**14 Глава**

Seaside на прикладі

Seaside це фреймворк для створення веб-додатків в Smalltalk, розроблений Аві Браянтом (Avi Bryant) та Джуліаном Фітзелем (Julian Fitzell) в 2002 році. Після освоєння Seaside, створення веб-додатків стає майже таким простим, як створення настільних програм. Seaside є справді цікавим для розробки швидких складних додатків. Наприклад <http://allstocker.com>, Quuve, FinWorks і CableExpertise - додатки з http://www.pharo.org/success, були розроблені на Seaside.

Seaside є незвичний в тому, що він є повністю об’єктно-орієнтована: немає ні HTML шаблонів, ні складних потоків керування через веб-сторінки, ні кодування стану в URL. Замість цього ми просто надсилаємо повідомлення об’єктам. Справді чудова ідея!

**14.1 Для чого нам потрібен Seaside?**

Сучасні веб-додатки намагаються взаємодіяти з користувачами так само, як настільні додатки: вони задають користувачу питання і користувач відповідає, зазвичай заповнюючи форму або натискаючи кнопку. Але веб працює іншим чином - браузер користувача робить запит на сервер, і сервер відповідає новою веб-сторінкою. Отже фреймворкам розробки веб-розробки доводиться мати справу з цілою низкою проблем, головною з яких є управління оберненим потоком управління. Через це, багато веб-додатків намагаються заборонити використання кнопки “Назад” в браузері в зв’язку з складнощами зберігання стану сесії. Виражати нетривіальні потоки управління спільні для кількох веб-сторінок зазвичай є складно, і множинні потоки управління може бути важко або неможливо виразити.

Seaside це є компонентно-базований фреймворк, який робить веб-розробку простіше декількома шляхами. По-перше, потік управління може бути природньо виражений використовуючи відправку повідомлень

**253**

Seaside на прикладі

Seaside слідкує за тим, яка сторінка відповідає якій точці в виконанні веб-додатку. Це означає, що кнопка “Назад” в браузері працює правильно.

По-друге, фреймворк управлятиме станом за нас. В нас буде вибір, чи вмикати повернення стану, так щоб навігація “Назад” скасовувала побічні ефекти. Як альтернатива, ми можемо використовувати вбудовану в Seaside підтримку трансакцій щоб не запобігати ануляції постійних побічних ефектів коли користувач використовує кнопку повернення назад. Нам не потрібно зберігати інформацію про стан в URL — це теж зроблено автоматично за нас.

По-третє, веб-сторінки будуються з вкладених компонентів, кожен з яких підтримує свій, незалежний потік управління. Тут немає жодних HTML шаблонів — замість цього правильний HTML генерується програмно, використовуючи простий протокол базований на Smalltalk. Seaside підтримує Cascading Style Sheets (CSS), отже вміст і розміщення елементів явно розділене..

Нарешті, Seaside надає зручний веб-базований інтерфейс для програміста, що дозволяє легко ітеративно розробляти додаток, інтерактивно зневаджувати додатки, та перекомпільовувати і розширювати додатки не зупиняючи роботу сервера.

**14.2 Розпочинаємо**

**Seaside спільнота**

Seaside.st веб-сайт (http://www.seaside.st) містить багато ресурсів, включаючи завантаження, документацію і тренування пов’язані з Seaside. (Зверніть увагу що Seaside помітно еволюціонував з роками і не всі доступні матеріали відносяться до останньої версії). Також є декілька активних емейл-розсилок, які можна знайти на http://www.seaside.st/community/mailinglist.

