マニュアルに戻る。

チャットを作ろう

ネットワーク機能を使って、教室の中で**チャット**（画面でのおしゃべり）をするプログラムを作ってみよう。

メッセージを送信する

最初に、メッセージを送るプログラムを作る。画面にメッセージを入力する入力欄を作り、そこにメッセージを入力して**リターンキー** を押すと、メッセージがサーバーに書き込まれるようにする。

次のプログラムでは、「送信フィールド」 というフィールドオブジェクトを作り、リターンキーを押したときに、送信フィールドに書かれた文字列をサーバーに □msg□ という名前で書き込む。送ったことが分かるように、最後にフィールドの中身を 「クリア」 で空にするようにした。

```
サーバ□□"localhost"□接続。
送信フィールド＝フィールド！作る。
フィールド：動作＝「
    サーバ□□"msg"□□送信フィールド！読む）書く。
    送信フィールド！クリア。
□□
```

サーバーに接続するプログラムを実行すると、ドリトルの編集画面には「サーバー」 というボタンが表示される。このボタンを押すことで、サーバーに書き込まれたオブジェクトを確認できる□□msg□文字列）:はろー」から、「はろー」という文字列がmsgという名前で書き込まれていることが分かる。

メッセージを受信する

次のプログラムは、メッセージを受信するプログラムである。実行すると、サーバーから □msg□ という文字列オブジェクトを読み出し、ラベルに表示する。

```
サーバ□□"localhost"□接続。
受信メッセージ＝サーバ□□"msg" 読む。
ラベル！（受信メッセージ）作る。
```

送信と受信のプログラムを1台のコンピュータで実行する場合には、次のようにする。

- 1台のコンピュータでドリトルを2つ起動する。
- それぞれの画面で送信と受信のプログラムを入力する。
- どちらか一方でサーバーを起動する。
- 送信側のプログラムを実行し、メッセージを書き込んだ後、受信側のプログラムでメッセージを受信する。

ネットワークで接続された複数台のコンピュータで動作を確認する場合には、次のようにする。

- 複数台のコンピュータでドリトルを起動する。2台以上であれば何台でも構わない。
- 1台のドリトルでサーバーを起動する。（他のドリトルでは、サーバーを「終了」ボタンで閉じておくとよい）
- ドリトルで送信または受信のプログラムを入力する。
- プログラムの「localhost□の部分にサーバーを実行しているコンピュータのIPアドレスを記述する。
- 送信側のプログラムを実行し、メッセージを書き込んだ後、受信側のプログラムでメッセージを受信する。[1)]

発言に名前を入れる

複数の人で発言を書き込む場合には、誰が書いた発言かが分かると便利である。そこで、「はろー」という発言するときは、前に自分の名前を入れて、「かめた:はろー」という形で書きこむことを考える。

次のプログラムでは、「名前」という変数に名前を入れている。ここでは「かめた」としているが、実際にはそれぞれの自分の名前を書いておく。サーバーには名前に“:”と発言の文字列を連結してから書き込んでいる。「連結」は、文字列に他の文字列を連結する命令である。

```
名前＝"かめた"。
サーバ□□"localhost"□接続。
送信フィールド＝フィールド！作る。
フィールド：動作＝「
  サーバ□□"msg"□名前！":"（送信フィールド！読む）連結）書く。
  送信フィールド！クリア。
」□□
```

受信の自動化（1）

他の人の発言はいつ書き込まれるか分からないので、受信プログラムを何度も実行して新しい発言をチェックするのは大変である。そこで、定期的にサーバーと通信して、新しい発言を表示するプログラムを考えてみる。

次のプログラムを実行すると、サーバーから1秒間隔でメッセージを読み、「受信表示」というラベルに出力する。この動作を600回（10分間）繰り返す。

```
サーバ□□"localhost"□接続。
受信表示＝ラベル！作る　300　45　大きさ。
タイマー！作る　1秒　間隔　600　回数「
    受信表示！（サーバ□□"msg"□読む）書く。
」実行。
```

受信の自動化（2）

前のプログラムでは、受信したメッセージは1秒ごとに自動的に書き換えられるため、よく見ていないと読まないうちに次のメッセージに進んでしまう。そこで、やり取りしたメッセージを画面に残すプロ

グラムを考える。次のプログラムでは、リストオブジェクトで複数個のメッセージを表示している。

```
サーバ□□"localhost"□接続。
受信表示＝リスト！作る　300　400　大きさ。
タイマー！作る　1秒　間隔　600　回数「
    受信メッセージ＝サーバ□□"msg"□読む。
    受信表示！（受信メッセージ）書く。
」実行。
```

実行すると、1秒ごとにサーバーと通信し、受信したメッセージを追記する。

実行結果を見ると分かるように、このままでは同じメッセージが何度も書かれてしまう。そこで、新しい発言であることをチェックするために、直前の発言と比較し、異なっている場合だけリストに書くようにプログラムを修正する。

次のプログラムでは、サーバーからメッセージを読むたびに、受信したメッセージ（「受信メッセージ」）を直前のメッセージ（「直前メッセージ」）と比較し、異なる場合だけ表示している。そして比較した後で、受信したメッセージを直前のメッセージに代入している。

```
サーバ□□"localhost"□接続。
受信表示＝リスト！作る　200　400　大きさ　0　200　位置。
直前メッセージ＝""。
タイマー！作る　1　間隔　600　回数「
    受信メッセージ＝サーバ□□"msg"□読む。
    「受信メッセージ!＝直前メッセージ」！なら「受信表示！（受信メッセージ）書く」実行。
    直前メッセージ＝受信メッセージ。
」実行。
```

実行すると、1秒ごとにサーバーと通信し、新しいメッセージがあったときだけ、画面に表示する。

チャットプログラム

ここまで作ってきた送信プログラムと受信プログラムを組み合わせると、送信と受信の両方を扱うチャットプログラムを作ることができる。

次のプログラムは、**sec□chatsend** と**sec□chatrecv** で作った2つのプログラムを結合させて作ったプログラムである。送信するメッセージを入力する 「メッセージ」 フィールドと、受信したメッセージを表示する 「受信メッセージ」 リストが重ならないように位置を調整している。

```
サーバ□□"localhost"□接続。
サーバ□□"msg" "" 書く。

名前＝"かめた"。
メッセージ＝フィールド！作る　200　45　大きさ　−250　200　位置。
フィールド：動作＝「
    サーバ□□"msg"□□名前！":"　（メッセージ！読む）連結）書く。
    メッセージ！クリア。
 □□
```

```
受信表示＝リスト！作る　200　400　大きさ　0　200　位置。
直前メッセージ＝""。
タイマー！作る　1　間隔　600　回数「
    受信メッセージ＝サーバ□□"msg"□読む。
    「受信メッセージ!＝直前メッセージ」！なら「受信表示！（受信メッセージ）書く」実行。
    直前メッセージ＝受信メッセージ。
」実行。
```

実行すると、送信するメッセージの入力欄と受信したメッセージの表示欄が画面に表示される。これは「かめきち」 がメッセージを送ろうとしている画面である。

[1)]

メッセージを複数のドリトルから送信することもできる。新しいメッセージが書き込まれると前のメッセージは上書きされてしまうので、複数人で実行する場合には、「書いたよ」と声を掛け合いながら送信と受信のプログラムを実行するとよい。