



The framework for developing
sophisticated web applications
in Smalltalk

En este capítulo ...

- Presentar el framework Seaside para construcción de aplicaciones Web en Smalltalk
- Conocer la anatomía de una aplicación Seaside
- Conocer los pasos básicos para construir y publicar una aplicación web simple

HTML en OO2

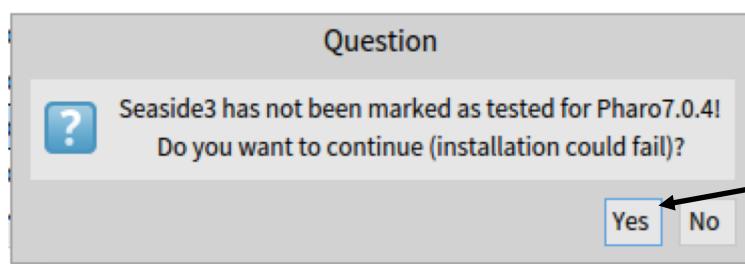
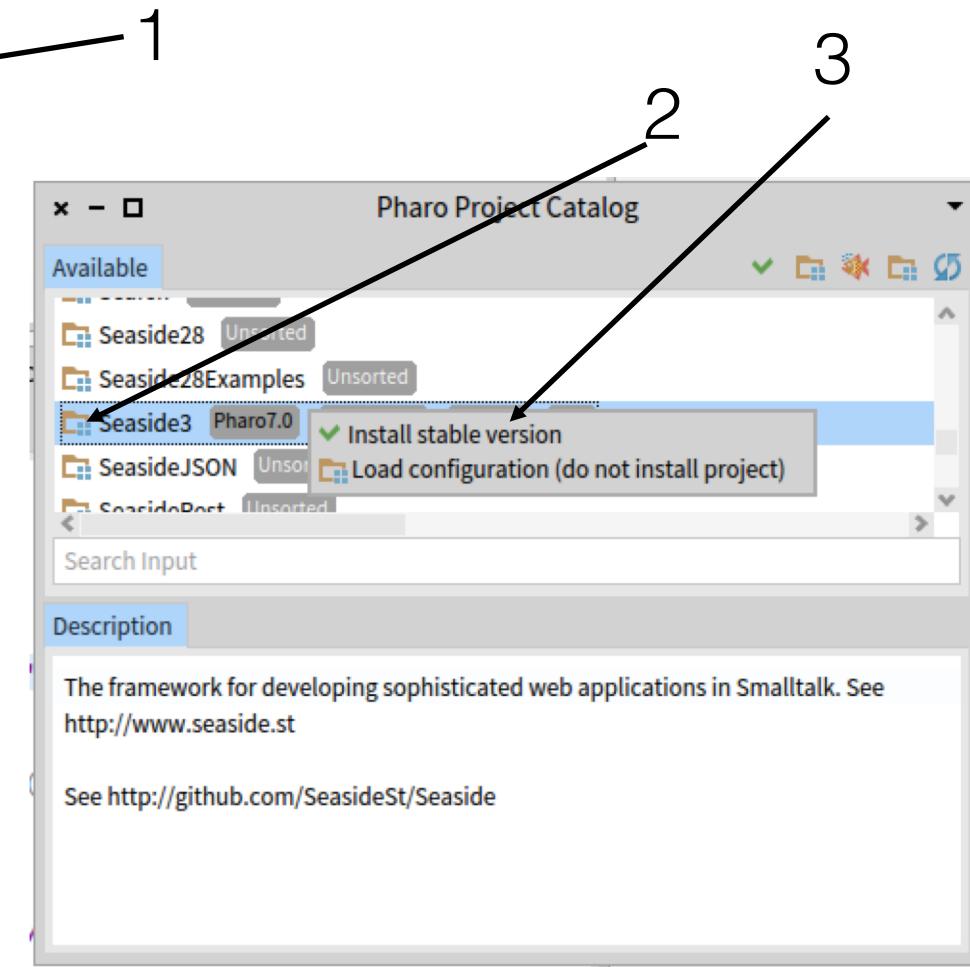
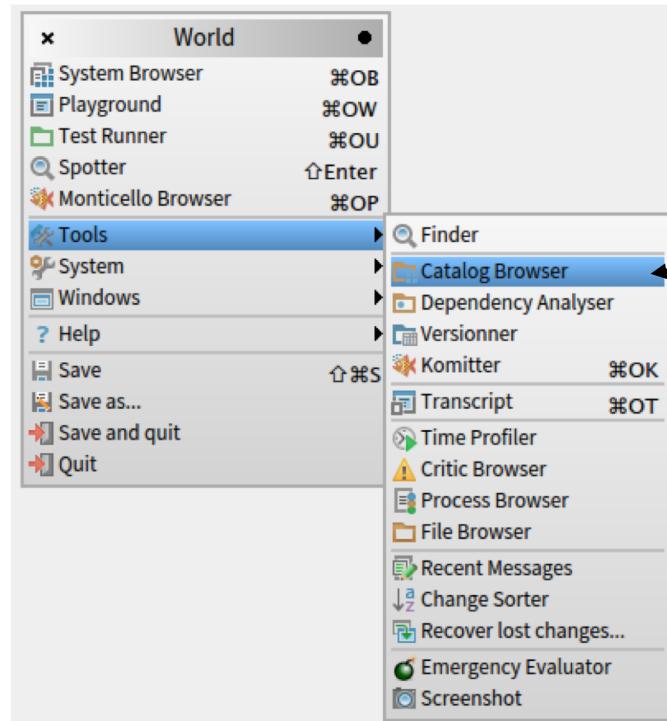
- ¿Qué nos interesa?
 - Cómo generar una página web desde Seaside
 - Cómo reaccionar a las interacciones del usuario con esa página web
 - Hacer todo eso en el marco que nos plantea Seaside
- ¿Qué NO nos interesa?
 - CSS, JS, Single page applications, Ajax ...

