Extended results of Tornado: A Run-Fail-Grow approach for Dynamic Application Tailoring
Guillermo Polito, Stéphane Ducasse, Noury Bouraqadi, Luc Fabresse

To cite this version:
Guillermo Polito, Stéphane Ducasse, Noury Bouraqadi, Luc Fabresse. Extended results of Tornado: A Run-Fail-Grow approach for Dynamic Application Tailoring. [Research Report] Inria. 2014. <hal-00996908v3>

HAL Id: hal-00996908
https://hal.archives-ouvertes.fr/hal-00996908v3
Submitted on 15 Jul 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Extended results of Tornado
A Run-Fail-Grow approach
for
Dynamic Application Tailoring
Abstract

Producing a small deployment version of an application is a challenge because static abstractions such as packages cannot anticipate the use of their parts. As such, an application often occupies more memory than actually needed. To solve this problem we propose Tornado, a technique to dynamically tailor applications to only embed code (classes and methods) they use. Tornado uses a run-fail-grow approach to prepare an application for deployment. It launches minimal version of an application and installs a minimal set of statements that will start the user’s application. This application is run and these statements are executed. When the application fails because there are classes or methods missing, the necessary code is installed. The application is executed until it reaches a stable point, allowing possibly human interaction for applications with UIs. Thus, Tornado creates minimal memory footprint versions of applications by tailoring the whole application’s code, including run-time and third party libraries.

In this report, we present the results we obtained from using Tornado to tailor two different applications. We succeeded to tailor a hello world application to occupy 1% of its original size. We also experimented with a Seaside web application tailoring in one case only the application’s and framework’s code and the whole application’s code in the other case. In this latter example, we reached memory savings of about 97%. In this report we present an overview on Tornado, and we give details of the results we obtained.
## Contents

1. Used Methodology .......... 3
2. Hello World Application ...... 3
3. Seaside Web Application ...... 3
   A. Appendix: Method List of a Nurtured Hello World Application .......... 6
   B. Appendix: Entry Points to Tailor the Seaside Web Application .......... 8
   C. Appendix: Method List of Seaside Counter Application with Full Pharo Seed .......... 9
   D. Appendix: Method List of Seaside Counter Application with Empty Seed .......... 15
1 Used Methodology

We tested our Tornado implementation by tailoring two different Pharo applications: a hello world application and a simple but yet interactive web application based on the Seaside framework [1]. Our methodology consisted in: setting up a seed for the application, preparing the application entry points and executing the application. In the case of the interactive web application, we interacted with it through a web browser. Once we finished the process, we extracted the resulting application by making a snapshot of it in a Pharo image file. We tested the generated snapshots to verify they work properly (under the assumption that only the previously used features of the application should work).

Finally, to present our results we measured the size of the generated snapshots files and compared them with the snapshots of the full applications under Pharo’s production option1. The results prove the soundness of our solution.

2 Hello World Application

We used Tornado to tailor a hello world application writing 10 times the ‘hello world’ string to the standard output (stdout). In this case study we used an empty seed to grow both base libraries and the application’s code. Figure 1 shows the installed entry point to tailor this application. Table 1 shows our results for this case. We succeed to reduce the application’s size to 1% of its original counterpart.

```
FileStream startUp: true.

1 to: 10 do: [ i | FileStream stdout nextPutAll: 'hello'; crlf ].
```

Figure 1: Entry point of the Hello World application with an empty seed.

<table>
<thead>
<tr>
<th>Size(KB)</th>
<th>Occupied(%)</th>
<th>Saved(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>12872</td>
<td>100%</td>
</tr>
<tr>
<td>Tailored</td>
<td>131</td>
<td>1%</td>
</tr>
</tbody>
</table>

Table 1: Results of the tailored Hello World application.

3 Seaside Web Application

We also used Tornado to tailor a simple web application consisting in a webpage with a counter containing two buttons. These two buttons perform requests to the web server to increase and decrease the counter. The Seaside application framework was configured with its default values, without making any customizations.

1Pharo allows to prepare a snapshot for production. This option cleans some caches and removes some well known objects from the system, thus, freeing space.
In this case, we used two different seeds for tailoring: a seed containing all Pharo base libraries and an empty seed. Appendix B presents the entry points for these both seeds. The tailoring was done by starting the application and exercising it by generating requests through a web browser, clicking on its decrease and increase buttons.

