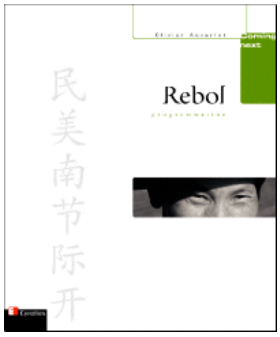
19

Étude de cas 3 : L'application IOS Brainstorm



Ce chapitre inédit aurait du figurer dans la réédition du livre "Programmation Rebol" (ISBN: 2-212-11017-0) publié par les éditions Eyrolles. Ce projet ayant été abandonné, le texte est versé au RDP (Rebol Documentation Project).

Auteur: Olivier Auverlot
Dernière modification: 14 mars 2005
Nombre de pages: 23

Brainstorm est une application destinée à Rebol/IOS. Fondée sur un modèle collaboratif, cette application consiste en un logiciel de mise en commun d'idées ouvert à l'ensemble des membres d'un groupe de travail. C'est un espace de discussion autorisant chacun à s'exprimer sans limitation, à introduire un thème de réflexion ou à proposer une idée.

Au sein du système d'information d'une entreprise, Brainstorm est l'outil idéal pour chercher la solution à un problème, concevoir de nouveaux produits ou consulter l'avis de collègues.

Le projet Brainstorm

Comme pour tout type de développement, la création d'une application destinée à Rebol/IOS débute par une étude de projet. Quels sont les objectifs à atteindre et les contraintes dont il faut tenir compte ? La rédaction d'une liste précise des fonctionnalités proposées par la reblet permet de cibler au mieux et au plus vite les choix techniques qui guideront l'organisation des données et l'écriture de l'application.

Brainstorm est un outil de communication et de productivité. Il permet à chacun de s'exprimer tout en archivant et en hiérarchisant scrupuleusement l'ensemble des messages en fonction d'un thème de réflexion donné. N'importe quel membre du groupe de travail, travaillant n'importe où et n'importe quand, peut

questionner les autres ou proposer des idées. Il s'agit de capitaliser les réflexions de chacun en n'écartant aucune hypothèse.

Parmi les applications standards de Rebol/IOS, Brainstorm vient compléter les reblets Messenger et Conference. Si la première permet à deux personnes de discuter d'un thème donné, elle ne permet pas à d'autres intervenants de donner leur avis. La seconde autorise la mise en commun d'idées mais ne dispose d'aucun moyen de structuration des informations recueillies. À mi-chemin entre ces deux produits, Brainstorm doit permettre d'avoir une vision claire, structurée et concise des thèmes de réflexion qui font l'actualité d'un groupe de travail.

Les contraintes de l'application

Brainstorm est une application qui manipule des informations mais sans être dédiée à une architecture matérielle ou logicielle précise. Elle doit être facilement installée, sans nécessiter d'autres opérations. Solution tout-en-un, fonctionnant directement avec n'importe quel serveur IOS, Brainstorm doit être indépendant pour son fonctionnement d'une base de données. Brainstorm tire parti des avantages et spécificités de Rebol/IOS. Les informations sont consultables, même si le client n'est pas connecté. L'ajout d'un thème de réflexion ou d'une idée n'est toutefois permis que lorsque le client communique avec le serveur IOS.

Brainstorm est une application dynamique. Les informations sont transmises en temps réel aux différents membres du groupe de travail. Lorsqu'un d'entre eux ajoute un thème ou propose une idée, les autres membres connectés et utilisant Brainstorm prennent connaissance des nouvelles données instantanément. Ce mécanisme favorise le dynamisme et la réactivité du groupe de travail.

Certaines informations sensibles pouvant être sujettes à d'éventuelles opérations d'espionnage, les données doivent pouvoir circuler sur les réseaux en toute sécurité. Le chiffrement systématique des données par Rebol/IOS assure un excellent niveau de confidentialité. Qu'ils soient sur le réseau local de leur entreprise ou qu'ils utilisent un fournisseur d'accès privé pour se connecter à leur serveur Rebol/Express, les membres du groupe de travail communiquent et échangent des idées avec sérénité.

Les fonctionnalités demandées

Brainstorm présente les données sous forme de liste, dont le contenu est hiérarchisé. Chaque thème de réflexion est suivi des idées proposées par les membres du groupe de travail. Les thèmes sont classés du plus récent au plus ancien. Pour chaque thème, en revanche, les idées sont classées selon leur ordre d'arrivée afin d'offrir un historique du projet permettant de suivre le cheminement des idées.

Les thèmes et idées appartiennent à l'utilisateur qui les crée. Ils ne peuvent donc être modifiés, et un utilisateur ne peut prendre possession de l'idée d'un autre membre du groupe de travail.

Dans Brainstorm, la saisie et la consultation doivent être les plus simples et les plus rapides possibles. L'utilisation du VID de Rebol/Link permet de limiter l'interface graphique au maximum de façon à mettre en avant le contenu.

Un utilisateur peut librement proposer un nouveau thème de réflexion. Celui-ci est ajouté à la liste et transmis aux autres membres du groupe. Pour chaque

thème, Brainstorm conserve le nom du rédacteur et la date à laquelle le thème a été ajouté. Ces informations sont accessibles aux autres utilisateurs, qui peuvent ensuite répondre en saisissant une nouvelle idée. Là aussi, Brainstorm conserve le nom de l'auteur et la date de création pour chaque nouvelle idée.

Afin de permettre la diffusion des différentes réflexions, Brainstorm propose l'exportation des thèmes et des idées associées en générant un document HTML.

Ce format présente les deux avantages suivants :

- La page HTML produite peut être diffusée sur l'intranet de l'entreprise ou même sur Internet si celle-ci dispose d'un site Web.
- Rebol/Link peut déclencher l'exécution d'un navigateur Web à l'aide du mot browse. Le document HTML est immédiatement consultable et peut être facilement imprimé afin d'en conserver une trace écrite.

Les écrans de saisie

Brainstorm n'a besoin que de deux écrans de saisie :

- l'écran principal de la reblet affichant les thèmes et le idées ;
- un écran de saisie permettant la création d'une nouvelle idée, ainsi que la consultation d'une idée déjà rédigée.

L'application Brainstorm est construite sur le même modèle ergonomique que les reblets livrées en standard avec Rebol/IOS. Priorité est donnée aux informations manipulées par l'utilisateur. Il n'existe donc pas de menu principal donnant accès à des rubriques ou à des sous-rubriques. Dès que la reblet est chargée, les données sont affichées, et l'utilisateur peut immédiatement travailler avec elles.