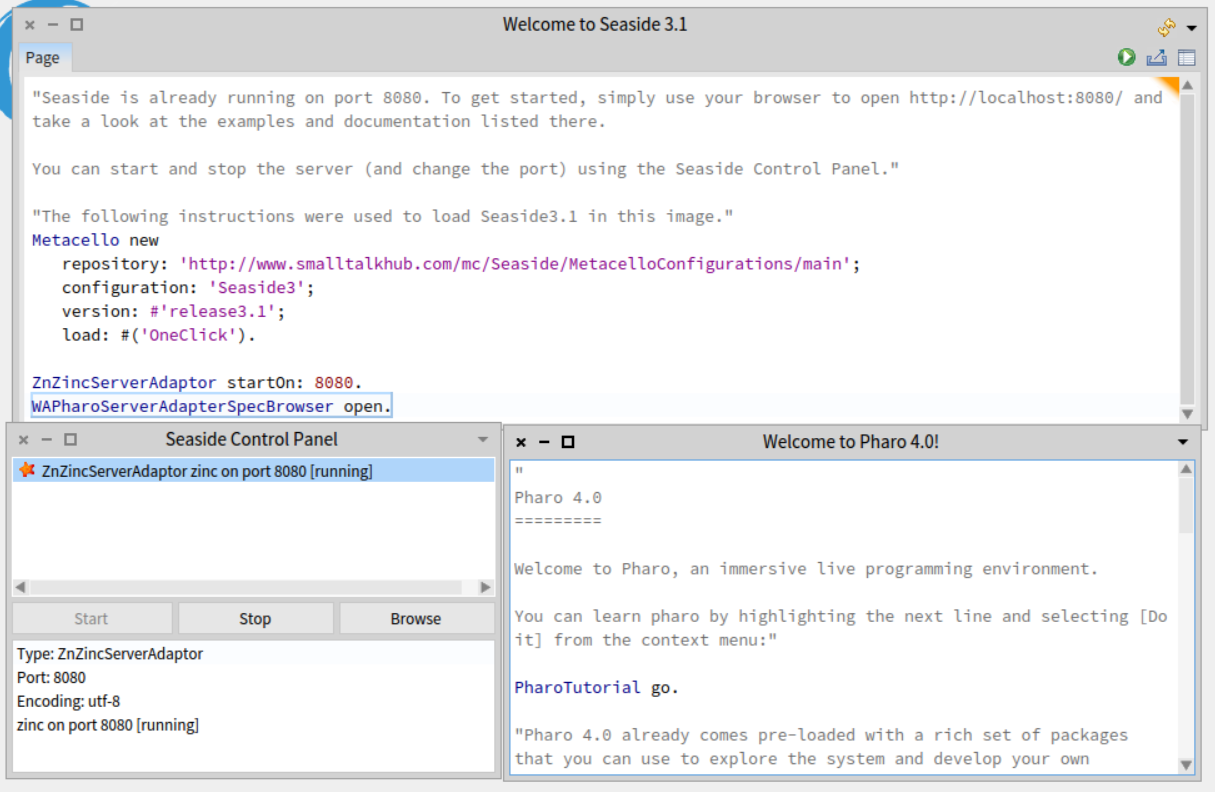
Також, вікі на GitHub репозиторії Seaside (https://github.com/seasidest/seaside/wiki) надає важливу документацію, включаючи примітки до релізу кожної версії Seaside.

**Встановлення Seaside в один клік**

Найпростіший спосіб розпочати, це завантажити Seaside One-Click Experience 3.1 з секції Pharo Downloads на веб-сайті Seaside (http://www.seaside.st/download/pharo). Це є запакована версія останньої стабільної версії Seaside для Mac OSX, Linux and Windows, зібрана на Pharo 4. Поточна остання версія Seaside - 3.2. Це є повний дистрибутив Pharo, який включає в себе Pharo VM так як і образ системи для розробника з наперед завантаженими пакетами повязаними з Seaside. Також, при запуску образу з’являється корисна панель керування Seaside.

Завантажуємо і запускаємо образ Seaside One-Click (можемо звернутись до секції “Розпочинаємо” з глави “Швидкий огляд Pharo” для глибшого пояснення

**254**

14.2 Розпочинаємо

Малюнок 14.1: Запуск образу Seaside One-Click Experience.

щодо встановлення образу Pharo). Після запуску, ми маємо побачити щось схоже на малюнок 14.1 - знайоме середовище Pharo з деякими наперед відкритими вікнами, такими я вікно привітання та панель керування Seaside.

