JAVAMAIL API - FOLDER MANAGEMENT

So far, we've worked in our previous chapters mostly with the INBOX folder. This is the default folder in which most mail resides. Some systems might call it as INBOX and some other might call it by some other name. But,you can always access it from the JavaMail API using the name INBOX.

The JavaMail API represents folders as instances of the abstract Folder class:

```
public abstract class Folder extends Object
```

This class declares methods for requesting named folders from servers, deleting messages from folders, searching for particular messages in folders, listing the messages in a folder, and so forth.

Opening a Folder

We can't create a folder directly as the only constructor in the *Folder* class is *protected*. We can get a *Folder* from:

- a Session
- a Store
- or another Folder

All the above classes have a similar getFolder method with similar signature:

```
public abstract Folder getFolder(String name) throws MessagingException
```

Some of the methods which help in getting the *Folder* object are:

Method	Description
boolean *exists*	Checks if the folder really exists. Use this method before getting the Folder object.
abstract void *open*intmode	When you get a *Folder*, its closed. Use this method to open it. *mode* can be Folder.READ_ONLY or Folder.READ_WRITE.
abstract boolean *isOpen*	This method returns *true* if the folder is open, *false* if it's closed
abstract void *close*booleanexpunge	Closes the folder. If the *expunge* argument is *true*, any deleted messages in the folder are deleted from the actual file on the server. Otherwise, they're simply marked as *deleted*, but the messages can still be undeleted.

Basic Folder Info

Following are some of the methods in Folder class which return basic information about a folder:

Method	Description
abstract String *getName*	Returns the name of the folder, such as "TutorialsPoint Mail"
abstract String *getFullName*	Returns the complete hierarchical name from the root

	such as “books/Manisha/TutorialsPoint Mail”.
URLName *getURLName*	Return a URLName representing this folder.
abstract Folder *getParent*	Returns the name of the folder that contains this folder i.e the parent folder. E.g "Manisha" from the previous "TutorialsPoint Mail" example.
abstract int *getType*	Returns an int indicating whether the folder can contain messages and/or other folders.
int *getMode*	It returns one of the two named constants Folder.READ_ONLY or Folder.READ_WRITE or -1 when the mode is unknown.
Store *getStore*	Returns the Store object from which this folder was retrieved.
abstract char *getSeparator*	Return the delimiter character that separates this Folder's pathname from the names of immediate subfolders.

Managing Folder

Following are some of the methods which help manage the Folder:

Method	Description
abstract boolean *create*inttype	This creates a new folder in this folder’s Store. Where *type* would be:Folder.HOLDS_MESSAGES or Folder.HOLDS_FOLDERS. Returns *true* if folder is successfully created else returns *false*.
abstract boolean *delete*booleanrecurse	This deletes the folder only if the folder is closed. Otherwise, it throws an *IllegalStateException*. If *recurse* is *true*, then subfolders are deleted.
abstract boolean *renameTo*Folderf	This changes the name of this folder. A folder must be closed to be renamed. Otherwise, an IllegalStateException is thrown.

Managing Messages in Folders

Following are some of the methods that help manage the messages in Folder:

Method	Description
abstract void *appendMessages*Message[]messages	As the name implies, the messages in the array are placed at the end of this folder.
void *copyMessages*Message[]messages, Folderdestination	This copies messages from this folder into a specified folder given as an argument.
abstract Message[] *expunge*	To delete a message from a folder, set its Flags.Flag.DELETED flag to

	true. To physically remove deleted messages from a folder, you have to call this method.

Listing the Contents of a Folder

There are four methods to list the folders that a folder contains:

Method	Description
Folder[] *list*	This returns an array listing the folders that this folder contains.
Folder[] *listSubscribed*	This returns an array listing all the subscribed folders that this folder contains.
abstract *Folder*[] list*Stringpattern*	This is similar to the *list* method except that it allows you to specify a pattern. The pattern is a string giving the name of the folders that match.
Folder[] *listSubscribed**Stringpattern*	This is similar to the *listSubscribed* method except that it allows you to specify a pattern. The pattern is a string giving the name of the folders that match.

Checking for Mail

Method	Description
abstract int *getMessageCount*	This method can be invoked on an open or closed folder. However, in the case of a closed folder, this method may *ormaynot* return -1 to indicate that the exact number of messages isn't easily available.
abstract boolean *hasNewMessages*	This returns *true* if new messages have been added to the folder since it was last opened.
int *getNewMessageCount*	It returns the new message count by checking messages in the folder whose RECENT flag is set.
int *getUnreadMessageCount*	This can be invoked on either an open or a closed folder. However, in the case of a closed folder, it may return -1 to indicate that the real answer would be too expensive to obtain.

Getting Messages from Folders

The Folder class provides four methods for retrieving messages from open folders:

Method	Description
abstract Message *getMessage**intmessageNumber*	This returns the nth message in the folder. The first message in the folder is number 1.
Message[] *getMessages*	This returns an array of *Message* objects representing all the messages in this folder.

Message[] *getMessages*int*start*, int*end*	This returns an array of *Message* objects from the folder, beginning with start and finishing with end, inclusive.
Message[] *getMessages*int[]*messageNumbers*	This returns an array containing only those messages specifically identified by number in the *messageNumbers* array.
void *fetch*Message[]*messages*, FetchProfile*fp*	Prefetch the items specified in the FetchProfile for the given Messages. The FetchProfile argument specifies which headers in the messages to prefetch.

Searching Folders

If the server supports searching *asmanyIMAPserversdoandmostPOPserversdon't*, it's easy to search a folder for the messages meeting certain criteria. The criteria are encoded in SearchTerm objects. Following are the two search methods:

Method	Description
Message[] *search*SearchTerm*term*	Search this Folder for messages matching the specified search criterion. Returns an array containing the matching messages. Returns an empty array if no matches were found.
Message[] *search*SearchTerm*term*, Message[]*messages*	Search the given array of messages for those that match the specified search criterion. Returns an array containing the matching messages. Returns an empty array if no matches were found. The the specified Message objects must belong to this folder.

Flags

Flag modification is useful when you need to change flags for the entire set of messages in a Folder. Following are the methods provided in the Folder class:

Method	Description
void *setFlags*Message[]*messages*, Flags*flag*, boolean*value*	Sets the specified flags on the messages specified in the array.
void *setFlags*int*start*, int*end*, Flags*flag*, boolean*value*	Sets the specified flags on the messages numbered from start through end, both start and end inclusive.
void *setFlags*int[]*messageNumbers*, Flags*flag*, boolean*value*	Sets the specified flags

	on the messages whose message numbers are in the array.
abstract Flags *getPermanentFlags*	Returns the flags that this folder supports for all messages.