Les thèmes et les idées sont rassemblés dans une liste. La lecture de la liste est facilitée par deux icônes et un style de caractères différent. Les thèmes apparaissent en gras, et les idées sont décalées vers la droite afin de mettre en avant un notion de hiérarchie.

Comme l'illustre la figure 19-1, un champ de saisie et un bouton Nouveau permettent à l'utilisateur de saisir un nouveau thème, qui est instantanément ajouté à la liste et transmis aux autres utilisateurs.

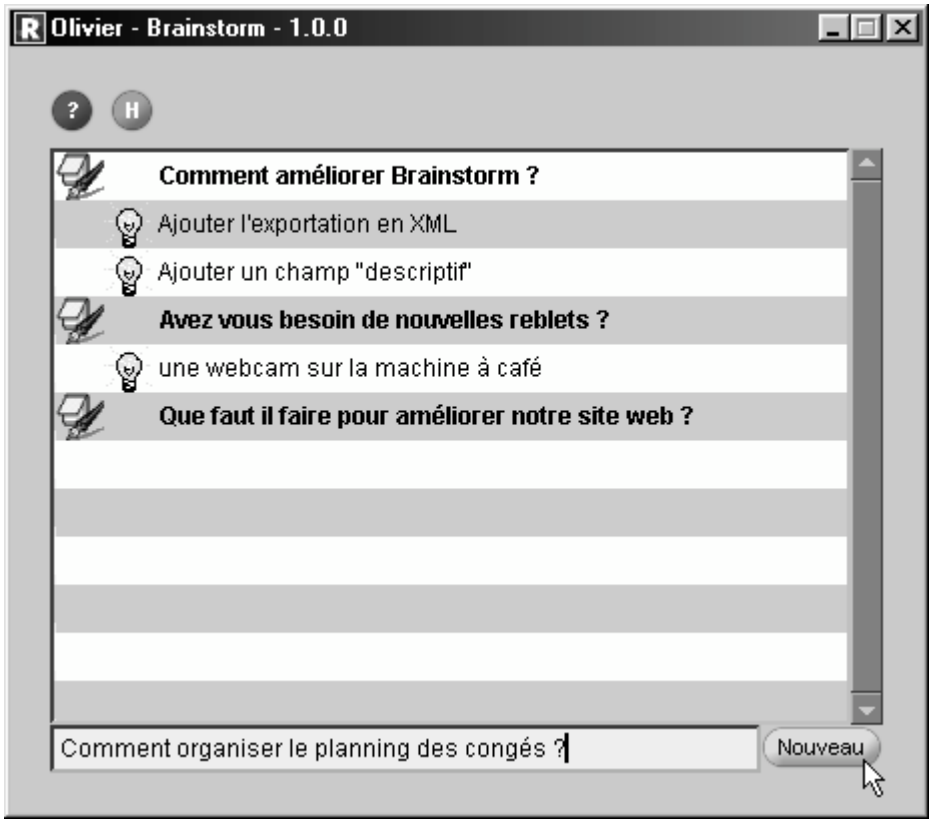


Figure 19-1 L'écran principal de la reblet Brainstorm

Pour répondre à un thème ou en consulter les détails, il suffit à un utilisateur de cliquer sur la ligne correspondante dans la liste. Un écran de saisie s'affiche, récapitulant le thème, le nom de son auteur et la date à laquelle il a été ajouté à la liste.

Comme illustré à la figure 19-2, l'utilisateur dispose d'un champ de saisie pour proposer une idée. Les boutons Fermer et Enregistrer sont respectivement destinés à revenir à l'écran principal et à enregistrer la nouvelle idée.

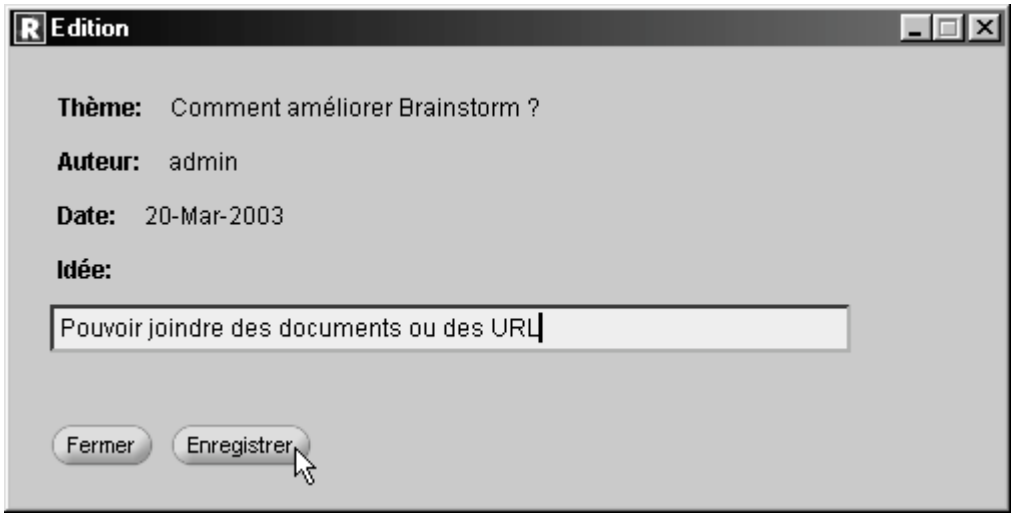


Figure 19-2 L'utilisateur propose une nouvelle idée

Lorsqu'un utilisateur clique sur une idée, il en obtient le descriptif complet, ainsi que celui du thème auquel l'idée est associée. Comme illustré à la figure 19-3, l'utilisateur sait qui a proposé l'idée et quand elle a été diffusée.

Figure 19-3 L'utilisateur consulte les détails d'une idée

Cet écran est identique à celui utilisé pour proposer une idée. Certains composants graphiques, tels que le bouton Enregistrer, sont cachés, tandis que d'autres sont affichés uniquement pour la consultation. Cette astuce évite la création de deux écrans pratiquement identiques et réduit sensiblement la taille du code.

Les écrans de l'application Brainstorm étant maintenant définis, il vous faut déterminer comment seront stockées les informations manipulées par l'application.

Les fichiers de données

Afin de ne pas être dépendante d'une architecture de communication particulière, l'application Brainstorm n'utilise pas une base de données externe telle que MySQL, Microsoft Access ou Oracle.

Le volume des informations manipulées par la reblet est insuffisant pour justifier l'usage de tels outils. Rebol dispose en standard de nombreuses fonctionnalités pour sauvegarder, consulter et trier les données. Il vous suffit d'exploiter ces capacités dans le domaine du traitement de données.

