Faça seu portal voar usando o plone.app.caching

Fabiano Weimar dos Santos [Xiru]
xiru@xiru.org

Roteiro

- Um pouco sobre mim...
- Cache É importante?
- Histórico
- plone.app.caching
- Demo

Um pouco sobre mim...

- Os amigos me chamam de Xiru
- Já escrevi algumas linhas de código do Plone, Archetypes e alguns Plone Products...
- Atualmente
  - Ministro cursos a distância sobre Plone, Segurança e Infraestrutura
  - Sysadmin do provedor PyTown.com
  - Consultor do Programa das Nações Unidas para o Desenvolvimento - PNUD

Cache **É** Importante?

- Cache é MUITO importante se você quer fazer seu site “voar”
- Não importa a tecnologia: nenhum site dinâmico sobrevive sem cache

Sobre o que eu não vou falar...

- A eficiência de um sistema depende de infraestrutura
- Eu não vou falar sobre
  - Plone + Squid
  - Plone + Varnish
  - Plone + Varnish + nginx
  - memcached
  - ... mas você deve preocupar-se em utlizá-los

Histórico

- O Plone sempre teve suporte a cache!
  - HTTP Cache Manager “/HTTPCache”
  - global_cache_settings
- Problemas:
  - Elementos não eram associados com o HTTPCache
  - global_cache_settings padrão não fazia cache compartilhado

global_cache_settings

```
<metal:cacheheaders define-macro="cacheheaders">
    <metal:block tal:content="structure python:here.enableHTTPCompression(request=request,
debug=0)" />
    <metal:block tal:define="dummy python:request.RESPONSE.setHeader('Content-Type',
'text/html;;charset=%s' % charset)" />
    <metal:block tal:define="dummy python:request.RESPONSE.setHeader('Content-Language', lang)" />
    <metal:block tal:condition="python: not here.portal_membership.isAnonymousUser()">
      <metal:block tal:define="dummy python:request.RESPONSE.setHeader('Expires', 'Sat, 1 Jan 2000
00:00:00 GMT')" />
      <metal:block tal:define="dummy python:request.RESPONSE.setHeader('Cache-Control', 'private, no-
cache, no-store, must-revalidate, post-check=0, pre-check=0')" />
    </metal:block>
    <metal:block tal:condition="python: here.portal_membership.isAnonymousUser()">
      <metal:block tal:define="dummy python:request.RESPONSE.setHeader('Expires', (DateTime()
+1.0/24).toZone('GMT').rfc822().replace('+0000', 'GMT'))" />
      <metal:block tal:define="dummy python:request.RESPONSE.setHeader('Cache-Control', 'max-age=0,
s-maxage=3600')" />
    </metal:block>
</metal:cacheheaders>
```

CacheFu “moleque”

- Uso de “monkey patches” para associar imagens e arquivos com o HTTPCache
- global_cache_settings ajustado para cache compartilhado por 10 minutos (sem PURGE)
- Em 2005, diversos bugs no Zope tiveram que ser corrigidos para isso funcionar

CacheFu

- Projeto iniciado como uma interface de configuração dos “monkey patches”
- Colaboração entre diversos desenvolvedores
- Passou a armazenar configurações de cabeçalhos e regras de cache numa “tool”

CacheFu

- Foi a melhor solução para gerenciamento de cache até o Plone 3.3.5
- Arquitetura “problemática” por armazenar as configurações no ZODB
- Deixou de ter releases a partir do Plone 4.0

plone.app.caching

- Sucessor do CacheFu a partir do Plone 4.0
- Utiliza recursos do Zope Component Architecture – ZCA
- Armazena configurações no portal_setup
  - Exportação e importação de configurações utilizando XML
  - Plone Products podem (e devem) empacotar configurações

plone.app.caching

- Empacotado no Plone 4.1!
- Perfis padrão de cache
  - Sem proxy
  - Com proxy
  - Com proxy e “split-view”

Conjuntos de Regras Padrão

- Content feed
  - RSS dinâmico
- Conteúdo de Imagens e Arquivos
  - Download de arquivos ou imagens ou utilizados na visualização de outros conteúdos
- Visualização de Pasta
  - Visualização de um objeto com comportamento de pasta, exibindo diversos conteúdos