Consultar: [HTML en Orientación a Objetos 2](#)

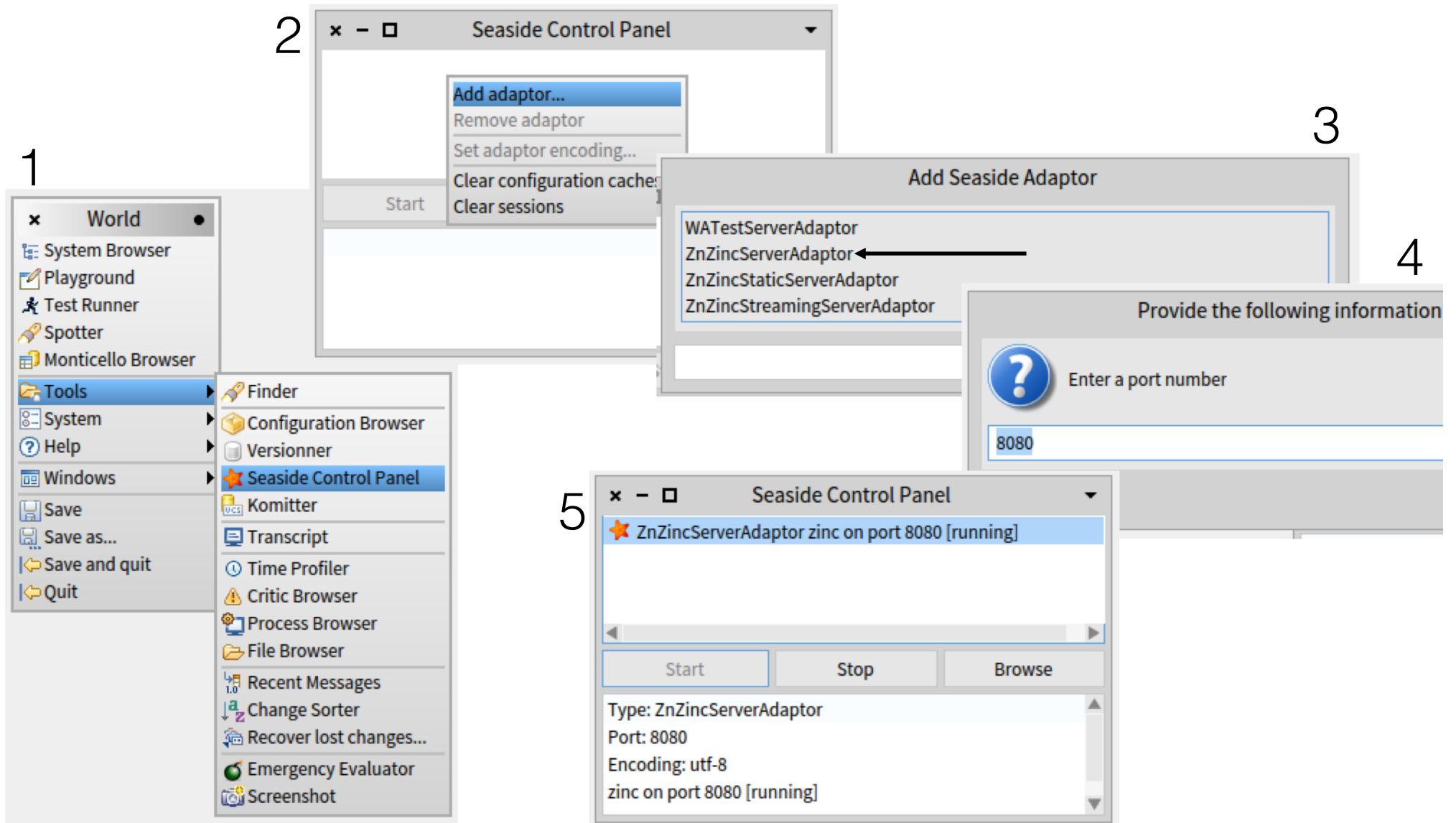


- Framework para construir aplicaciones web en Smalltalk
 - Incluye su propio servidor de aplicaciones
- Las aplicaciones se construyen totalmente en Smalltalk (no se escribe html)
- Una aplicación consiste de un conjunto de **componentes** conectados y reutilizables
 - Conecto componentes para modelar la navegación
 - Compongo componentes para construir otros mas complejos
- El estado de la sesión se mantiene en el servidor

Instalar Seaside



Configurar un servidor



Comprobar que funciona

The screenshot shows a web browser window with the address bar set to 'localhost'. The main content area displays the 'Welcome to Seaside 3.2' page. On the left, there's a yellow star icon next to the title 'Welcome to Seaside 3.2' and a message 'Congratulations, you have a working Seaside environment.' Below this, under 'Getting started', there are three numbered steps: 1. Try out some examples, 2. Create your first component, and 3. Browse the documentation. Step 1 lists 'Counter', 'Multi-Counter', and 'Task' as examples. Step 2 shows a form to 'Name your component' with 'MyFirstComponent' entered and a 'Create' button. Step 3 lists the 'Seaside Book' and 'Seaside Tutorial'. To the right of the main content are two sidebar sections: 'Join the community' (with links to the mailing list and Seaside site) and 'Diving In' (with links to browse applications, configure the environment, check examples, and access changes and add-on libraries). The bottom of the browser window shows standard navigation and status bars.

Welcome to Seaside 3.2

Congratulations, you have a working Seaside environment.

Getting started

Test the water with the steps below:

1. Try out some examples
 - [Counter](#), a simple Seaside component.
 - [Multi-Counter](#), showing how Seaside components can be re-used.
 - [Task](#), illustrating Seaside's innovative approach to application control flow.
2. Create your first component

Name your component:
3. Browse the documentation
 - The [Seaside Book](#) will teach you all you need to know about Seaside and how to build killer web applications.
 - The [Seaside Tutorial](#) has 12 chapters and introduces a sample application to explain the main features of Seaside.

the Seaside site Search

Join the community

Join the [mailing list](#) to ask questions and get help.

the mailing list Search

Diving In

[Browse](#) the applications installed in your image.

[Configure](#) your Seaside development environment.

Check out examples of Seaside's [JQuery](#) and [JQuery UI](#) integration.

[Seaside 3.2 changes](#)

[Seaside add-on libraries](#)