Le principe retenu pour Brainstorm est identique à celui utilisé par les reblets standards de Rebol/IOS. Les informations sont stockées dans des fichiers, lesquels sont rattachés à un fileset.

Cette solution permet de mettre en place simplement la synchronisation des données vers les postes clients. Lorsqu'un utilisateur propose un thème ou une

idée, les données sont transmises au serveur IOS. Celui-ci crée alors un fichier et le rattache à un fileset particulier destiné à conserver les informations.

Ce fileset étant synchronisé avec les postes clients, son contenu est transmis aux différents utilisateurs du serveur IOS. Ainsi, lorsqu'un utilisateur consulte les thèmes et les idées avec la reblet Brainstorm, il utilise les données qui sont présentes sur son poste local. Cette astuce permet la consultation off-line des données, puisque aucune connexion n'est nécessaire.

Pour ne pas tout mélanger sur le serveur et sur le client, les fichiers sont répartis selon leur nature dans deux répertoires, idees et themes. Les informations sont stockées dans une multitude de fichiers Rebol. Pour que chacun d'entre eux porte un nom différent, un compteur est utilisé. Comme illustré à la figure 19-4, chaque fichier se voit affecter un nom composé de son numéro et de l'extension **.r.** Le fileset étant répliqué sur le poste client, cette organisation des fichiers est strictement identique sur le client et sur le serveur.

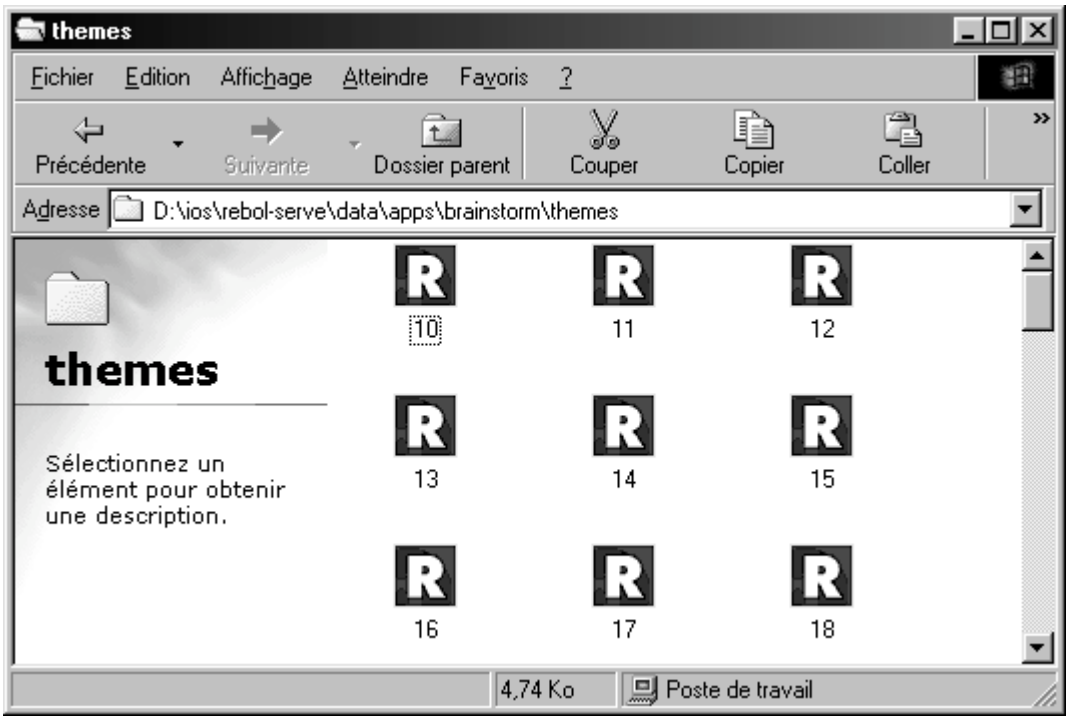


Figure 19-4 Les fichiers contenant les thèmes de réflexion

Reste maintenant à définir comment sont stockées les données dans chacun de ces fichiers. Plusieurs solutions s'offrent à vous :

- Utiliser de simples fichiers texte formatés, mais ceux-ci présentent peu de souplesse.
- Utiliser le format natif de Rebol et stocker les données à l'aide de séries. C'est une bonne solution, mais vous être tributaire de l'organisation des données. Si l'ordre des informations change, vous devez modifier le code exploitant ces fichiers.
- Utiliser XML, qui permet de dissocier les données de la structure du fichier les contenant. Vous pouvez ainsi aisément ajouter de nouvelles informations sans devoir réécrire des sections de code dans l'application Brainstorm.

- Utiliser un objet Rebol. Chaque thème et chaque idée sont stockés sous la forme d'un objet, chaque information correspondant à une propriété de l'objet.

Cette dernière solution, cent pour cent Rebol, est la plus élégante et la plus efficace en terme d'optimisation. Il est en effet inutile de convertir les données pour que celles-ci soient exploitables par Rebol. Le format est évolutif, et l'exploitation, la sauvegarde et le chargement des informations sont d'une simplicité extrême.

Le code suivant est le contenu du fichier **48.r** présent dans le répertoire themes et décrivant le thème « Comment améliorer Brainstorm ? ». Quatre propriétés sont utilisées afin d'indiquer le numéro du thème, son auteur, son intitulé et la date à laquelle il a été créé :

```
make object! [
    numero: 48
    auteur: "admin"
    texte: "Comment améliorer Brainstorm ?"
    date: 20-Mar-2003
]
```

Les idées sont également stockées à l'aide d'un objet Rebol. Le code suivant est le contenu du fichier **6.r** présent dans le répertoire idees et proposant l'idée « Ajouter l'exportation en XML ». La propriété theme indique le numéro du thème auquel cette idée est associée. L'objet contient également le nom de l'auteur et la date à laquelle l'idée a été soumise :

```
make object! [
    numero: 6
    auteur: "olivier"
    texte: "Ajouter l'exportation en XML"
    theme: 48
    date: 20-Mar-2003
]
```

L'étude du projet arrive maintenant à son terme. Les fonctionnalités, l'ergonomie et l'organisation des données sont définies, et vous pouvez passer à la construction de Brainstorm.

La première étape consiste à mettre en place votre projet sur le serveur IOS en générant les filesets de l'application.

Création des filesets

L'application Brainstorm utilise deux filesets distincts. L'un d'eux est destiné au stockage des fichiers contenant les thèmes et les idées. Le second contient la reblet qui sera exécutée par le poste client ainsi que la méthode POST contenant le code exécuté par le serveur Rebol/Express.