**Запуск сервера Seaside**

Найпростіший спосіб запуску і зупинки веб-серверу Seaside - через контрольну панель Seaside. Якщо ми використовуємо образ Seaside One-Click Experience, вона вже має бути відкрита (з запущеним адаптером сервера) коли ми відкриємо образ вперше. Ми можемо вручну відкрити її коли захочемо виконавши WAPharoServerAdapterSpecBrowser open.

Натисканням правої клавіші поверх контрольної панелі Seaside дозволить додавати нам нові серверні адаптери. Наприклад, ми можемо додати ZnZincServerAdaptor, визначити номер порту, і запустити його використовуючи кнопку Старт на контрольній панелі, яка запускає новий Seaside сервер який прослуховує заданий порт

ZnZincServerAdaptor startOn: 8080. "start on port 8080"

ZnZincServerAdaptor stop.

Ми також можемо запускати і зупиняти веб-сервер Seaside з пісочниці, надсилаючи повідомлення startOn: i stop: нашим серверним адаптерам..

**255**

Seaside на прикладі

**Вітальна сторінка Seaside**

Коли сервер Seaside запущено, перейдемо за посиланням http://localhost:8080/ в веб-браузері. Ми повинні побачити веб-сторінку, яка виглядає подібно до малюнку 14.2.

Сторінка привітання містить посилання на приклади додатків Seaside. Також тут є посилання на різні документи і ресурси, і (на боковій панелі) є посилання на додатки Configuration i Browse, які дозволяють взаємодіяти з додатками зареєстрованими в нашому образі Seaside.

Глянемо на один з прикладів додатка який демонструє компонент Counter - переходимо за посиланням Counter.

Ця сторінка - невеликий додаток: він відображає рахівник, який може бути зменшений або збільшений натисканням на посилання ++ та -- (дивись малюнок 14.3)

Пограємось з рахівником натискаючи на ці посилання. Використаємо кнопку Назад, щоб повернутись до попереднього стану, і натиснемо ++ знову. Зауважимо, що рахівник коректно збільшив своє значення, враховуючи поточний стан, а не стан який в нього був перед використанням кнопки назад.

Зауважимо, що панель внизу веб-сторінки на малюнку 14.2. Seaside підтримує поняття сесій, щоб слідкувати за станом додатка для різних користувачів. New Session почне нову сесію додатка-рахівника. Configure дозволить нам налаштувати наш додаток через веб-інтерфейс. (Щоб закрити Configure, натиснемо на “х” в правому верхньому правому куті.) Halos надає можливість переглянути стан додатна який запущений на сервері Seaside. Profile and Memory надає детальну інформацію про продуктивність під час виконання додатка. XHTML може бути використаний для перевірки згенерованої веб-сторінки, але працює тільки коли веб-сторінка є публічно доступна в мережі, адже використовує W3C сервіс перевірки.