Display a menu Configure Halos Profile Memory XHTML 0/0 ms

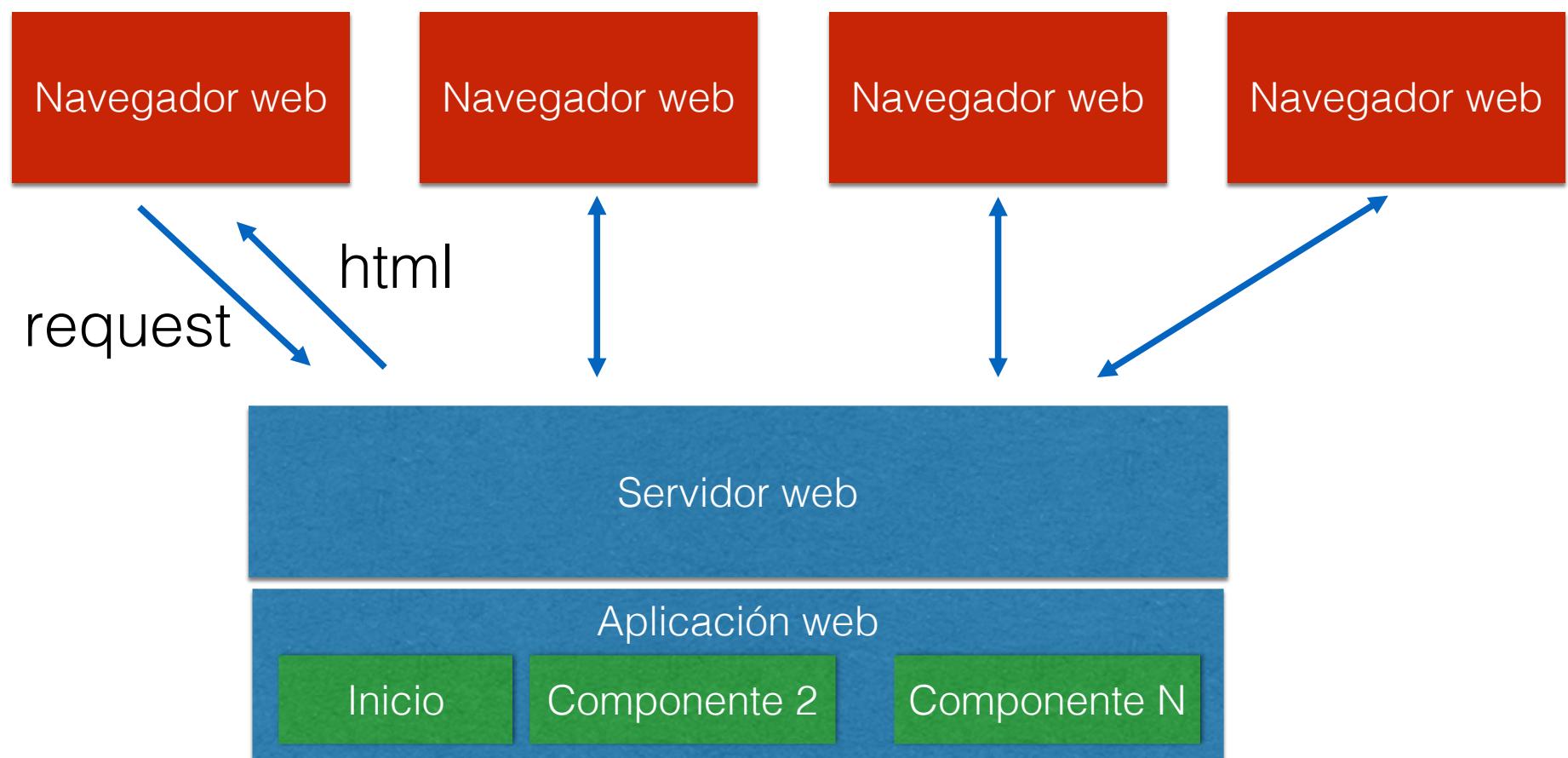
Seaside: framework web

- Resuelve (no debo preocuparme por) lo que es común a todas las aplicaciones web
 - Atender pedidos web
 - Mantener las sesiones (estado de la navegación, algunos datos, etc.)
 - Generar y devolver HTML válido
 - Etc.

Seaside: framework web

- Establece las reglas que debo seguir para:
 - Definir componentes (páginas de mi aplicación)
 - Conectar componentes (navegación, composición)
 - Responder a acciones del usuario (formularios, links, botones)
- Determina que se puede hacer fácil y que no

Anatomía (simplificada) de una aplicación Web de Seaside



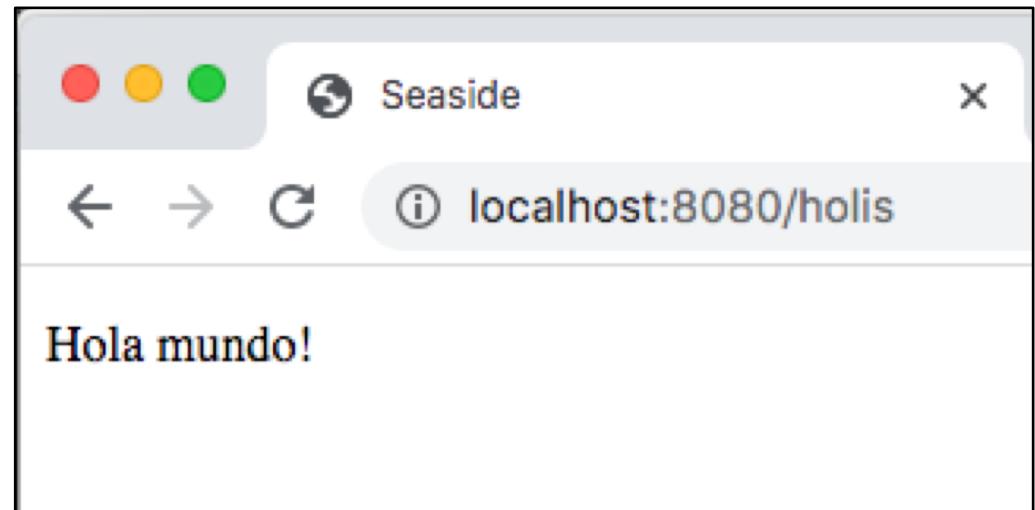
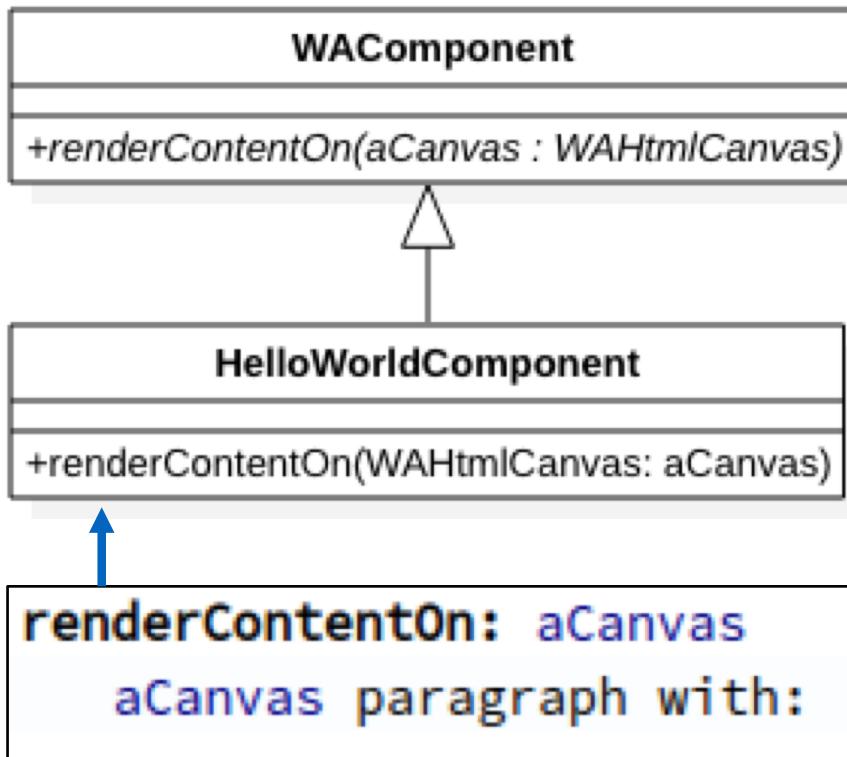


- El servidor escucha pedidos de los navegadores web
- Por cada primer pedido, crea una sesión que recordará el estado de la interacción de ese usuario
- Los pedidos son derivados a componentes (objetos) que se encargan de construir las páginas que ve el usuario

Seaside en 3 pasos

- 1) Creo una subclase de WAComponent para el componente principal. (inicio) de mi aplicación
- 2) Implemento el método #renderContentOn: en ese componente
- 3) Registro el componente para que atienda los pedidos para mi aplicación

Aplicación "Hola mundo"



¿Qué vimos?

- Como preparar Seaside
- Como programar una aplicación de una sola página
- Como registrar la aplicación en el servidor
- Cómo crear hipervínculos (anchors) y reaccionar cuando los navegan (callbacks)

Lienzo y pinceles

- `#renderContentOn:` recibe como parámetro un lienzo (WAHtmlCanvas)
- El lienzo entiende mensajes para cambiar de pincel y hacer cosas diferentes
 - `#paragraph` si quiero un párrafo
 - `#strong` un pincel de negrita
 - ...
 - Los pinceles entienden `#with:` y el contenido puede ser un String o Bloque
 - Cada pincel entiende mensajes específicos ...