Conjuntos de Regras Padrão

- Visualização de Conteúdo
  - Visualização de um objeto único, que não tem comportamento de pasta
- Recursos: Arquivos e Imagens
  - Inclui arquivos e imagens criados ou personalizados através da ZMI, aqueles disponibilizados através do portal_skins ou diretórios registrados no sistema de arquivos
- Arquivos e Imagens Estáveis
  - Recursos que não mudam, como CSS, JS e KSS registrados através de sua respectivas ferramentas
  - Podem ter cache para SEMPRE pois, quando um elemento muda, a URL de acesso também muda

Operações Cache Padrão

- Sem Cache
- Strong caching
  - Cache no navegador e no proxy (padrão: 24 hours).
- Moderate caching
  - Cache no navegador que expira automaticamente (o mesmo que “weak caching”), e cache no proxy (padrão: 24 hours).
  - Idealizado para utilizar uma estratégia de PURGE
    - Cuidado: pode existir mais de um proxy na rede, não apenas o Squid/Varnish configurado para essa tarefa

Operações Cache Padrão

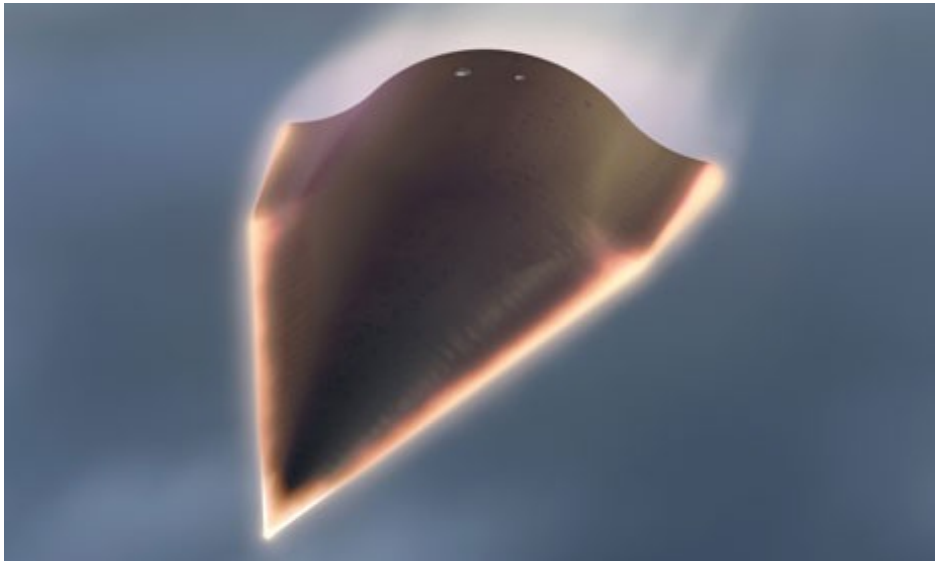
- Weak caching
  - Cache no navegador, expirado automaticamente
  - Espera um código 304 (“Not Modified”)
  - Requer cabeçalhos “Last-Modified” e/ou “e-tags”
  - Pode ser combinado com o PageCacheManager
    - Cache na memória do Zope do conteúdo da página e respectivos cabeçalhos
    - Pode ser ineficiente em ambientes distribuídos
  - Cuidado: validação condicional pode fazer com que todos os requests sejam enviados para o backend
    - Depende do proxy e de como o proxy foi configurado

Mapeamento Padrão de Regras e Operações de Cache

	without-caching-proxy	with-caching-proxy	with-caching-proxy-splitviews
itemView	weakCaching	weakCaching	moderateCaching
folderView	weakCaching	weakCaching	moderateCaching
feed	weakCaching	moderateCaching	moderateCaching
file	weakCaching	moderateCaching	moderateCaching
resource	strongCaching	strongCaching	strongCaching
stableResource	strongCaching	strongCaching	strongCaching

PageCacheManager

- E-tags muito granulares = explosão combinatória
- E-tags muito abrangentes = problemas com conteúdo
- Cuidado: ao utilizar a data de última atualização do portal_catalog pode tornar o cache em memória muito ineficiente



Demo

Obrigado

Fabiano Weimar dos Santos
xiru@xiru.org
Twitter @xiru