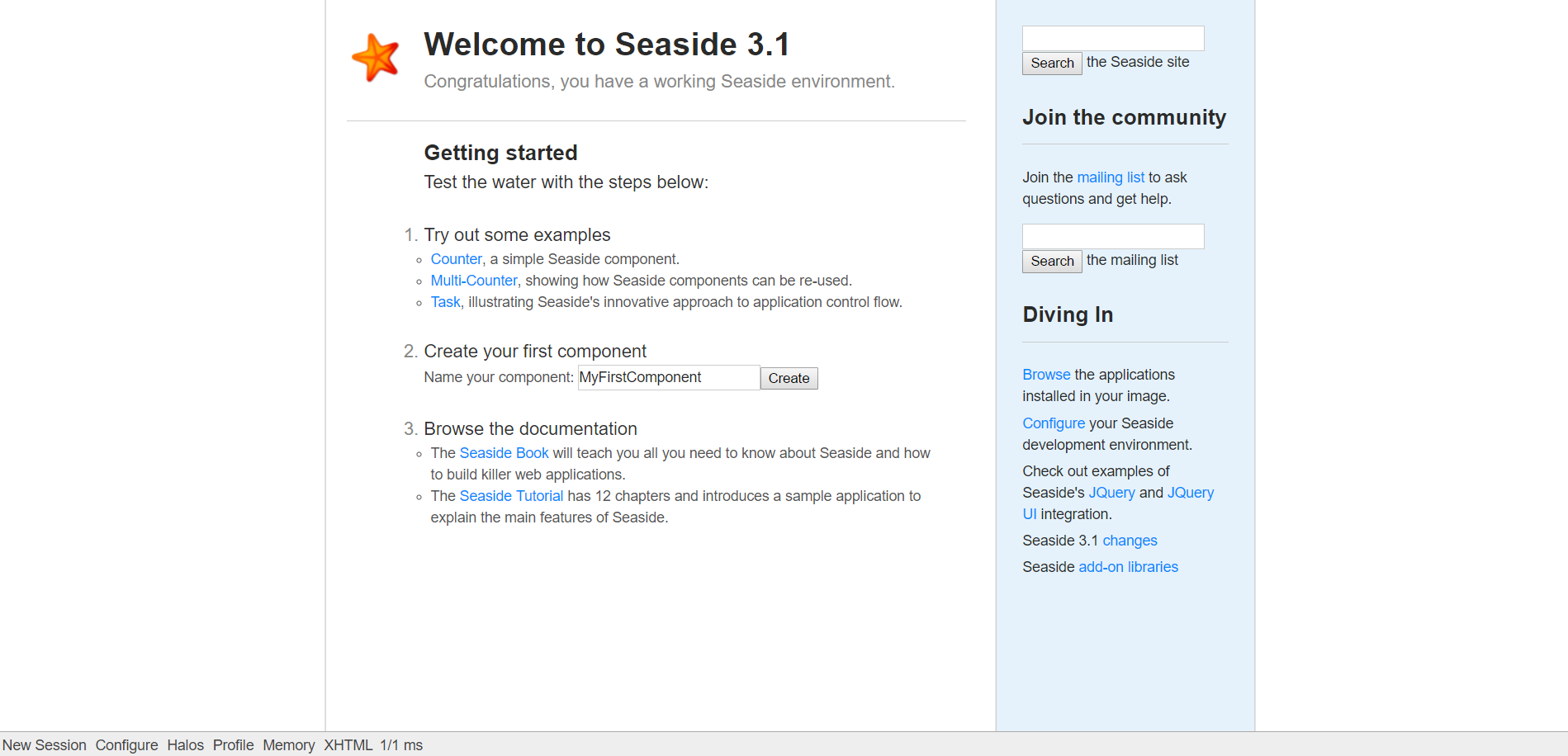
**Окремі компоненти**

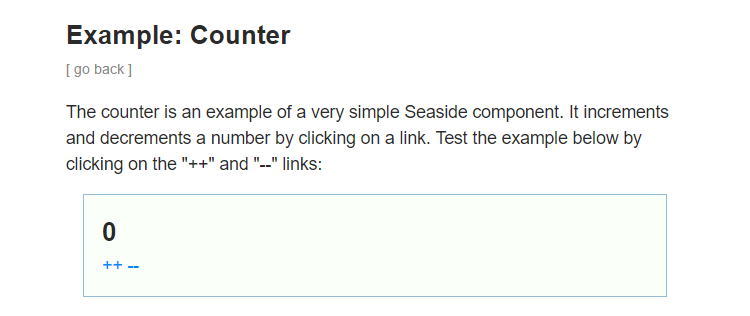
Seaside додатки зроблені з вставних компонентів. По факту, компоненти це звичайні об’єкти Smalltalk. Єдина їх особливість це те, що всі вони наслідуються від класу з фреймворку Seaside - WAComponent. Ми можемо оглядати компоненти на їх класи з образу Pharo, або напряму з веб-інтерфейсу використовуючи ореоли (Halo).

Натиснемо на посилання Halos внизу сторінки, щоб перемикати показування ореолів. Ми бачимо декілька вкладених компонентів, кожен з яких має свої іконки ореолів, включаючи компонент Counter (можливо доведеться проскролити трохи до низу, щоб побачити його), як на малюнку 14.4.

В верхньому лівому куті компонента, текст “WACounter” вказує на клас компонента Seaside, який реалізовує його поведінку. Біля нього є три іконки на які можна натиснути. Перша - Class Browser (іконка з записником і олівцем),

**256**

Малюнок 14.2: Вітальний додаток Seaside на <http://localhost:8080/>.