Cette séparation des données et du code de l'application permet de conserver les thèmes et les idées lorsqu'une nouvelle version de Brainstorm est installée sur le serveur. Dans le cas contraire, l'ensemble des informations saisies seraient perdues à chaque mise à jour du programme.

Stockage des données sur le serveur

Le fileset brainstorm-data a pour fonction de contenir les fichiers de données présents dans les répertoires themes et idees. Pour définir ce fileset, il vous suffit de créer le fichier install-brainstorm.r.

Ce script débute son exécution en désactivant le gestionnaire de sécurité de votre client Link afin d'éviter que celui-ci vous demande en permanence de confirmer les différentes actions nécessaires à la création du fileset sur le serveur Express. Le script vous demande ensuite si vous voulez conserver les données présentes éventuellement sur le serveur Express :

```
REBOL [
    title: "brainstorm"
    Type: 'link-app
]
secure none
keep-msgs: confirm "Conserver les données ?"
```

Si votre réponse est affirmative, l'installation du fileset brainstorm-data n'est pas réalisée, et l'ancienne version est conservée sur le serveur. Par contre, une réponse affirmative déclenche la création du fileset à l'aide du mot install-fileset. Les spécifications indiquées sont des plus simples puisque vous n'avez ici qu'à spécifier quels utilisateurs ont le droit de modifier la définition du fileset. Ici, seul l'administrateur de Rebol/IOS reçoit le droit de manipuler le fileset. Les tags users et groups n'étant pas définis, ce fileset est synchronisé avec l'intégralité des membres du groupe de travail :

```
if not keep-msgs [
    install-fileset [
        fileset: 'brainstorm-data
        tags: [
            access: [properties: rights: change: delete: ["admin"]]
        ]
    ]
]
```

L'espace de stockage des fichiers sur le serveur est maintenant défini. Vous pouvez donc créer un second fileset, destiné à recevoir le code de l'application.

Rédaction du fileset Brainstorm

Le code de la reblet et celui destiné au serveur Express sont placés dans le fileset brainstorm. Pour le définir, vous n'avez pas besoin de créer un second fichier d'installation. Il vous suffit d'utiliser le fichier install-brainstorm.r en le complétant. Pour séparer les définitions de deux filesets et exécuter tour à tour leurs procédures d'installation, il vous faut insérer le mot unview. Ainsi, lorsque vous installerez l'application, l'ensemble des éléments sera créé en une seule étape.

La définition du fileset brainstorm est un peu plus complexe que celle du fileset brainstorm-data. Les tags users et groups sont absents afin de ne pas limiter l'utilisation de l'application à une liste d'utilisateurs et de groupes. Au niveau des droits de modification du fileset, seul l'administrateur est autorisé à changer la valeur des tags. En initialisant la directive POST avec la valeur 'all, l'ensemble des membres du groupe de travail est autorisé à transférer des informations en direction de la section de code exécutée par le serveur Express :

```
access: [
    properties: rights:
    desktop: folders:
    icons: serve-code:
    change: delete: ["admin"]
```

```
    post: 'all
]
```

Pour apparaître sur le bureau de Rebol/Link, l'application Brainstorm a besoin d'une icône. Celle-ci se nomme brainstorm.png. Elle a pour dimensions 48 × 48 pixels. L'icône est stockée dans les répertoires desktop/icons/ du serveur Express et du client Link. Le tag folder indique que cette icône apparaît dans le classeur Applications du bureau Link. Le nom de l'application est "Brainstorm".

Comme illustré à la figure 19-5, le passage de la souris sur l'icône de la reblet provoque l'affichage du texte « Recherche d'idées » sur la ligne d'information située au bas du bureau :

```
icons: [
    [
        name: "Brainstorm"
        item: %apps/brainstorm/brainstorm.r
        folder: %apps/
        image: %desktop/icons/brainstorm.png
        info: "Recherche d'idées"
    ]
]
```

Figure 19-5 La reblet Brainstorm est installée dans le classeur Applications

Il vous faut maintenant définir la liste des fichiers qui seront transférés dans le fileset brainstorm. Pas moins de quatre fichiers sont nécessaires au bon fonctionnement de l'application. Le principal élément est le script de la reblet destinée aux postes clients. Celui-ci est contenu dans le fichier brainstorm.r et est installé dans le répertoire apps/brainstorm/.

Vous devez transmettre trois éléments graphiques au serveur :

- l'icône destinée au bureau de Link ;
- une image idee.png mesurant 24 × 24 pixels destinée à être affichée dans la liste faisant partie de la reblet ;
- une image theme.png mesurant 24 × 24 pixels également destinée à être utilisée avec le composant graphique de type liste présent dans la reblet.

```
files: [
    [%apps/brainstorm/brainstorm.r %brainstorm.r]
    [%desktop/icons/brainstorm.png %brainstorm.png ]
    [%apps/brainstorm/images/idee.png %images/idee.png ]
    [%apps/brainstorm/images/theme.png %images/theme.png ]
]
```

Les fichiers theme.png et idee.png sont tous deux placés dans le répertoire apps/brainstorm/images/. Comme illustré à la figure 19-6, ces fichiers présentent la particularité d'utiliser un fond rouge, dont la valeur RGB est 255.0.0. Cette couleur permettra par la suite d'appliquer un effet de transparence.

Figure 19-6 Les éléments graphiques utilisés pour Brainstorm

La description des propriétés des deux filesets utilisés par l'application Brainstorm est pratiquement terminé. Il vous faut maintenant insérer dans le fileset brainstorm la méthode POST nécessaire au bon fonctionnement de l'application.

Les commandes transmises par le client

La méthode POST incluse dans le fileset brainstorm a pour finalité de créer sur le serveur Express les fichiers contenant les thèmes et les idées transmis par les clients Link. Le fait d'assembler et stocker ces fichiers sur le serveur présente un énorme avantage, puisqu'il est de la sorte extrêmement aisé de les rattacher directement à un fileset afin qu'ils soient synchronisés.

Avant d'écrire la méthode POST proprement dite, il vous faut réfléchir aux différentes requêtes que les clients sont susceptibles d'émettre en direction du

serveur Express. Il vous faut également déterminer le format utilisé pour l'échange des données.

