazz Pilot:
 	
function( ) {
  console.log("Hello World")
}
 	
: which is a valid Yazz program
	This is incorrect, as Yazz Pilot is uses standard Javascript, so the following line will be flagged as an error:
	
	Output "hello world"

Question 6

How are all Yazz Pilot programs stored?

As files in a filesystem with the extension .pilot	*In many different formats, but always identified by a hash of their content*

How are all Yazz Pilot programs stored?	
As files in a filesystem with the extension .pilot	*In many different formats, but always identified by a hash of their content*
This is incorrect as while Yazz Pilot programs can be saved to disk as a .pilot file, this is just one way to store a program. A Yazz Pilot program stored as a .pilot file still needs to be imported into the Yazz system	This is correct, as Yazz Pilot programs can be stored as files, in a database, or in any other way, and all programs are identified by a hash of their contents

Question 7

What is the ideal use case for Yazz?

For building highly scalable, public facing websites like Facebook or LinkedIn	*For building small glue code webapps in an enterprise intranet environment*

What is the ideal use case for Yazz?	
For building highly scalable, public facing websites like Facebook or LinkedIn	*For building small glue code webapps in an enterprise intranet environment*
This is incorrect as this is probably the worst use case for Yazz. If you wish to create a highly scalable public service then you are best off using custom programming languages and tools like React, VueJS, Java, javascript, NodeJS, Go, and other tools	This is correct, as Yazz Pilot excels in making small webapps which serve a small internal audience within an intranet

Question 8

What are the closest alternatives to Yazz Pilot in the market?

There are no alternatives to Yazz Pilot. Yazz Pilot is the only enterprise integration tool on the market	*Retool, Mendix, Outsystems, among hundreds of other products*

What are the closest alternatives to Yazz Pilot in the market?	
There are no alternatives to Yazz Pilot. Yazz Pilot is the only enterprise integration tool on the market	*Retool, Mendix, Outsystems, among hundreds of other products*
This is incorrect as there are many other choices on the market that you should consider first.	This is correct, as there are many other tools you should consider first.

Question 9

What is unique about Yazz on the market?

The tool is completely open source and free, even for commercial purposes	*Yazz has a visual drag and drop designer*

What are the closest alternatives to Yazz Pilot in the market?	
The tool is completely open source and free, even for commercial purposes	*Yazz has a visual drag and drop designer*
This is correct as Yazz is the only developer tool which is completely open source and free to use for your business	This is incorrect, as there are many other tools which have a good visual designer

Future features

SSO - IBM ISAM, Red Hat SSO, Okta, Auth0, OpenID Connect, Keycloak are all essential SSO services which enterprises rely on for end users to sign in, so must be supported by Yazz Pilot

Databases - Postgres, Oracle, DB2 must all be supported

Red Hat Fuse Integration - It would be very useful to access all the integration features of Fuse/Camel

About us

Omar Quraishi
CEO

Zubair Quraishi
Open Source Developer

Back Issues


JAN 2020
YAZZ
MAGAZINE
YAZZ (Printed edition)
Open Source Automation for the Enterprise
How Yazz Pilot can increase your ROI by allowing business users to build self service applications
Getting ready for the self service enterprise
Companies want business agility, enabled by IT, but WITHOUT having to go through the IT department.