فري إسبايدر لبرمجة الويب

لبيئة

فري باسكال/لازاراس

دليل المستخدم

FreeSpider for Web Development

نسخة رقم 1.2.6
تعديل رجب 1433 هجرية
المؤلف: معتز عبدالعظيم الطاهر
الترخيص: LGPL
http://code.sd

مقدمة

فري إسبايدر **FreeSpider** هي عبارة عن حزمة ومكتبة لبرمجة الويب في بيئة فري باسكال/لازاراس. عن طريقها يُمكن تصميم برامج ويب تفاعلية في بيئة لينكس، وندوز أو مكانتوش.
مترجم فري باسكال يُنتج برنامج تنفيذية معتمدة على نفسها، لايحتاج إلى مكتبات خارجية لتشغيلها مما يسهِّل تثبيتها في مخدمات الويب المختلفة.
البروتكول المتوفر الآن في أداة فري إسبايدر هو برتكول CGI. وفي المستقبل يمكن إضافة بروتكولات أخرى إن شاء الله.

كيفية إنشاء برامج ويب عن طريق فري سبايدر

نتبع الخطوات التالية:

1. في القائمة الرئيسية من بيئة لازاراس نختار File/New.. ثم نختار Free Spider CGI Web Application
2. نضع TSpiderCGI من صفحة FreeSpider في الـ DataModule

3. مناداة الإجراء Execute في الحدث OnCreate للـ DataModule1 بهذه الطريقة:

```
SpiderCGI1.Execute;
```

4. قم بكتابة هذا الكود في الحدث OnRequest للمكوّن SpiderCGI1

```
Response.ContentType:= 'text/html; charset=UTF-8';
Response.Add('Hello world');
```

5. تم بتغيير دليل إستخراج الملف التنفيذي إلى cgi-bin الخاص ببرنامج الـ web server الذي تستخدمه. ويمكن تغييره بواسطة:

Options/Application Tab/Output settings group/Target file name

6. إذا كان اسم المشروع هو first يمكن مناداته عن طريق متصفح الويب عن طريق كتابة

هذا الرابط:

```
http://localhost/cgi-bin/first
```

الكائن Request

وهو كائن يحتوي على معلومات الطلب مثل: عنوان المتصفح ونوعه، والبراميترات التي تم نداء برنامج الويب عن طريقها مثلاً في النداء التالي:

```
http://localhost/cgi-bin/first?name=Mohammed&address=Khartoum
```

يمكن الوصول لقيم هذه المتغيرات عن طريق:

```
UserName:= Request.Query('name');
Address:= Request.Query('address');
```

Also you can access it from Query's String List:

```
UserName:= Request.QueryFields.values['name'];
Address:= Request.QueryFields.values['address'];
```

هذه الطريقة لإرسال وإستقبال البيانات تسمى GET، أما طريقة POST فهي الطريقة التي يستخدم فيها فورم HTML وفي هذه الحالة لاتظهر المتغيرات في الرابط، كذلك يمكن إرسال بيانات بكمية أكبر عن طريق POST مثل إرسال ملف كامل. يمكن قراءة محتويات الفورم المستخدمة فيه طريقة POST كالآتي:

```
  Login:= Request.Form('login');
  Password:= Request.Form('password');
```

أيضاً يمكن استخدام ContentFields والتي نوعها TstringList للوصول لنفس البيانات بالإضافة إلى عددها وأسماء المتغيرات أو البراميترات (حقول الفورم) التي تم إرسالها.

```
  Login:= ContentFields.values['login'];
  Password:= ContentFields.values['password'];
```

كذلك فإن هذا الكائن يحتوي على خصائص أخرى يتضح معناها من إسمها وهي:

```
Request.RemoteAddress
Request.UserAgent
Request.RequestType
```

الكائن Response

عن طريق هذا الكائن يقوم المبرمج بإنشاء الصفحة التي تمثل الرد على الطلب Request. ويمكن إنشاء صفحة الرد على المستخدم لطلبه بواسطة الحدث OnRequest، مثلاً:

```
  Response.Add('Hello world<br>');
  Response.Add('This has been written in <b>Object Pascal</b>');
```