Малюнок 14.3: Рахівник.

Seaside на прикладі

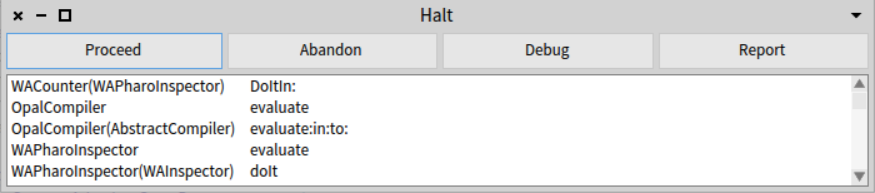
Figure 14.4: Ореоли.

Figure 14.5: Зневадження та запуск додатка-рахівника в образі Pharo.

який відкриває (веб-базований) браузер класів Seaside на класі цього компонента. Другий - це Інспектор Об’єктів (записник зі збільшувальною лінзою), який відкриває інспектор об’єктів на конкретному екземплярі WACounter. Третій - це Менеджер Стилів CSS (іконка з кольоровими кільцями), відкриває вікно Seaside відображення CSS таблиці стилів для цього компонента.

В правому верхньому куті компонента, посилання Render/Source дозволяють перемикатись між Рендереним і Вихідним виглядом вихідного HTML коду компонента. Поексперементуємо з усіма посиланнями. Зауважимо, що посилання ++ і -- також активні в відображенні вихідного коду.

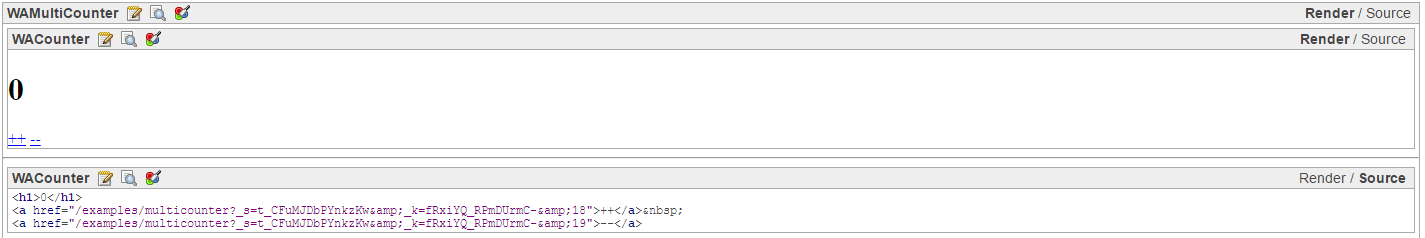
Браузер класів Seaside і Інспектор об’єктів є досить зручними, коли сервер запущений на іншому комп’ютеры, особливо, якщо сервер не має дисплею або розміщений віддалено. При першій розробці додатка Seaside, сервер буде запущений локально, і можна просто використати звичні засоби Pharo на образі сервера.

Використання посилання на Інспектор Об’єктів в браузері відкриє інспектор на об’єкті рахівника в Pharo наберемо  
self halt і натиснемо кнопку doIt.

Форма надішлеться і браузер зависне з індикатором завантаження. Тепер відкриємо образ Pharo Seaside. Ми бачимо вікно пре-дебагера (малюнок 14.5), який показує об’єкт WACounter, що виконує зупинку. Оглянемо це виконання в дебагері, і натиснемо Proceed. Повернемось до веб-браузера і зауважимо, що додаток-рахівник знову працює

**258**

14.2 Розпочинаємо



Малюнок 14.6: Незалежні підкомпоненти.

**Декілька компонентів**

Компоненти Seaside можуть бути інстанційовані декілька разів в різних контекстах.

Зараз, перейдемо за посиланням http://localhost:8080/examples/multicounter.