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref. Pharo Base Libraries (P)</td>
<td>12872</td>
</tr>
<tr>
<td>Ref. Seaside Framework (S)</td>
<td>4326</td>
</tr>
<tr>
<td>Ref. Counter Application (R)</td>
<td>52</td>
</tr>
<tr>
<td>Total Ref. Application (P+S+C)</td>
<td>17250</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Size (KB)</th>
<th>Occupied (%)</th>
<th>Saved (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P'+S'+C'/P+S+C$</td>
<td>573</td>
<td>3%</td>
</tr>
<tr>
<td>$P''+S''+C''/P+S+C$</td>
<td>13090</td>
<td>76%</td>
</tr>
<tr>
<td>$S''+C''/S+C$</td>
<td>218</td>
<td>5%</td>
</tr>
</tbody>
</table>

Table 2: Results of second case study. Results of tailoring a web application with two different seeds. On the left, the total sizes of the original application deployment components (base libraries, application framework and counter application). On the right, our results when applying after tailoring. The first two results rows are compared against the total of the reference application. The third row presents the comparison without including base libraries, already inside the seed.

Table 2 shows the results obtained when tailoring this application with each of these two seeds. Figure 2 presents a tailoring map illustrating how Tornado selects the code units from a reference application given a seeds. This figure also presents the notation we use in Table 2: $P$ is the Pharo base libraries, $S$ is the Seaside Framework and $C$ is the Counter application code units present in the reference application. $P'$, $S'$ and $C'$ are their counterparts selected by Tornado when using an empty seed. $P''$, $S''$ and $C''$ are their counterparts, as selected by Tornado when using a seed with all base libraries. In the latter, we can note that $P=P''$.

Figure 2: Tailoring Map. Tailoring map describing the Seaside application generated with the empty seed (left) and the full Pharo seed (right).

Acknowledgements. This work was supported by Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council, FEDER via the ‘Contrat de Projets Etat Region
References

Appendix: Method List of a Nurtured Hello World Application

List of methods extracted from the nurtured Hello World application. This list includes all methods installed from the Pharo base libraries and the simple Hello World application.

Array class»new:
ArrayedCollection»size
Association class»key:value:
Association»value:
Association»value
BlockClosure»on:do:
BlockClosure»repeat
BlockClosure»valueNoContextSwitch
ByteString class»compare:with:collated:
ByteString class»findFirstInString:inSet:startingAt:
ByteString class»stringHash:initialHash:
ByteString»at:put:
ByteString»at:
ByteString»isByteString
ByteString»replaceFrom:to:with:startingAt:
FileStream class»openOnHandle:name:forWrite:Standard
FileStream»primWrite:from:startingAt:count:Standard
GreekEnvironment class»supportedLanguages
HashTableSizes class»atLeast:
HashTableSizes class»sizes
HashedCollection class»newProto
HashedCollection»atNewIndex:put:
HashedCollection»findElementOrNil:
HashedCollection»fullCheck
HashedCollection»grow
JapaneseEnvironment class»supportedLanguages
KoreanEnvironment class»supportedLanguages
Latin1Environment class»supportedLanguages
Latin2Environment class»supportedLanguages
Latin9Environment class»systemConverterClass
Locale class»currentPlatform
Locale class»determineCurrentLocale
Locale»determineLocaleID
Locale»localeID:
Locale»localeID
LocaleID class»isoLanguage:isoCountry:
LocaleID class»isoLanguage:
LocaleID class»isoString:
LocaleID class»hash
LocaleID»isoCountry:
LocaleID»isoLanguage:
LocaleID»isoLanguage:
LocaleID»isoLanguage:
LocaleID»isoLanguage
LocaleID»isoLanguage:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LocaleID»isoCountry:
LookupKey class»key:
B Appendix: Entry Points to Tailor the Seaside Web Application

Entry points as used to tailor the Seaside web application with a full Pharo seed and an empty seed. The first one (Figure 3) only consists in starting the web server as the base libraries are initialized and available in the seed. The latter one (Figure 4) includes the initialization of the minimal runtime needed to do networking.

```
ZnZincServerAdaptor startOn: 8888.
```

Figure 3: Entry point of the Seaside application with a full Pharo seed.

```
"We initialize some classes of the system"
SmalltalkImage initializeForTornado.
Symbol initializeForTornado.
Object initialize.
ExternalSemaphoreTable initialize.
Socket initialize.
Delay initialize.
Delay startUp: true.
Delay shutDown: true.
OSPlatform initialize.
DiskStore initialize.
FileStream initialize.
NetNameResolver initialize.
DateAndTime initialize.
ProcessorScheduler initialize.
WeakFinalizationList initialize.
UUIDGenerator initialize.
WeakArray initialize.
GRPharoRandomProvider initialize.
WASlime initialize.
UIManager basicDefault: DummyUIManager new.
ZnServer initialize.
WAServerManager initialize.
Smalltalk instVarNamed: 'session' put: Smalltalk newSessionObject.
Smalltalk startupImage: true snapshotWorked: true.
"Finally we start the web server"
ZnZincServerAdaptor startOn: 8888.
```

Figure 4: Entry point of the Seaside application with an empty seed.
Appendix: Method List of Seaside Counter Application with Full Pharo Seed

List of methods extracted from the nurtured Web application when using a seed containing all base libraries from Pharo. This list includes all methods installed from Seaside framework and the counter application. The list of methods part of the base library are excluded as it is the same list of the methods found in Pharo base library.