الكائن TSpiderAction

الكائن TspiderCGI هو أساسي لكل برامج فري إسبايدر، حيث لابد من وجود كائن واحد فقط لكل برنامج. لكنه يحتمل فقط طلب واحط يمكن أن ينادى بهذه الطريقة:

```
http://localhost/cgi-bin/first
http://localhost/cgi-bin/first?name=Motaz
```

أما بالنسبة للكائن TspiderAction فيمكن أن يحتوي البرامج الواحد على عدد كبير منه على حسب إختلاف الطلبات. حيث يُوضع كائن واحد مقابل كل طلب، ويتم التفرقة بينها بما يعرف بالمسار Path. مثلاً إذا افترضنا أن البرنامج هو بريد إلكتروني، فيمكن أن يحتوي على الطبات مثل:
register user, login, logout, view Inbox, send
حيث تتم مناداتها من المتصفح أو صفحة الويب كالآتي:

```
http://localhost/cgi-bin/mail/login
http://localhost/cgi-bin/mail/logout
http://localhost/cgi-bin/mail/reg
http://localhost/cgi-bin/mail/inbox
```

لكل كائن من نوع TspiderAction كائنات للطلب والرد Request, Response، كذلك فإن له حدث OnRequest، فهو مشابه للكائن TspiderCGI.

الكائن TSpiderTable

هذا الكائن يقوم بإنشاء جدول ويب، إما بطريقة يدوية أو بربطه بمجموعة بيانات أُخرى.

ومثال للطريقة اليدوية:

```
SpiderTable1.SetHeader(['ID', 'Name', 'Telephone Number']);
SpiderTable1.AddRow('', ['1', 'Mohammed', '01223311']);
SpiderTable1.AddRow('#FFEEDD', ['2', 'Ahmed', '01754341']);
SpiderTable1.AddRow('#FFDDEE', ['3', 'Omer', '045667890']);
Response.Add(SpiderTable1.Contents);
```

فينتج عنها جدول في المتصفح كالآتي:

ID	Name	Telephone Number
1	Mohammed	01223311
2	Ahmed	01754341
3	Omer	045667890

الطريقة الثانية هي ربطه بمجموعة بيانات، حيث يمكن ربطه بأي مكوّن يرث من النوع TdataSet.
نقوم في هذه الحالة بإختيار مجموعة البيانات في الخاصية DataSet إما في وقت التصميم أو التنفيذ، ثم نضع محتويات الجدول في مكوّن الرد Response كالآتي:

```
SpiderTable1.DataSet:= ATable;
ATable.Open;
Response.Add(SpiderTable1.Contents);
```

يحتوي هذا المكوّن على الأحداث التالية:

OnDrawHeader: ويتم ندائه عند كتابة رأس الجدول (أسماء الحقول).
OnDrawDataCell: ويتم ندائه عند كتابة محتويات الجدول (محتويات الحقول).
كلتا الحدثين السابقين يحتوي على المتغيرات **CellData** و **BgColor** والتي تسمح للمبرمج أن يقوم بتغيير اللون أو المحتويات بناءً على شرط معين يقوم بكتابته في هذه الأحداث.

الكائن TSpiderForm

وهو يقوم بإنشاء فورم ويب، والذي يستخدم لإدخال بيانات ليقوم المستخدم بإرسال هذه المعلومات كطلب معين، وغالباً مايكون نوع الطلب هو POST.
هذا الكائن يحتوي على الخصائص التالية:

1. Method: وقيمته الإفتراضية هي POST والتي تتميز بإمكانيتها بإرسال بيانات ذات سعة كبيرة أو ملفات. والبيانات المرسلة لاتظهر على الرابط، بخلاف الطريقة الثانية GET.

2. Action: وهي تمثل مُستقبِل الطلب، فإذا كان مثلاً:

```
/cgi-bin/first/login
```

فإن محتويات هذا الفورم سوف تُرسل إلى الـ SpiderAction الذي يحتوي على المسار / login

3. ExtraParam: يمكن أن تحتوي على خصائص إضافية لحقل الفور، مثلاً أبعاد المساحة النصّية Text Area ، أو كود جافا سكريبت الذي يتم تنفيذه مثلاً عن الضغط على زر في الفورم.