Anchor / Callback

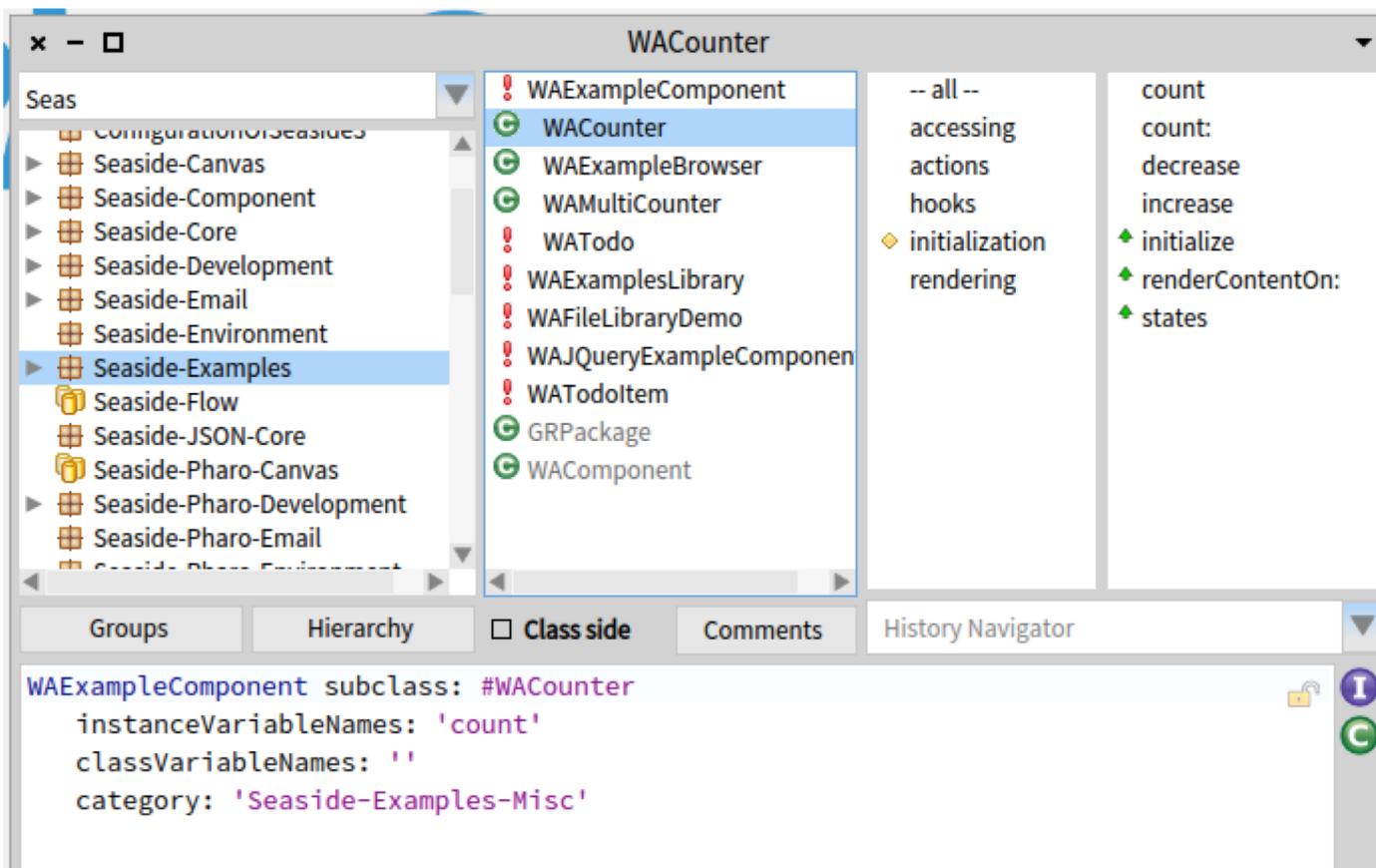
Nope, intentá de nuevo

```
aCanvas anchor  
  callback: [ self guessAgain ];  
  with: 'Nope, intentá de nuevo'
```

- Anchor es el pincel que se usa para crear links
- #callback: configura al link con un bloque
- #with: (siempre último), establece el contenido del link (texto, imagen..)

- Cuando el usuario navega (cliquea) el link:
 - Se ejecuta el bloque del callback
 - Se refresca la página ó
 - Se muestra un componente nuevo
- De esa manera, el usuario percibe el efecto de su acción

Explorar el framework



¿Qué no vimos?

- Como tener un modelo de dominio más interesante (con más objetos)
 - Como separar bien el modelo de dominio y la aplicación (la interfaz gráfica)
- Como hacer formularios
- Como conectar componentes entre sí (navegación)