WAActionCallback class
- WAAccessIntervalReapingStrategy ➤ defaultConfiguration
- WAAccessIntervalReapingStrategy ➤ initialize
- WAAccessIntervalReapingStrategy ➤ interval
- WAAccessIntervalReapingStrategy ➤ reap
- WAAccessIntervalReapingStrategy ➤ stored: key:
- WAActionCallback ➤ block:
- WAActionCallback ➤ evaluateWithArgument:
- WAActionCallback ➤ isEnabledFor:
- WAActionCallback ➤ signalRenderNotification
- WAActionPhaseContinuation ➤ continue
- WAActionPhaseContinuation ➤ handleRequest
- WAActionPhaseContinuation ➤ renderContext:
- WAActionPhaseContinuation ➤ renderContext
- WAActionPhaseContinuation ➤ runCallbacks
- WAActionPhaseContinuation ➤ shouldRedirect
- WAAdmin ➤ defaultServerManager
- WAAction ➤ enableFor:
- WAAdmin ➤ serverAdaptors
- WAAnchorTag ➤ callback:
- WAAnchorTag ➤ tag
- WAAnchorTag ➤ url
- WAAnchorTag ➤ with:
- WAApplication ➤ content
- WAApplication ➤ doesHandlerSupportCookies:
- WAApplication ➤ handleDefault:
- WAApplication ➤ handleFiltered:
- WAApplication ➤ isApplication
- WAApplication ➤ isImplemented:
- WAApplication ➤ keyField
- WAApplication ➤ libraries
- WAApplication ➤ mainClass
- WAApplication ➤ mimeType
- WAApplication ➤ newSession
- WAApplication ➤ resourceBaseUrl
- WAApplication ➤ sessionClass
- WAApplication ➤ configuration ➤ parents
- WAAccessIntervalReapingStrategy ➤ defaultConfiguration
- WAAttributeSearchContext ➤ initializeWithKey:
- WAAttributeSearchContext ➤ isAttributeInheritedOn:
- WAAttributeSearchContext ➤ isAttributeLocalOn:
- WACache ➤ at: ifAbsent:
- WACache ➤ at: ifPresent:
- WACache ➤ at: ifAbsent:
- WACache ➤ at: ifPresent:
- WACache ➤ at: ifAbsent:
- WACache ➤ at: ifPresent:
- WACache ➤ at: ifAbsent:
- WACache ➤ at: ifPresent:
- WACacheCapacityConfiguration ➤ describeOn:
- WACache ➤ notifyRemoved: key:
- WACache ➤ notifyRetrieved: key:
- WACache ➤ notifyStored: key:
- WACache ➤ pluginsDo:
- WACache ➤ reapingStrategy
- WACache ➤ reap
- WACache ➤ removalAction
- WACache ➤ setExpirePolicy:
- WACache ➤ setExpirePolicy:
- WACache ➤ setMissStrategy:
- WACache ➤ setMissStrategy:
- WACache ➤ setReapStrategy:
- WACache ➤ setReapStrategy:
- WACache ➤ setRemovalAction:
- WACache ➤ store:
- WACache ➤ capacityConfiguration ➤ describeOn:
D Appendix: Method List of Seaside Counter Application with Empty Seed