4. PutInTable: قيمتها الإفتراضية هي True والتي تقوم بوضع حقول الفورم وعناوين هذه الحقول في جدول يتكون من عمودين، حيث أن العمود الأول يحتوي على العنوان الذي تم إدخاله بواسطة AddText والعمود الثاني يحتوي على الحقل الذي تمت إضافته بواسطة AddInput. ويجب التقيد بتنفيذ الإجراء AddText ثم AddInput في حالة إن كان True خصوصاً مع مربعات النص. ويمكن تجاهل AddText إذا كان نوع الحقل هو زِر أو كانت قيمته False.

مثال لطريقة إنشاء فورم

```
SpiderForm1.AddText('Enter Subscriber ID: ');
SpiderForm1.AddInput(itText, 'id', '');

SpiderForm1.AddText('Enter Subscriber Name: ');
SpiderForm1.AddInput(itText, 'subname', '');

SpiderForm1.AddText('Address: ');
SpiderForm1.AddInput(itText, 'address', '');

SpiderForm1.AddText('Telephone Number: ');
SpiderForm1.AddInput(itText, 'telephone', '');

SpiderForm1.AddInput(itSubmit, 'add', 'Add');

Response.Add(SpiderForm1.Contents);
```

This will generate form in user's browser like this:

Enter Subscriber ID: [ ]
Enter Subscriber Name: [ ]
Address: [ ]
Telephone Number: [ ]
Add

الكائن TSpiderPage

وهو يُمكّن المبرمج من إنشاء صفحة ويب عن طريق أي برنامج خارجي مثل Open Office Writer ثم ربط هذه الصفحة بمحتويات حيّة تمثل قيم تمت قرائتها من قاعدة بيانات مثلاً.
ويستخدم في هذه الطريقة بمايعرف بالـ **Tags** وهي رموز توضع في الصفحة على الشكل @tag;
يتم إستبدالها لاحقاً في الحدث **OnTag** بالنسبة لهذا الكائن. ويمكن وضع هذه الصفحة مع البرنامج في نفس الدليل ولايشترط أن يكون هذا الدليل قابل للتصفح من قبل المتصفح.
وبهذه الطريقة نُمكن من فصل تصميم صفحات الويب من البرنامج الرئيسي، ويمكن أن نستعين بمصمم صفحات ليس بالضرورة أن يكون مبرمج ليقوم بتصميم الشكل العام وصفحات الويب كقوالب يستخدمها المبرمج لعرض البيانات الحية.

دورة حياة الـ CGI

يتميز برنامج الويب من النوع CGI بقصر دورة حياته. حيث أن البرنامج يتم تحميله في الذاكرة ليعمل ثم ينغلق في الفترة مابين ضغطة المستخدم لرابط أو زر ما إلى أن تظهر الصفحة الرد، والتي غالبا ماتكون ثواني أو أجزاء من الثانية. لذلك فإن كتابة برنامج ويب من نوع CGI هو من الأشياء السهلة في لغة فري باسكال، لأنه بإنغلاق البرنامج يتم تحرير كل الذاكرة المستخدمة وتغلق الجداول والملفات المفتوحة، حيث يمكن للمبرمج أن لايأخذ وقتاً طويلاً في التأكد من أنه قام بتحرير كل الموارد المحجوزة، لأنه بإنتهاء البرنامج تحرر كل الموارد المحجوزة حتى ولو لم تحرر بالكود. وهذه الميزة تجعل برامج الويب هذه تعمل في المخدمات لفترات طويلة بدون حدوث تسريب للذاكرة أو فشل في مخدم الويب.

الطلبات في ذات اللحظة Concurrent requests

من الأشياء المهمة في برامج الويب هو عدد الطلبات التي تُطلب في لحظة واحدة، مثلاً إذا كانت هناك ثلاث طلبات في ذات اللحظة ففي هذه الحالة يتم تحميل وتشغيل ثلاث نُسخ من نفس برنامج الويب المصمم بواسطة فري باسكال. وهذه الخاصية تجعل برامج الـ CGI غير مناسبة للنُظم التي تتعامل مع عدد كبير جداً من الطلبات في نفس اللحظة. وعدد الطلبات الأقصى الذي يمكن أن يتحمله برنامج الويب يعتمد على هذه المتغيرات:

1. حجم ذاكرة المُخدم
2. إحتياج برنامج الويب من الذاكرة
3. متوسط سرعة الرد بالنسبة للطلبات.

فنجد أنه كلما كانت سرعة الرد عالية والذاكرة كبيرة وإحتياج البرنامج من الذاكرة قليل كلما زادت إمكانية إستقبال طلبات أكثر في نفس اللحظة.
وتوجد هناك نقطة مهمة في هذا الحساب، فإذا كان العدد التي يتحمله مخدم ما هو 50 طلب في نفس اللحظة مثلاً فهذا لايعني أن هذا النظام يصلح فقط لخمسين مستخدم، بل يمكن أن يصلح لمائتي مستخدم، بحيث أن خمسين فقط منهم يمكن أن يطلب في نفس اللحظة. فإذا كان المستخدمين لايتعاملون مع هذه الصفحات وهذا المخدم طوال الوقت، أي أن طلباتهم قليلة أثناء اليوم، فيمكن أن يصلح هذا النظام لعدد ألف مستخدم مثلاً. وإذا أحسسنا ببطء إستجابة الطلبات يمكن إضافة مخدم ويب آخر وإنزال نفس النسخة من البرنامج وتقسيم المستخدمين بين هذه المخدمات لتقليل الإزدحام.

جلسة المستخدم

جلسة المستخدم User session يقصد بها متابعة وتفريق برنامج الويب لطلبات المتسخدمين في حالة وجود علاقة بين الطلب الأول والثاني، فبما أن برامج الويب من نوع CGI تتميز بدورة حياة قصيرة، نجد أنها تعجز عن ربط الطلب الحالي لمستخدم ما بالطلب اللاحق. ويتم سد هذه الفجوة بما يعرف بالسكاكر أو الـ Cookies وهي عبارة عن بيانات يضعها برنامج الويب في متصفح المستخدم، ويقوم المتصفح تلقائياً بإرسالها مرة أخرة لبرنامج الويب حتى يتعّرف برنامج الويب على صاحب الطلب ومتابعته. ويمكن إرسال سكاكر لمتصفح الويب بمثل هذه الطريقة:

```
Response.SetCookie('sessionid', '2', '/');
```

والمُدخل الأخير وهو المسار والذي يخصص للمتصفح متى يقوم بإرسال هذه السكاكر. فإذا كان يحمل القيمة '/' فهو يعني أن كل الطلبات لهذا المخدم يتم إرسال قيمة هذه السكاكر. وبهذه الطريقة يمكن وضع أكثر من برنامج في نفس المخدم وجميعها تستطيع قراءة محتويات هذه السكاكر.
قيمة sessionid سوف يحتفظ بها المتصفح حتى يتم غلقه.
فإذا اردنا أن نحدد تاريخ ينتهي عنده صلاحية المتغير sessionid فيجب إدخال قيمة المتغير Expiration، فمثلاً في المثال التالي سوف تكون صلاحية هذا المتغير يوماً كاملاً:

```
Response.SetCookie('sessionid', '2', '/', Now + 1);
```

وعندما نريد تحديد عمر لهذه القيم بالساعات أو الدقائق، فيجب علينا أولاً معرفة توقيت الدولة ثم طرحه من هذه القيمة. مثلاً عندما نريد تحديد عمر السكاكر بخمس دقائق فقط، نكتب الكود التالي:

```
  Response.SetCookie('sessionid', '2', '/', Now -
    EncodeTime(3, 0, 0, 0) + EncodeTime(0, 5, 0, 0));
```

يمكن قراءة محتويات السكاكر بمثل هذه الطريقة:

```
Response.Add(Request.GetCookie('sessionid'));
```