La reblet Brainstorm peut envoyer deux types de requêtes à la méthode POST : lui signifier la création d'un nouveau thème ou celle d'une nouvelle idée. Pour indiquer à la méthode POST qu'il s'agit de l'un ou de l'autre cas, le premier élément reçu dans la bloc message de la méthode POST est réservé afin d'indiquer le type de l'opération. Celui ci peut prendre l'une des valeurs suivants :

- nv-theme, s'il s'agit d'un nouveau thème.
- nv-idee, s'il agit d'une nouvelle idée.

Selon que la méthode post reçoit l'une de ces deux commandes, elle s'attend à trouver différentes informations dans la suite du bloc message.

Dans le cas d'un nouveau thème, les informations reçues sont :

- le nom de l'auteur ;
- l'intitulé du thème.

S'il s'agit d'une nouvelle idée, une valeur supplémentaire est reçue. Les informations présentes sont :

- le nom de l'auteur ;
- l'intitulé de l'idée ;
- le numéro du thème auquel est associée l'idée.

Le format des messages émis par la reblet en direction du fileset brainstorm est maintenant défini. Vous pouvez passer à l'écriture du code de la méthode POST.

Conception de la méthode POST

La méthode POST débute son exécution en sécurisant l'évaluation du code serveur. De cette façon, si un problème survient, celui-ci ne remet pas en cause la stabilité du serveur Express.

Le code initialise la variable num avec la valeur 0. Par défaut, c'est ce numéro qui sera utilisé pour nommer un nouveau fichier. La variable base est initialisée avec le répertoire apps/brainstorm/ afin de simplifier la rédaction des chemins d'accès et, éventuellement, de faciliter le déplacement de l'application dans un autre répertoire.

La méthode POST prélève ensuite la première valeur présente dans le bloc message et adapte son comportement selon que le client demande la création d'un nouveau thème ou celle d'une nouvelle idée.

Avant de sauvegarder l'objet dans le répertoire apps/brainstorm/themes/ ou apps/brainstorm/idees, la méthode POST doit rechercher un numéro de fichier. Pour cela, elle utilise deux fichiers non synchronisés, apps/brainstorm/themes.r et apps/brainstorm/idee.r, contenant chacun un chiffre. Il est important de retenir que ces deux compteurs ne sont jamais transmis aux clients car ils ne sont rattachés à aucun fileset.

La méthode POST vérifie tout d'abord l'existence de l'un de ces fichiers à l'aide du mot data-exists?. Si le fichier est disponible, son contenu est affecté à la variable num en utilisant load-data. Dans le cas contraire, la variable num conserve la valeur 0 affectée au début du code. Le compteur peut maintenant être incrémenté d'une unité. Sa nouvelle valeur est sauvegardée à l'aide du mot write-data.

La création du fichier contenant le thème ou l'idée se déroule en deux étapes. Tout d'abord, les différentes informations reçues du client sont assemblées dans un objet. Il vous faut ensuite sauvegarder l'objet obtenu dans un fichier en le rattachant au fileset brainstorm-data. Cette opération est réalisée à l'aide du mot add-file.

Une fois le code de la méthode POST rédigé, le contenu du fichier install-brainstorm.r est le suivant :

```
REBOL [
    title: "brainstorm"
    Type: 'link-app
]
secure none
keep-msgs: confirm "Conserver les données ?"
if not keep-msgs [
    install-fileset [
        fileset: 'brainstorm-data
        tags: [
            access: [properties: rights: change: delete: ["admin"]]
        ]
    ]
]
unview
install-fileset [
    fileset: 'brainstorm
    tags: [
        access: [
            properties: rights:
            desktop: folders:
            icons: serve-code:
            change: delete: ["admin"]
            post: 'all
        ]
        icons: [
            [
                name: "Brainstorm"
                item: %apps/brainstorm/brainstorm.r
                folder: %apps/
                image: %desktop/icons/brainstorm.png
                info: "Recherche d'idées"
            ]
        ]
    ]
    files: [
        [%apps/brainstorm/brainstorm.r %brainstorm.r]
        [%desktop/icons/brainstorm.png %brainstorm.png ]
        [%apps/brainstorm/images/idee.png %images/idee.png ]
        [%apps/brainstorm/images/theme.png %images/theme.png ]
    ]

    post-locals: [ num-theme num-idee base theme idee err ]
    post-func: [
        if error? err: try [
            num: 0
            base: %apps/brainstorm/
            switch message/1 [
                nv-theme [
                    if data-exists? base/themes.r [
                        num: load-data base/themes.r
                    ]
                    write-data base/themes.r mold num + 1
```

```
                    fichier: join num ".r"
                    theme: context compose [
                        numero: (num)
                        auteur: (message/2)
                        texte: (message/3)
                        date: (now/date)
                    ]
                    add-file 'brainstorm-data base/themes/:fichier
(compress mold theme)
                ]
                nv-idee [
                    if data-exists? base/idees.r [
                        num: load-data base/idees.r
                    ]
                    write-data base/idees.r mold num + 1
                    fichier: join num ".r"
                    idee: context compose [
                        numero: (num)
                        auteur: (message/2)
                        texte: (message/3)
                        theme: (message/4)
                        date: (now/date)
                    ]
                    add-file 'brainstorm-data base/idees/:fichier
(compress mold idee)
                ]
            ]
        ] [ print mold disarm :err ]
    ]
]
```

Vous venez de terminer la définition des deux filesets utilisés pour l'application Brainstorm, et vous avez rédigé le code de la méthode POST exécutée par le serveur Express. Il ne vous manque plus que le dernier élément du projet, c'est-à-dire la reblet destinée au client.

Écriture de la reblet

Pour rédiger la reblet Brainstorm, vous devez créer un fichier nommé brainstorm.r et le placer dans le même répertoire que celui contenant le script d'installation install-brainstorm.r. Le code que vous allez écrire est celui destiné à être exécuté par le client Link et donc par le poste client.

Les trois fonctions de ce script sont d'afficher la liste des thèmes et idées associées, de permettre la création d'un nouveau thème ou la proposition d'une nouvelle idée et de générer un rapport au format HTML.