List of methods extracted from the nurtured Web application when using an empty seed. This list includes all methods installed from Seaside framework, the Counter application and the base library of Pharo.

```
Array class»new:
Array»isSelfEvaluating
Array»printOn:
Array»replaceFrom:to:with:startingAt:
ArrayedCollection class»new:withAll:
ArrayedCollection class»new
ArrayedCollection class»with:with:with:
ArrayedCollection class»with:with:
ArrayedCollection class»with:with:
ArrayedCollection class»mergeSortFrom:to:by:
ArrayedCollection»size
ArrayedCollection»sort:
Association class»key:value:
Association class»key:value:
Association»expireWeakKey
Association»expiredWeakKey
Association»key:WeakKey
Association»key:value:WeakKey
Association»keyWeakKey
Association»value:WeakKey
Association»value:
Association»valueWeakKey
Association»value:
BlockClosure»argumentCount
BlockClosure»asContextWithSender:
BlockClosure»asContext
BlockClosure»assert
BlockClosure»cull:
BlockClosure»ensure:
BlockClosure»fixCallbackTemps
BlockClosure»forkAt:.named:
BlockClosure»forkAt:
BlockClosure»ifCurtailed:
BlockClosure»ifError:
BlockClosure»isClosure
BlockClosure»newProcess
BlockClosure»numArgs
BlockClosure»numCopiedValues
BlockClosure»on:do:
BlockClosure»on:fork:
BlockClosure»outerContext
```

```
PositionableStream»isEmpty
PositionableStream»on:
PositionableStream»originalContents
PositionableStream»peekFor:
PositionableStream»peek
PositionableStream»position:
PositionableStream»position
PositionableStream»reset
PositionableStream»skip:
PositionableStream»skipSeparators
PositionableStream»skipTo:
Process class»forContext:priority:
Process class»primitiveResume
Process class»SuspendingList
Process class»terminate
ProcessLocalVariable class»value:
ProcessLocalVariable class»value:
ProcessSpecificVariable class»default
ProcessorScheduler class»idleProcess
ProcessorScheduler class»relinquishProcessorForMicroseconds:
Random»initialize
ReadStream»on:from:
Semaphore class»forMutualExclusion
Semaphore class»new
Semaphore»critical:
Time»nanoSecond
Time»print24:showSeconds:on:
Time»printOn:
Time»seconds
Time»second
Time»ticks:
TimeZone»offset
Timespan class»<
Timespan class»dayOfMonth
Timespan class»duration:
Timespan class»month
Timespan class»start:
Timespan class»year
True»ifFalse:
True»not
True»|

UIManager class»basicDefault:
UIManager class»default:WAUnescapedDocument»initializeWithStream:codec:
UIManager class»default
UIManager»activate
UIManager»beDefault
UIManager»boot:during:
UIManager»deactivate
UIManager»onSnapshot:
UTF16TextConverter class»encodingNames
UTF8DecomposedTextConverter class»encodingNames
UUIDGenerator class»initialize
UUIDGenerator class»startUp
UndefinedObject»encodeOn:
UndefinedObject»isNil
UndefinedObject»notNil
UndefinedObject»seasideUrl
UndefinedObject»shallowCopy
Unicode class»isDigit:
Unicode class»isLetter:
Unicode class»toUpperCase:
VirtualMachine class»allocationsBetweenGC:
VirtualMachine class»getSystemAttribute:
VirtualMachine class»interpreterClass
VirtualMachine class»interpreterSourceDate
VirtualMachine class»isPharoVM
VirtualMachine class»isRunningCogit
VirtualMachine class»maxExternalSemaphores
VirtualMachine class»parameterAt:put:
VirtualMachine class»parameterAt:
VirtualMachine class»setGCPARAMETERS
VirtualMachine class»tenuringThreshold:
ZnUrl\-enforceKnownScheme  ZnUtils class\-signalProgress:total:
ZnUrl\-hasFragment  ZnUtils class\-streamingBufferSize
ZnUrl\-hasHost  ZnZincServerAdaptor\-basicStart
ZnUrl\-hasPath  ZnZincServerAdaptor\-configureDelegate
ZnUrl\-hasPort  ZnZincServerAdaptor\-configureServerForBinaryReading
ZnUrl\-hasQuery  ZnZincServerAdaptor\-defaultCodec
ZnUrl\-hasScheme  ZnZincServerAdaptor\-defaultDelegate
ZnUrl\-hasUsername  ZnZincServerAdaptor\-defaultZnServer
ZnUrl\-isSchemeUsingPath  ZnZincServerAdaptor\-isRunning
ZnUrl\-parseFrom:defaultScheme:  ZnZincServerAdaptor\-isStopped
ZnUrl\-parseFrom:  ZnZincServerAdaptor\-requestAddressFor:
ZnUrl\-parsePath:  ZnZincServerAdaptor\-requestBodyFor:
ZnUrl\-printAuthorityOn:  ZnZincServerAdaptor\-requestCookiesFor:
ZnUrl\-printOn:  ZnZincServerAdaptor\-requestFieldsFor:
ZnUrl\-printPathOn:  ZnZincServerAdaptor\-requestHeadersFor:
ZnUrl\-printPathQueryFragmentOn:  ZnZincServerAdaptor\-requestMethodFor:
ZnUrl\-printQueryOn:  ZnZincServerAdaptor\-requestUrlFor:
ZnUrl\-query:  ZnZincServerAdaptor\-requestVersionFor:
ZnUrl\-query  ZnZincServerAdaptor\-responseFrom:
ZnUrl\-scheme  ZnZincServerAdaptor\-server
ZnUtils class\-httpDate:  ZnZincServerAdaptor\-shutDown
ZnUtils class\-httpDate  ZnZincServerAdaptor\-startUp
ZnUtils class\-nextPutAll:on: