
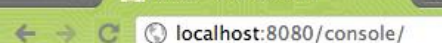
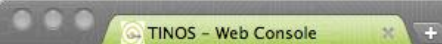



# TINOS WEB CONSOLE (SNAPS-BASED)

Modular Web – here we come....

# TINOS Web Console








## WELCOME TO TINOS WEB CONSOLE


Tinos is a fully functional componentised Internet Protocol stack, Link Layer through to Application Layer, with ancillary support infrastructure. The network stack is implemented as a set of independent software components. A component is typically a well-defined unit in the stack, such as a transport protocol e.g. TCP is a single component. These components are then deployed as an OSGi based application within an application server.


Tinos has its origins as a prototype in the EU FP7 research project ICT 4WARD, built exclusively by the TSSG, a research group of the Waterford Institute of Technology. The source code of Tinos is now being released as a protocol experimentation framework (on GitHub) with a plan to further develop it towards the specific application of virtual networking.




**Pouzin Society**  
Find out more about the community behind Tinos


**Tinos Blog**  
Tinos related blog posts, beware we are only getting warmed up.

**GitHub Repository**  
Have fun, go check out the code

**Introduction Slides**  
Download and Read or Watch and Listen

**Watch a Demo**  
Watch the demo which made us open source TINOS

Powered by Eclipse Virgo Server, Virgo Snaps

Sponsored by 

# Web Applications (The Problem)

- Are typically monolithic
  - Well in Java anyway
  - WAR file – one big lump of gunk (all your website is in here)
- Why do I care
  - TINOS is expanding and modular to the core
  - It is plug-n-play at deployment
    - Cannot bind users to a single web console that has dependencies on everything.
    - This would result in full deployment of all bundles (necessary or not)
    - Constant merging to the Console bundle.
- 3<sup>rd</sup> parties need to add content to the web console
- Web Console needs to be dynamic to match deployments

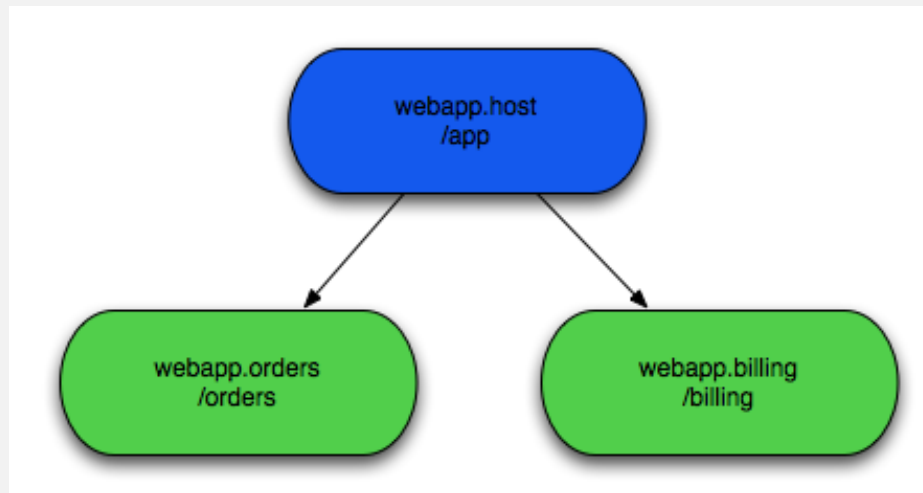
# Solutions

- Even with OSGi Applications, it is typically a single WAR file for the Web Application. The underlying functionality can be modular.
- Option 1 : Web-Fragments
  - tight knit from design & development,
  - Not really a good choice for us at the moment
  - Web-Fragments are not modular at deployment or runtime. Only at development time. The Servlet 3.0 Spec requires all libraries that include a Web-Fragment to be in the lib directory of the War file.
  - Web-Fragments also have non-trivial rules about the resultant composite servlet context, in snaps, each snap is simply dealt with in isolation once the appropriate snap has been found from the requested context path.
- Option 2 : SNAPS
  - Deployment Time, no visibility required of other developers
  - SOA Based.
  - \* We will supply the console.host (root)

See: [https://bugs.eclipse.org/bugs/show\\_bug.cgi?id=329816](https://bugs.eclipse.org/bugs/show_bug.cgi?id=329816)

# From Eclipse Virgo

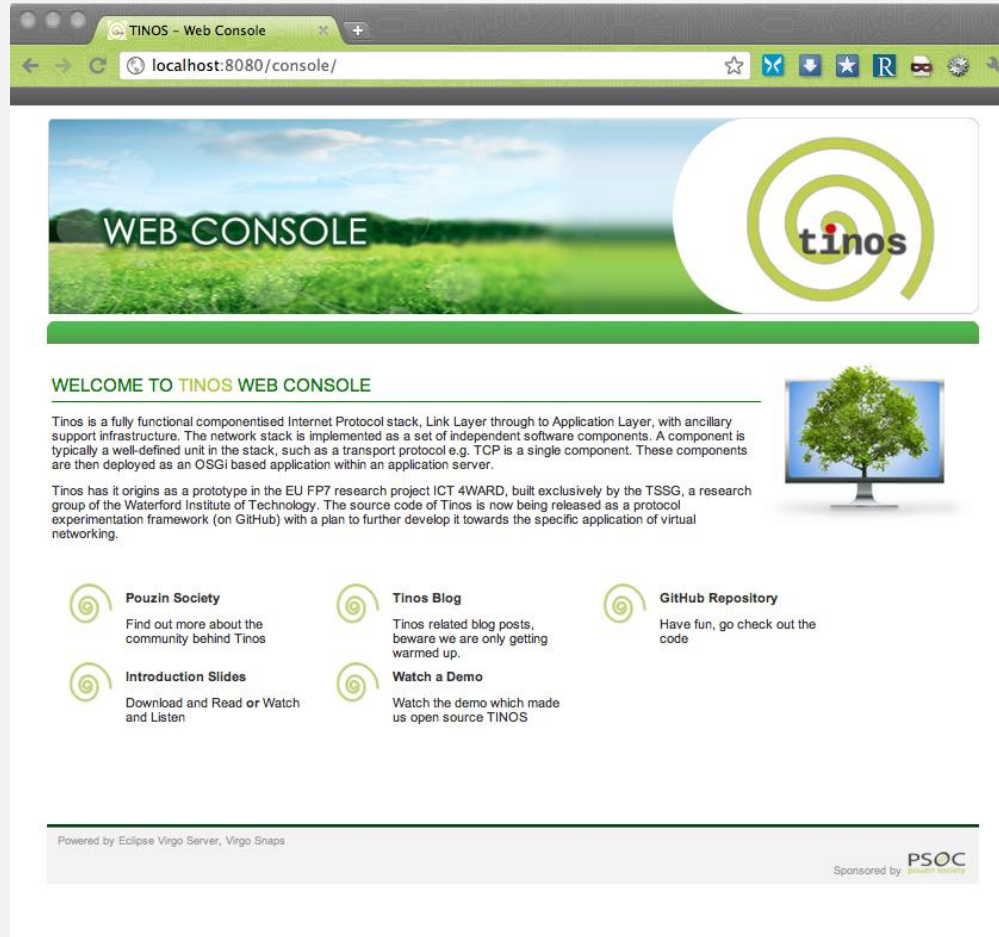
- <http://wiki.eclipse.org/Virgo/Future#Snaps>
- Snaps is a prototype also from dm Server where it was known as Slices. It aims to give you the full benefit of a dynamic and modular OSGi environment for your web applications by allowing you to spread the front end across many bundles, or Snaps. The Snaps source code is available from one of our Git repos listed on the 'Source' tab. Thanks to Patsy Phelan and Dmitry Sklyut for comments on this content.
- A Snap is an OSGi Web Application Bundle that is responsible for handling requests for part of the total application. Requests are served up to each snap depending on what part of the context path the snap has registered to handle. Each Snap will be given requests by a host bundle. A Host bundle is also just a normal web application bundle with filter configured to dispatch certain requests to any registered Snaps.



- The host will handle requests to '/app/\*' while '/app/orders' and '/app/billing' will be dispatched to the relevant Snap bundle.
- Snaps is different from other technologies such as Web Fragments in that it is much more decoupled and takes advantage of OSGi to allow for the dynamic composition of a web app at runtime. The lifecycle of the Snap and the Host is completely decoupled. This is due to the services like approach. Web Fragments are static and much more closely coupled, they share the same classpath and the modularity it offers is only at development time. This extra decoupling means that snaps can be developed in isolation from the host application, even after the host application has been released.

