CHAPTER 1

L'application TinyBlog : présentation et modèle

Tout au long de ce projet, nous allons vous guider pour développer et enrichir une application Web pour gérer un ou plusieurs blogs (voir l'état final de l'application dans la Figure 1.1). Dans la suite, nous appellerons cette application: TinyBlog. L'idée est qu'un visiteur du site Web puisse voir les posts et que l'auteur du blog puisse se connecter sur le site pour administrer le blog c'est-à-dire ajouter, supprimer ou modifier des posts.

TinyBlog est une application à but pédagogique qui va vous montrer comment définir et déployer une application en utilisant Pharo/Seaside/Mongo ainsi que des frameworks disponibles en Pharo comme NeoJSON. Nous exposerons aussi comment exposer votre application via un serveur REST. Les solutions proposées dans ce tutoriel sont parfois non optimales afin de vous faire réagir et que vous puissiez proposer d'autres solutions et des améliorations. Notre objectif n'est pas d'être exhaustif. Nous montrons une façon de faire cependant nous invitons le lecteur à lire les références sur les autres chapitres, livres et tutoriaux Pharo afin d'approfondir son expertise et enrichir son application.

1.1 Installation de Pharo

Attention, dans ce cours nous supposons que vous utilisez Pharo 5.0 avec une image spéciale avec des packages pré-chargés. Dans TinyBlog, nous allons également utiliser des bibliothèques et frameworks supplémentaires pour le développement d'applications Web: Seaside, Magritte, Bootstrap, Voyage, VoyageMongo, ... Vous pouvez télécharger cette image (machine virtuelle et

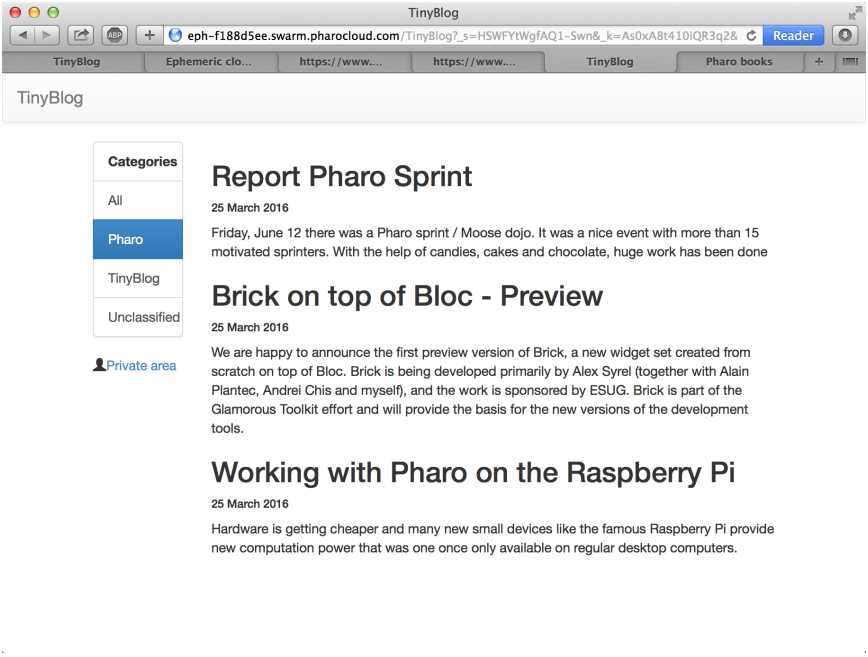

Figure 1.1 Final state of the TinyBlog application

image) à l'adresse : http://mooc.pharo.org/.

Dans le cadre de ce cours, nous vous suggérons de toujours utiliser l'image avec tous les packages web chargés que vous trouvez sur http://mooc.pharo.org car cette image contient tous les packages nécessaires.

1.2 Le modèle des posts

Le modèle de TinyBlog est extrêmement simple. Nous commencerons ici par la classe `TBPost`.

1.3 La classe TBPost

Dans la suite, nous préfixons tous les noms de classe par `TB` (pour TinyBlog). Il serait préférable que choisissiez un autre préfixe (par exemple `TBM`) afin de pouvoir ensuite charger la correction dans la même image Pharo et la comparer à votre propre classe.