Ми побачимо простий додаток створений з декількох незалежних екземплярів компонента WACounter вкладених в WAMultiCounter. Збільшимо і зменшимо декілька рахівників. Перевіримо, що вони поводять себе коректно, навіть якщо ми використаємо кнопку Назад. Перемкнемо ореоли, щоб побачити як додаток побудований з вкладених компонентів. Використаємо браузер класів Seaside щоб переглянути імплементацію WAMultiCounter. Ми маємо бачити два методи на стороні класу (description, і initialize) і три на стороні екземпляру (children, initialize, і renderContentOn:). Зауважимо, що додаток це просто компонент, який хоче бути в корені ієрархії вкладеностей компонентів; це бажання показується викначенням статичного методу canBeRoot який відповідає true.

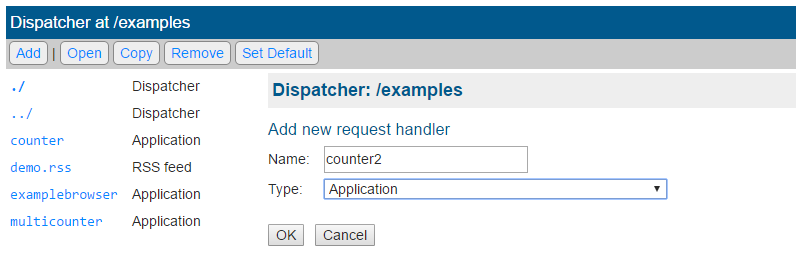
Ми можемо використати веб-інтерфейс Seaside щоб налаштовувати, копіювати чи видаляти окремі додатки (які є елементами кореневого рівня). Спробуємо зробити наступну зміну конфігурації

Перейдемо в браузері на http://localhost:8080/config.

Якщо нас попросять залогінитись, вказуємо логін і пароль за замовчуванням (admin and seaside). Ми можемо побачити список зареєстрованих додатків і диспетчерів.

**259**

Seaside на прикладі

Малюнок 14.7: Додавання нового додатка.

Перейдемо на список диспетчерів з examples натиснувши посилання зліва. За допомогою кнопок на панелі вгорі, ми можемо додати нові обробники запитів, відкрити поточні, копіювати, видаляти чи визначити його як сторінку за замовчуванням. Ми хочемо додати новий додаток-приклад. Натиснемо Add, введемо назву “counter2” і виберемо тип - Application, натиснемо OK (малюнок 14.7).

На наступному екрані визначимо кореневий компонент (Root Component) - WACounter, застосуємо зміни (малюнок 14.8).

Тепер ми маємо новий рахівник на http://localhost:8080/examples/counter2. Використаємо кнопку Remove, щоб видалити його.

Seaside працює в двох режимах: режим розробки, який ми бачили до цього, та режим розгортання, в якому панель інструментів недоступна.

Ми можемо помістити додаток в режим розгортання забравши унаслідуваний Root Decoration Class.

Також, можна вимкнути панель розробника для всіх нових додатків запустивши код::

AAdmin applicationDefaults

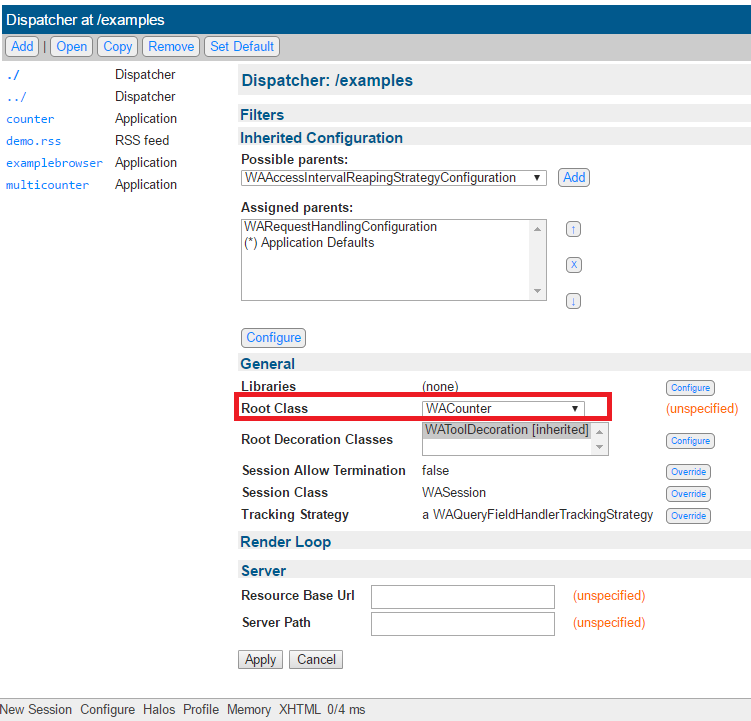
removeParent: WADevelopmentConfiguration instance