# Example: TINOS Web Console

- No SNAPs present
  - Deployment of org.pouzinsociety.web.console.host (root)



- Nice and pretty, but doesn't do much.
- The plan is below, the SNAP is commented out.
- Lets add that back in and see what happens
  - Using the Samples under TINOS
  - *These are also being donated to Eclipse Virgo Snaps*

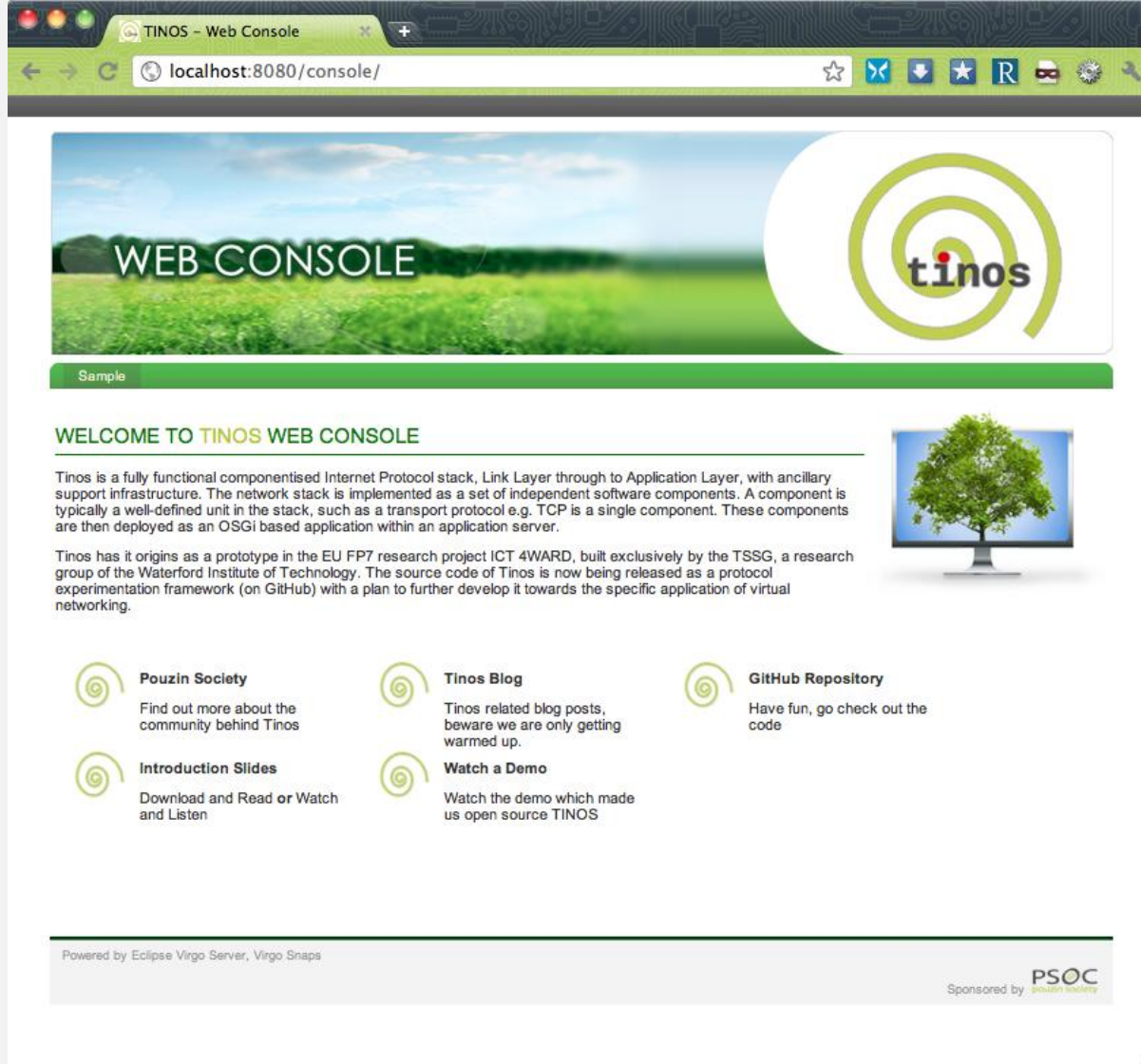
```
<?xml version="1.0" encoding="UTF-8"?>
<plan name="sample.plan" version="1.0.0" scoped="true" atomic="true"
      xmlns="http://www.springsource.org/schema/dm-server/plan"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springsource.org/schema/dm-server/plan
                          http://www.springsource.org/schema/dm-server/plan/springsource-dm-server-plan.xsd">

  <!-- Web Console -->
    <artifact type="bundle" name="org.pouzinsociety.web.console.host" version="[1, 2]">
      <!--
        <property name="header:Web-ContextPath" value="node1"/>
      -->
    </artifact>
    <artifact type="bundle" name="sample.api" version="[1, 2]" />
    <artifact type="bundle" name="sample.impl" version="[1, 2]" />
    <!--
    <artifact type="bundle" name="sample.snap" version="[1, 2]" />
    -->

</plan>
~
```