Initialisation de la reblet

Les premières lignes de la reblet sont des plus classiques. Il s'agit d'un bloc d'en-tête qui, en addition aux renseignements décrivant le script, contient le type 'link-app afin de signifier que ce code est destiné au client Link :

```
REBOL [
    author: "Olivier Auverlot"
    type: 'link-app
    version: 1.0.0
]
```

Les images idee.png et theme.png font partie du fileset brainstorm. Elles sont synchronisées et téléchargées sur le poste client en même temps que la reblet. Vous pouvez simplement les utiliser à l'aide du mot load et les affecter aux variables img1 et img2 :

```
img1: load %images/idee.png
img2: load %images/theme.png
```

Pour générer une ligne de titre destinée à l'écran principal de la reblet, vous pouvez d'ores et déjà initialiser la variable ligne-titre contenant le nom du serveur Express utilisé, ainsi que celui de la reblet et son numéro de version :

```
ligne-titre: reform [
    user-prefs/express "-" "Brainstorm"
    "-" system/script/header/version
]
```

Afin de prévenir toute erreur d'exécution sur le serveur, vous pouvez mettre en place la détection de l'événement serve-error. La fonction traitant cet événement a pour mission d'afficher la nature de l'erreur et de stopper l'exécution de l'application :

```
erreur: func [ event data ] [
    alert join "Erreur sur le serveur: " (mold event)
quit
]

insert-notify 'serve-error none :erreur
```

Lorsqu'un utilisateur modifie le fileset brainstorm-data en ajoutant un nouveau thème ou en proposant une nouvelle idée, la reblet doit être capable de détecter cette action afin de mettre à jour automatiquement l'affichage des informations. Pour cela, vous devez définir une gestion de l'événement fileset-downloaded. Lorsque cet événement est détecté, la fonction registre-ok est appelée. Celle-ci utilise la commande GET du dialecte d'administration de Rebol/Link afin de consulter la liste des fichiers présents dans le fileset brainstorm-data :

```
registre-ok: func [ event data ] [
    ; on récupère les thèmes et les idées
    insert-notify 'get none :lire-fichiers
    send-link 'get 'app-files 'brainstorm-data
]

insert-notify 'fileset-downloaded 'brainstorm-data :registre-ok
```

C'est cette méthode qui est utilisée au démarrage de la reblet pour lire les fichiers contenant les thèmes et idées déjà présents sur le disque dur du poste client :

```
insert-notify 'get none :lire-fichiers
send-link 'get 'app-files 'brainstorm-data
do-events
```

Il vous faut maintenant rédiger la fonction lire-fichiers afin d'initier correctement les composants graphiques présents sur l'écran principal de la reblet.

Lecture des fichiers de données

La fonction lire-fichiers est utilisée pour lire les fichiers contenant les thèmes et les idées. Elle a également pour objectif de classer ces informations afin de rattacher chaque idée à son thème et, ainsi, de faciliter l'affichage des données dans un composant liste du VID.

Vous avez donc besoin de trois séries, themes, idees et items, afin de stocker les données en mémoire. La valeur de la variable cnt est définie à 0. Vous la retrouverez plus tard lors de la construction du composant liste nommé table.

Vous devez également désactiver la surveillance de l'événement GET, devenu inutile :

```
lire-fichiers: func [ event data /local chemin obj ] [
    themes: copy []
    idees: copy []
    items: copy []
    cnt: 0
    remove-notify 'get 'brainstorm-data
```

La fonction lire-fichiers a reçu dans son paramètre data la liste des fichiers présents dans le fileset brainstorm-data. Pour déterminer s'il s'agit d'un thème ou d'une idée, il suffit d'étudier le chemin d'accès de chaque fichier. Le chemin est extrait du nom du fichier à l'aide du mot split-path. Selon l'un ou l'autre cas, l'objet présent dans le fichier est chargé en mémoire, et son contenu est ajouté à la liste themes ou à la liste idees :

```
foreach fichier data [
    chemin: split-path fichier
    switch chemin/1 [
        %apps/brainstorm/themes/ [
            obj: do load join link-root fichier
            append themes reduce [
                obj/numero
                obj/auteur
                obj/texte
                obj/date
            ]
        ]
        %apps/brainstorm/idees/ [
            obj: do load join link-root fichier
            append idees reduce [
                obj/numero
                obj/auteur
                obj/texte
                obj/theme
                obj/date
            ]
        ]
    ]
]
```

Il vous faut maintenant trier ces informations. La liste contenant les thèmes est triée de quatre en quatre éléments, selon la date de création et par ordre décroissant. Les thèmes les plus récents viennent donc en tête de la série :

```
sort/skip/compare/reverse themes 4 4
```

La série contenant les idées est triée de cinq en cinq éléments, selon la valeur de la date de création par ordre croissant :

```
sort/skip/compare idees 5 5
```

Pour l'instant, vous avez deux listes triées. Pour les afficher dans un composant graphique de type liste, vous devez les fusionner. Pour cela, vous utilisez la liste items, dans laquelle chaque thème est suivi des idées qui lui sont associées :

```
forskip themes 4 [
    append items reduce [ themes/1 img2 none themes/3 ]
    idees: head idees
    forskip idees 5 [
        if idees/4 = themes/1 [
            append items reduce [ idees/1 none img1 idees/3 ]
        ]
    ]
]
```

Il ne vous reste qu'à initialiser les composants graphiques présents dans l'écran principal. Un ascenseur est nécessaire si le nombre de thèmes et d'idées devient

trop important pour être affiché dans le composant liste. Vous devez donc ajuster certains de ses paramètres.

La variable lst-high détermine le nombre de lignes pouvant être affichées dans la liste en divisant la hauteur de la liste par la hauteur de chaque ligne. La hauteur du curseur présent dans l'ascenseur est également initialisée en fonction du nombre d'éléments devant être affichés dans la liste :

```
lst-high: table/size/y / table/subface/size/y
sld/redrag table/size/y / max 1 table/subface/size/y * ((length?
items) / 4)
```

Le focus est donné au composant champ-theme pour permettre la saisie d'un nouveau thème. Un test utilisant le mot viewed? est utilisé pour afficher l'écran principal. Ce mot retourne une valeur booléenne selon qu'un élément graphique est ou non affiché. Dans le cas d'un simple rafraîchissement des données, tel que celui déclenché par un événement fileset-downloaded, il suffit d'afficher à nouveau le composant de type liste nommé table, qui affiche les thèmes et les idées :

```
focus champ-theme
either not viewed? main [
    view/new/title center-face main ligne-titre
] [ show table ]
```

Tout est maintenant prêt pour que les informations s'affichent à l'écran et que l'utilisateur les manipule à l'aide de l'écran principal de votre reblet.

Définition de l'écran principal

Comme illustré à la figure 19-7, l'écran principal de la reblet brainstorm comporte les six éléments suivants :

- trois boutons ;
- une liste nommée table ;
- un ascenseur nommé sld ;
- un champ de saisie nommé champ-theme.