Définissez la classe `TBPost` comme suit:

```
Object subclass: #TBPost
   instanceVariableNames: 'title text date category visible'
```

```
    classVariableNames: ''
    package: 'TinyBlog'
```

1.4 Description d'un post

Nous utilisons cinq variables d'instance pour décrire un post sur le blog.

Variable	Signification
title	Titre du post
text	Texte du post
date	Date de redaction
category	Rubrique contenant le post
visible	Post visible ou pas ?

Ces variables ont des méthodes accesseurs dans le protocole 'accessing'.

```
TBPost >> title
    ^ title
```

```
TBPost >> title: anObject
    title := anObject
```

```
TBPost >> text
    ^ text
```

```
TBPost >> text: anObject
    text := anObject
```

```
TBPost >> date
    ^ date
```

```
TBPost >> date: anObject
    date := anObject
```

```
TBPost >> visible
    ^ visible
```

```
TBPost >> visible: anObject
    visible := anObject
```

```
TBPost >> category
    ^ category
```

```
TBPost >> category: anObject
    category := anObject
```

1.5 Gérer la visibilité d'un post

Il faut avoir la possibilité d'indiquer qu'un post est visible ou pas. Il faut également pouvoir demander à un post s'il est visible. Les méthodes sont définies dans la protocole 'action'.

```
TBPost >> beVisible
   self visible: true
```

```
TBPost >> notVisible
   self visible: false
```

```
TBPost >> isVisible
   ^ self visible
```

1.6 Initialisation

La méthode `initialize` (protocole 'initialization') fixe la date à celle du jour et la visibilité à faux. L'utilisateur devra par la suite activer la visibilité ce qui permet de rédiger des brouillons et de publier lorsque le post est terminé. Un post est également rangé par défaut dans la catégorie 'Unclassified' que l'on définit au niveau classe. La méthode `unclassifiedTag` renvoie une valeur indiquant que le post n'est pas rangé dans une catégorie.

```
TBPost class >> unclassifiedTag
   ^ 'Unclassified'
```

Attention la méthode `unclassifiedTag` est définie au niveau de la classe (cliquer le bouton classe pour la définir). Les autres méthodes sont des méthodes d'instances c'est-à-dire qu'elles seront exécutées sur des instances de la classe `TBPost`.

```
TBPost >> initialize
   self category: TBPost unclassifiedTag.
   self date: Date today.
   self notVisible
```

Dans la solution proposée ci-dessus pour la méthode `initialize`, il serait préférable de ne pas faire une référence en dur à la classe `TBPost`. Nous traiterons ce point ultérieurement dans le MOOC.

1.7 Méthodes de création

Coté classe, on spécifie des méthodes pour faciliter la création de post appartenant ou pas à une catégorie.

```
TBPost class >> title: aTitle text: aText
   ^ self new
        title: aTitle;
        text: aText;
        yourself
```

```
TBPost class >> title: aTitle text: aText category: aCategory
   ^ (self title: aTitle text: aText)
            category: aCategory;
            yourself
```

1.8 Créer des posts

Vous pouvez dès maintenant créer des posts. Ouvrez l'outil Playground et exécuter l'expression suivante :

```
TBPost
  title: 'Welcome in TinyBlog'
  text: 'TinyBlog is a small blog engine made with Pharo.'
  category: 'TinyBlog'
```

Si vous inspectez le code ci-dessus (clique droit sur l'expression et "inspect"), vous allez obtenir un inspecteur sur l'objet post nouvellement créé.

1.9 Test

Nous définissons une méthode afin de savoir si un post est classé dans une catégorie.

```
TBPost >> isUnclassified
   ^ self category = TBPost unclassifiedTag
```

De même il serait préférable de ne pas faire une référence en dur à la classe `TBPost`.

1.10 Conclusion

Nous avons maintenant une première partie du modèle et nous vous suggèrons de sauver votre image Pharo (menu item 'save'). Lors de la prochaine séance nous vous montrerons comment sauver votre code (package) avec le système de version de Pharo.