# With the Sample SNAP



Notice, the new entry on the tab bar

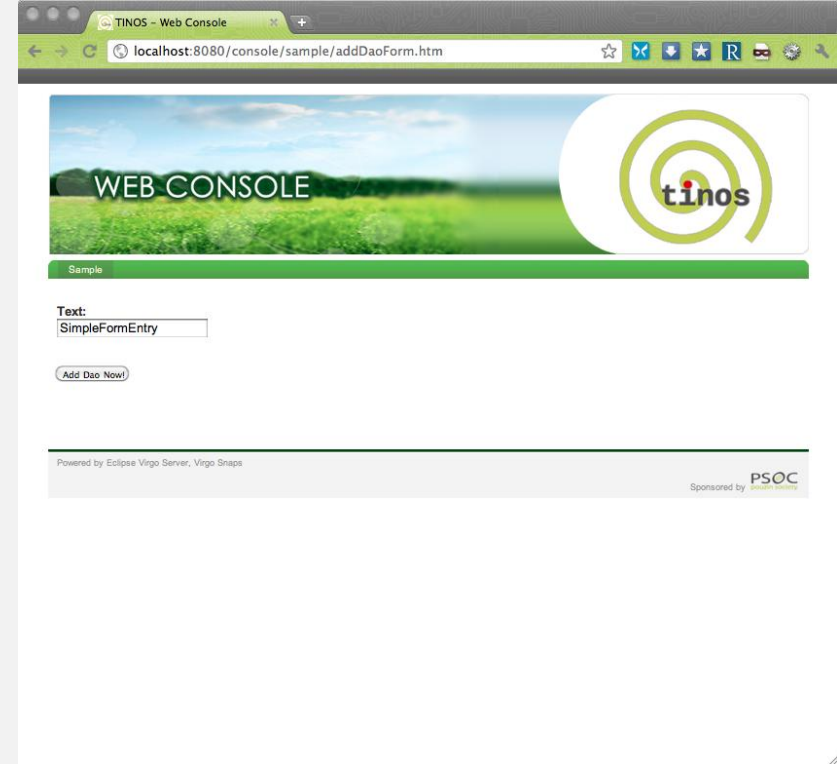
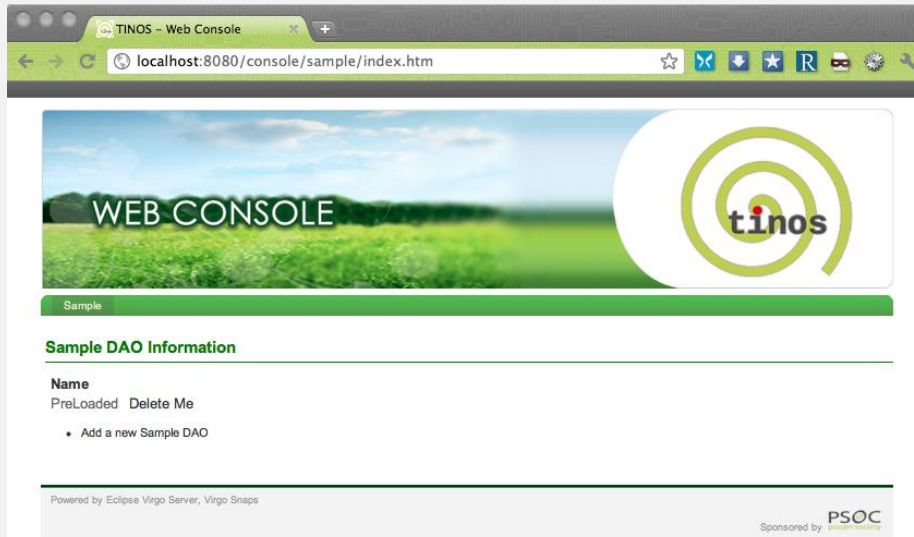
- Sample