Figure 19-7 L'écran principal de la reblet Brainstorm

En haut et à gauche du layout, deux petits boutons ronds permettent d'obtenir des informations sur la reblet et de déclencher la génération d'un rapport en HTML grâce à l'appel de la fonction gen-html. Il s'agit d'un nouveau style de bouton introduit par Rebol/Link, dont le nom est btn-help :

```
main: layout [
across
btn-help [
    alert reform [ "Brainstorm -" system/script/header/version ]
]
btn-help "H" gold - 40 [
    if confirm "Générer un rapport en HTML ?" [ gen-html ]
]
return
```

Juste en dessous, se trouve la liste table. Chacune des lignes de cette liste comportent quatre colonnes composées des éléments suivants :

- Un texte invisible mesurant un pixel correspondant au premier élément de chaque bloc de quatre valeurs présent dans la liste item. Il s'agit de la clé du thème ou de l'idée présent sur la ligne.
- L'icône img1 indiquant que la ligne est un thème.
- L'icône img2 indiquant que la ligne est une idée.
- L'intitulé du thème ou de l'idée.

Le remplissage de cette ligne est assuré par le bloc passé en paramètre à l'attribut supply. Selon que la ligne porte un numéro pair ou impair, une couleur lui est attribuée. Selon la position courante retournée par la propriété count de la liste, une valeur de la liste items est prélevée et affectée à la variable information. Si la valeur de cette variable est différente de none, la colonne de la ligne courante est initialisée avec cette donnée. La propriété index retourne le numéro de la colonne active, c'est-à-dire en cours de construction.

Les règles appliquées pour le remplissage de la liste `table` sont les suivantes :

- Si la colonne courante est la première, la variable cle est initialisée avec le numéro du thème ou de l'idée.
- Si la colonne courante est la seconde, l'image contenue dans la colonne est initialisée avec celle présente dans la variable information. Si cette dernière contient une image, la variable theme prend la valeur true afin d'indiquer que cette ligne correspond à un thème.
- Si la colonne courante est la troisième, l'image contenue dans la colonne est initialisée avec celle présente dans la variable information. Si cette dernière contient une image, la variable theme prend la valeur false afin d'indiquer que cette ligne correspond à une idée.
- Si la colonne courante est la quatrième, le texte présent dans la colonne est initialisé avec celui présent dans la variable information. Le texte est centré verticalement, et la hauteur de la ligne est définie à 24 pixels, soit la même hauteur que celle des images. Si la variable theme a la valeur true, le texte est écrit en gras. La propriété face/data, qui est une valeur associée à l'élément graphique, est initialisée avec un bloc composé de deux éléments précisant la nature de la ligne (thème ou idée) et la clé de l'élément affiché :

```
space 0
table: list 400x288 [
    origin 0 space 0x0 across
    txt 1
    image 24x24 effect [ key 255.0.0 ]
    image 24x24 effect [ key 255.0.0 ]
    txt 352 [
        if not none? face/data [
            editer face/data
        ]
    ]
] supply [
    count: count + cnt
    items: head items
    either even? count [
        face/color: ivory - 50.50.50
    ] [ face/color: ivory ]

    information: pick items (index + ((count - 1) * 4))
    either not any [ (information = 'none) (none? information) ] [
        switch/default index [
            1 [ cle: information ]
            2 [
                face/image: copy information
                if not none? face/image [ theme: true ]
            ]
            3 [
                face/image: copy information
                if not none? face/image [ theme: false ]
            ]
        ] [
            face/text: copy to-string information
            face/font/valign: 'middle
            either theme = true [
```

```
                face/data: copy reduce [ 'theme cle ]
                face/font/style: 'bold
            ] [
                face/data: copy reduce [ 'idee cle ]
                face/font/style: none
            ]
            face/size/y: 24
        ]
    ] [
        face/image: none
        face/text: copy ""
        face/data: none
    ]
]
```

Placé juste à coté du composant graphique table, un ascenseur permet à l'utilisateur de parcourir les informations contenues dans la liste. Lorsqu'il est utilisé, il modifie la valeur de la variable cnt afin de sélectionner la première ligne affichée dans la liste :

```
sld: scroller 16x288 [
    c: max 0 to-integer ((length? items) / 4) - lst-high * value
    if c <> cnt [
        cnt: c
        show table
    ]
] return
```

Tout en bas du layout, se trouve un champ de saisie nommé champ-theme et un bouton intitulé "Nouveau". Ils permettent à l'utilisateur de saisir un nouveau thème et de l'envoyer au serveur. Pour cela, la reblet communique avec la méthode POST placée dans le fileset brainstorm-data. Après avoir vérifié que l'utilisateur est bien connecté au serveur Express, elle transmet un bloc contenant les trois éléments suivants :

- La valeur 'nv-theme indiquant à la méthode POST que les données reçues concernent la création d'un thème.
- Le nom de l'utilisateur.
- Le texte saisi dans le champ theme-champ.

```
champ-theme: field 355
btn "Nouveau" [
    either connected? [
        either (length? trim champ-theme/text) > 0 [
            insert-notify 'post-reply 'brainstorm :nv-theme
            send-server 'post reduce [
                'brainstorm
                'nv-theme
                user-prefs/name
                (trim champ-theme/text)
            ]
        ] [ alert "Vous devez saisir un thème." ]
    ] [ alert "Vous n'êtes pas connecté au serveur." ]
]
```

Une fois les données envoyées au serveur, il n'est pas nécessaire de mettre à jour vous-même le contenu de la liste table. En effet, une fois que le serveur Express a reçu les données et qu'il a créé un nouveau fichier dans le fileset brainstorm-data, ce fichier est automatiquement synchronisé avec les postes clients. À réception de ce fichier, un événement fileset-downloaded est déclenché, et la liste est immédiatement mise à jour. Le gestionnaire d'événements nv-theme n'a donc qu'un rôle restreint, consistant à désactiver la surveillance de l'événement post-reply et à vider le contenu du champ theme-champ :

```
nv-theme: func [ event data ] [
    remove-notify 'post-reply 'brainstorm
    champ-theme/text: copy ""
    show champ-theme
    focus champ-theme
]
```

La création d'un nouveau thème est maintenant terminée. Vous pouvez mettre en place les éléments permettant de créer une nouvelle idée ou de consulter un thème ou une idée existante.

Créer ou consulter un thème ou une idée

Lorsque l'utilisateur clique sur une ligne de la liste table, celle-ci appelle une fonction nommée editer et lui passe en paramètre la valeur de la propriété face/data associée à l'élément courant de la liste. À l'aide du premier élément de ce bloc, la fonction editer détermine si la clé numérique, qui est le second élément du bloc, concerne un thème ou une idée et adapte son comportement en conséquence. S'il s'agit d'un idée, l'utilisateur désire consulter des informations complémentaires, telles que la date de proposition et le nom de l'auteur. S'il s'agit d'un thème, l'utilisateur désire proposer une idée.