Якщо ми хочемо ввімкнути захист паролем, ми повинні додати WAAuthConfiguration в наслідувану конфігурацію. Оберемо WAAuthConfiguration зі списку можливих батьків, після додавання цієї конфігурації, ми можемо визначити налаштування логіну і паролю..

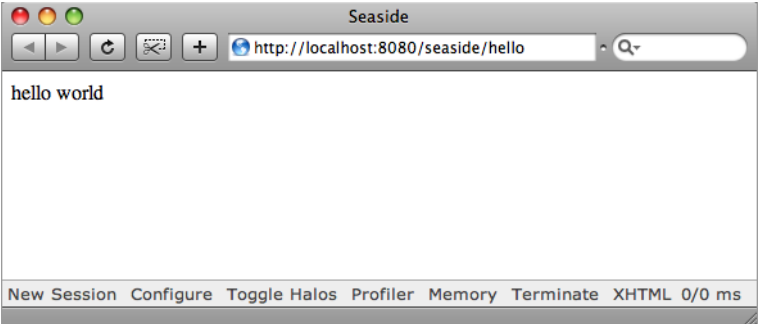
**14.3 Компоненти Seaside**

Як ми вже згадували в попередній секції, додатки Seaside побудовані на компонентах. Розглянемо детальніше, як працює Seaside реалізувавши

**260**

Малюнок 14.8: Налаштування нового додатка.

Seaside на прикладі



**Figure 14.9: Hello World in Seaside.**

компонент HelloWorld.

Кожен компонент Seaside повинен наслідуватись напряму, чи не напряму від WAComponent, як показано на малюнку 14.10. (Префікс ’WA’ часто використовується в Seaside і означає’Web Application’.)

Визначимо підклас WAComponent і назвемо його WAHelloWorld.

Компоненти мають знати, як промалювати себе. Зазвичай це досягається реалізацією методу renderContentOn:, який отримує аргументом екземпляр WAHtmlCanvas, який знає як промальовувати HTML.

Реалізуємо наступний метод і помістимо його в протокол rendering:

WAHelloWorld >> renderContentOn: html

html text: 'hello world'

Тепер ми маємо повідомити Seaside що цей компонент може бути окремим додатком..

Реалізуємо наступний статичний метод в класі WAHelloWorld.

WAHelloWorld class >> canBeRoot

^ true

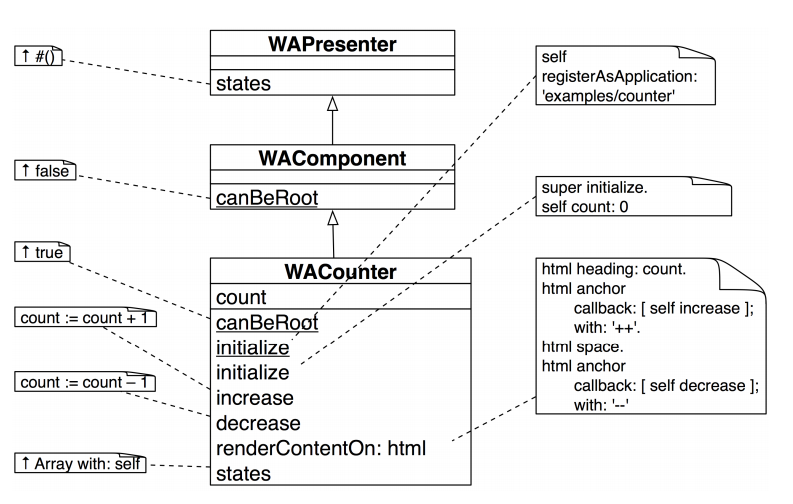
Майже готово!

Відкриємо браузер на сторінці <http://localhost:8080/config>, додамо нову точку входу з назвою hello, і визначимо їй кореневий компонент - WAHelloWorld.

Тепер перейдемо на http://localhost:8080/hello. Ось і все! Ми бачимо сторінку схожу до малюнку 14.9.

**262**

14.3 Компоненти Seaside



Малюнок 14.10: Клас WACounter, який імплементує додаток-рахівник. Підкреслені методи - статичні; решта - на стороні екземпляра класу.

**Зворотнє прослідковування стану і додаток-рахівник**

Додаток рахівник лиш трохи складніший за додаток HelloWorld.

Клас WACounter є окремим додатком, отже він повинен відповідати true на повідомлення canBeRoot. Також він повинен зареєструвати себе як додаток; це робиться в статичному методі initialize, як показано на малюнку 14.10.

WACounter визначає два методи, increase і decrease, які будуть викликані з посилань ++ і -- на веб-сторінці. Він такожи визначає поле count щоб записувати стан рахівника. Однак, ми також хочемо щоб Seaside синхронізував рахівник з сторінкою браузера: коли користувач натискає Назад, ми хочемо, щоб Seaside прослідкував стан об’єкту WACounter. Seaside включає загальний механізм для зворотнього прослідковування, але кожен додаток також повинен вказати, які частини стану повинні прослідковуватись.

Компотент вмикає зворотнє прослідковування визначивши метод на states на стороні екземпляру: states повинен відповідати масивом, який містить всі об’єкти, які повинні прослідковуватись. В цьому випадку об’єкт WACounter додає себе в Seaside до таблиці об’єктів які прослідковуються повертаючи  
Array with: self.

**Нюанс зворотнього прослідковування** Є тонке, але важливе зауваження щодо об’єктів, які декларуються для прослідковування. Seaside прослідковує стан роблячи копії

**263**

Seaside на прикладі

всіх об’єктів визначених в масиві states. Це робиться з використанням об’єкту WASnapshot object; WASnapshot є підкласом IdentityDictionary який записує об’єкти як ключі, а їх неглибокі копії - як значення. Якщо стан додатка прослідковується до певного збереженого стану, то стан кожного об’єкту переписується копією зі збереженого стану.

Тут є момент, за яким варто дивитись: в випадку WACounter, ми можемо подумати, що стан який потрібно прослідковувати це число - значення поля count. Але відповідь states методу - Array with: count не спрацює. Це станеться тому, що об’єкт count є integer, а integer-и є незмінними. Методи збільшення і зменшення не змінюють стан об’єкта 0 в 1 чи 3 в 2. Замість цього, вони змусять count вказувати на інше число: щоразу коли count збільшується чи зменшується, в count перезаписується інший об’єкт. Ось чому WACounter>>states повинен повертати Array with: self. Коли стан об’єкта WACounter заміняється на попередній, значення кожного поля заміняється на попереднє значення; це правильно замінить поточне значення count попереднім значенням.

**14.4 Відображення HTML**

Метою веб-додатка є створення або відображення веб-сторінок. Як ми згадували в секції 14.3, кожен компонент Seaside відповідальний за відображення самого себе. Тож, почнемо наш огляд відображення з того, що побачимо, як компонент-рахівник відображає сам себе.

**Відображення рахівника**

Відображення рахівника є досить тривіальним; код зображено на малюнку 14.10. Поточне значення рахівника зображене, як заголовок HTML, а операції збільшення і зменшення реалізовані, як HTML якорі (тобто, посилання) з зворотнім викликом блоків які надішлють increase і decrease об’єкту рахівника.

Далі ми детальніше розглянемо протокол відображення. Але перед цим, швидко проглянемо мульти-рахівник.

**Від рахівника до мульти-рахівника**

WAMultiCounter, зображений на малюнку 14.11 також є окремим додатком, тож від перевизначає canBeRoot на відповідь true. Більше того, це композитний компонент, тож Seaside вимагає щоб він визначав свої дочірні компоненти, реалізуючи метод children який відповідає масивом всіх компонентів які він містить. Від відмальовує себе, відмальовуючи свої підкомпоненти, розділені горизонтально. Окрім

**264**