- Depending on the number of SNAPs deployed, the tab bar options will fill out.

In this case, /console is the root of the web application.

/console/sample is owned by the Sample SNAP.

# Sample SNAP

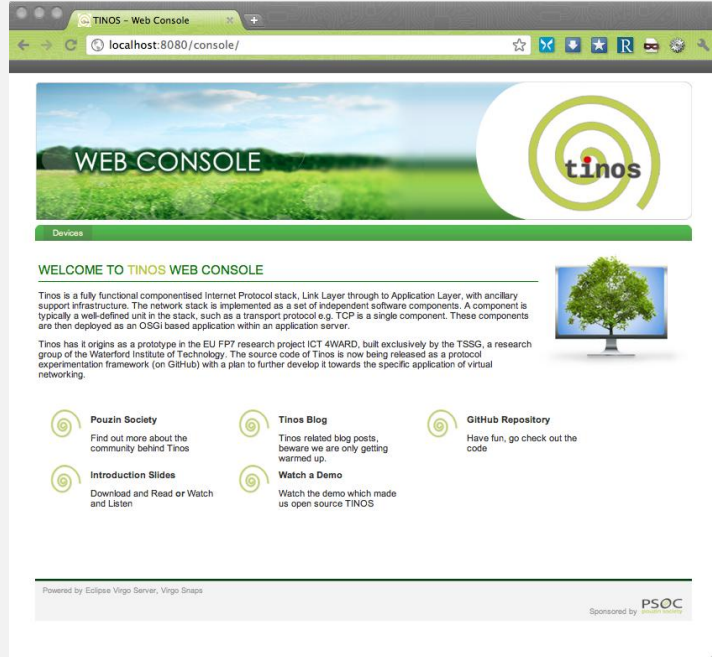


- You can do everything you normally can : Forms, Index pages and so on.
- Styling is all “inherited” from the web.console.host
- You could override this with local styling / images etc. if need be.

## Note:

*Forms combined with <submit> could be directly linked to a method call on an API exposed as an OSGi service by some other bundle....  
Think of manually driving a demo scenario from the web console.....*

# Back to Reality (Devices)



- Devices Snap : [org.pouzinsociety.web.console.devices](http://org.pouzinsociety.web.console.devices)
  - Dependency to DeviceManager
  - Optional to use with Console
  - Get Device Listing with parameters.

# A real “Node” Plan file snippet

```
<!-- Jinos Support -->
  <artifact type="bundle" name="org.pouzinsociety.support.jnode" version="[1, 2]" />
  <artifact type="bundle" name="org.pouzinsociety.driver.net.idrive" version="[1, 2]" />
<!-- Web Console -->
  <artifact type="bundle" name="org.pouzinsociety.web.console.host" version="[1, 2]">
    <!--
      <property name="header:Web-ContextPath" value="node1"/> -->
    </artifact>
  <artifact type="bundle" name="org.pouzinsociety.web.console.devices" version="[1, 2]" />
```

You can add more SNAPs as needed to dynamically create the web console of your choice just like you can dynamically create “nodes” of your choice for deployment.

## DEMOS

Branding can be changed to suit the client by replacing the web.console.host on a demo basis as this styling is inherited by the SNAPs. This means one bundle (which only holds images/css) has to be changed – not everything.

# Now to the Dark Arts...

- How does it work
- Plans
  - You need to deploy Virgo.Snaps
  - You need to then update add SNAPs to your plan and deploy.
- Bundles
  - Mostly contained within the MANIFEST.MF and the SERVLET.XML files of the WAR
  - Minor tweaks in how you handle requests
    - See SAMPLES : sample.snap, sample.api, sample.impl
  - Ask questions on the mailing list