Dans l'un ou l'autre cas, la fonction editer utilise un même layout pour constituer la boîte de dialogue. Certains éléments sont désactivés ou cachés selon les besoins.

Le bouton btn-enregistrer, par exemple, n'apparaît que si l'utilisateur désire proposer une nouvelle idée et qu'il a donc précédemment cliqué sur un thème. Pour transmettre l'idée saisie dans le champ champ-idee, la reblet utilise la méthode POST présente dans le fileset brainstorm. Les données transmises contiennent le type du message ('nv-idee), le nom de l'utilisateur, le texte décrivant l'idée et la clé numérique du thème auquel l'idée doit être associée :

```
edition: layout [
    style rubrique text 'bold
    style info text black
    across
    rubrique "Thème:" lbl-theme: info 400 return
    rubrique "Auteur:" lbl-theme-auteur: info 150 return
    rubrique "Date:" lbl-theme-date: info 200 return
    rubrique "Idée:" return
    champ-idee: field 400 return
    rub-idee-date: rubrique "Proposée le:" lbl-idee-date: info 100
    rub-idee-par: rubrique "Par:" lbl-idee-auteur: info 150 return
    btn-fermer: btn "Fermer" [ unview edition ]
    btn-enregistrer: btn "Enregistrer" [
        either connected? [
            either (length? trim champ-idee/text) > 0 [
                insert-notify 'post-reply 'brainstorm :nv-idee
                send-server 'post reduce [
                    'brainstorm
                    'nv-idee
                    user-prefs/name
                    (trim champ-idee/text)
                    cle-theme
                ]
            ] [ alert "Vous devez saisir un thème." ]
        ] [ alert "Vous n'êtes pas connecté au serveur." ]
    ]
]
```

Une fois que la méthode POST a terminé son travail, un événement post-reply est intercepté, et la fonction nv-idee est appelée. Cette fonction peut désactiver la surveillance de l'événement et faire disparaître le layout edition de l'écran. Comme pour la création d'un nouveau thème, le rafraîchissement des données

dans la liste table est assuré automatiquement par la surveillance de l'événement fileset-downloaded :

```
nv-idee: func [ event data ] [
    remove-notify 'post-reply 'brainstorm
    champ-idee/text: copy ""
    unview edition
]
```

La fonction editer affecte la clé du thème ou de l'idée manipulé à la variable globale cle-theme. Deux cas peuvent maintenant se présenter :

- L'élément manipulé est un thème. L'objet qui le décrit est chargé en mémoire afin d'initialiser les champs présents dans le layout edition. Le bouton "Enregistrer" est affiché, et certains champs inutiles sont cachés.
- L'élément manipulé est une idée. À l'aide de la valeur contenue dans cle-theme, le script recherche et charge en mémoire l'objet décrivant le thème concerné. Le bouton "Enregistrer" et les champs inutiles sont cachés. Les informations sur le thème et l'idée sont affectées à ceux restés visibles.

```
editer: function [ element ] [ obj-theme obj-idee ] [
    cle-theme: element/2
    either element/1 = 'theme [
        ; ajout d'une idée
        obj-theme: do load join link-root reduce [
            %apps/brainstorm/themes/ cle-theme ".r"
        ]
        btn-enregistrer/show?: true
        rub-idee-date/show?: false
        rub-idee-par/show?: false
        lbl-idee-date/show?: false
        lbl-idee-auteur/show?: false
        champ-idee/text: copy ""
        focus champ-idee
    ] [
        ; lecture d'une idée
        idees: head idees
        forskip idees 5 [
            if idees/1 = cle-theme [
                obj-theme: do load join link-root reduce [
                    %apps/brainstorm/themes/ idees/4 ".r"
                ]
                break
            ]
        ]
        btn-enregistrer/show?: false
        rub-idee-date/show?: true
        rub-idee-par/show?: true
        lbl-idee-date/show?: true
        lbl-idee-auteur/show?: true
        champ-idee/text: copy idees/3
        lbl-idee-auteur/text: copy idees/2
        lbl-idee-date/text: copy to-string idees/5
    ]
    lbl-theme/text: copy obj-theme/texte
    lbl-theme-auteur/text: copy obj-theme/auteur
    lbl-theme-date/text: copy to-string obj-theme/date
    view/new/title center-face edition "Edition"
]
```

Vous voici presque au terme de votre première application pour Rebol/IOS. Pour la terminer, il ne vous reste qu'à générer un rapport en HTML.

Générer un rapport en HTML

La production d'un document au format HTML ne pose aucune difficulté pour Rebol. Le document est construit par l'ajout de données dans la variable locale html. Le script exploite les données présentent dans la liste items. Sur la page HTML, chaque thème est suivi de la liste des idées qui lui sont associées. Grâce à la balise <ul>, qui décale vers la droite l'affichage d'un texte, l'organisation hiérarchique est respectée :

```
gen-html: has [ html emit ] [
    html: copy ""
    emit: func [ val ] [ repend html val ]
    emit [ <html><body><h2>"Brainstorm " now/date </h2> ]
    items: head items
    forskip items 4 [
        either not none? items/2 [
            if (index? items) <> 1 [
                emit [ </ul> ]
            ]
            emit [ <b> items/4 </b><br><ul> ]
        ] [
            emit [ items/4 <br> ]
        ]
    ]
    emit [ </body></html> ]
    write %temp-out.html html
    browse %temp-out.html
]
```

Au terme de la génération, le contenu de la variable html est sauvegardé dans un fichier nommé temp-out.html sur le poste client. Comme illustré à la figure 19-8, il suffit d'utiliser le mot browse pour utiliser le navigateur par défaut de la machine hôte permettant la visualisation du contenu de ce fichier.

Figure 19-8 Affichage du rapport HTML

Résumé

Vous venez de développer une application utilisant les spécificités de Rebol/IOS et permettant la proposition et le partage d'idées au sein d'un groupe de travail.

Vous avez défini deux filesets destinés au stockage de l'application et des fichiers de données sur le serveur. À l'aide d'une méthode POST, vous avez établi une communication entre le client Link et le serveur Express afin de répartir les traitements entre les deux partis.

Vous avez utilisé plusieurs types d'événements permettant la surveillance de l'activité du serveur Express. L'usage de fileset-downloaded vous a montré la simplicité de conception d'une application IOS dans laquelle la modification d'une donnée partagée implique que les autres utilisateurs connectés en soient immédiatement informés.