تحميل وإنزال الملفات

لتحميل ملف من جهاز المستخدم بواسطة المتصفح يجب أن نقوم بتحويل نوع ترميز الفورم إلى النوع **multipart/form-data** ، وذلك عن طريق كتابة هذا الكود أو وضع هذه القيمة في الخاصية **ExtraParam** في الفورم:

```
SpiderForm1.ExtraParam:= 'enctype="multipart/form-data"';
```

مثال:

```
    Response.Add('<h2>File Upload sample</h2>');
    SpiderForm1.AddInput(itFile, 'upload');
    SpiderForm1.AddInput(itSubmit, 'upload', 'Upload');
    Response.Add(sfUpload.Contents);
```

يمكن إستقبال الملف المرسل بواسطة الخاصية **ContentFiles**، ويمكن إنزال الملف مرة أخرى إلى جهاز المستخدم عن طريق هذا الكود:

```
    Response.ContentType:= Request.ContentFiles[0].ContentType;
    Response.CustomHeader.Add('Content-Disposition: filename="' +
      Request.ContentFiles[0].FileName + '"');
    Response.Content.Add(Request.ContentFiles[0].FileContent);
```

طريقة تصميم الموديول الذكي
Smart Module Loading Design

وهي طريقة جديدة تمتاز بها فري إسبايدر تُمكّن المبرمج من تصميم برامج ويب تحتوي على عدد كبير من الـ Data Modules. حيث أن كل موديول يمكن أن يحتوي على Spider Actions, Pages, Tables, forms وماتحتاجه من مكونات قواعد بيانات وغيرها من الكائنات.
ويتم فقط تحميل وإنشاء الموديول فقط الذي يحتوي على الرد للطلب المرسل. بالإضافة للموديول الرئيسي الذي يحتوي على SpiderCGI. فمثلاً إذا افترضنا أن لدينا برنامج ويب يحتوي على 100 موديول وتم طلب من المستخدم وهذا الطلب له حدث في إحدى الموديولات فسوف يتم فقط في هذه الحالة إنشاء وتحميل فقط إثنين من الموديولات، الموديول الرئيسي ومعه الموديول الذي يحتوي على الرد لهذا الطلب. وبهذه الطريقة نضمن إستخدام أقل للذاكرة وسرعة تحميل وإستجابة عاليتين للطلبات مهما كان حجم وتعقيد البرنامج.

لعمل برنامج بإستخدام هذه الطريقة نقوم بالآتي:

1. إنشاء برنامج فري إسبايدر جديد بنفس الطريقة المشروحة في هذا الكُتيّب.
2. نقوم بإضافة Data Module جديد.
3. نقوم بنداء الإجراء RegisterClass لتسجيل هذا الموديول. فمثاً إذا افترضنا أن إسم هذا الموديول هو TdmMod2 فنقوم بكتابة الكود التالي في جزئية الـ initialization:

```
initialization
  {$I mod2.lrs}
  RegisterClass(TdmMod2);
```

4. نقوم بإضافة إسم هذه الوحدة التي تحتوي على هذا الموديول في الموديول الرئيسي الأول الذي يحتوي على SpiderCGI.
5. نضع في الموديول الجديد مكونات TspiderAction ونقوم بتحديد مسارات جديدة مختلفة عن المسارات الموجودة في الموديول الرئيسي.

6. في الحدث OnCreate للموديول الرئيسي نقوم بكتابة الكود التالي

```
SpiderCGI1.AddDataModule('TdmMod2', ['/path2', '/path3']);
SpiderCGI1.Execute;
```

المُدخل Path للإجراء AddDataModule يجب أن يحتوي على كل المسارات الموجودة في المكونات SpiderActions الموجودة في الموديول الجديد، وإلا لن يتم ندائه، لأنه بهذه الطريقة يتعرف فري إسبايدر على موقع المسارات ليتم ربطها مع الموديول المعني.

برنامج SpiderSample تم تصميمه بهذه الطريقة ويمكن الرجوع له لمعرفة كيفية عمل مثل هذا النوع
من البرامج.

ملحوظة:
يمكن وضع كائنات قواعد البيانات التي يتم عن طريقها الإتصال بقاعدة البيانات مثل Connection Objects في الموديول الرئيسي ثم ربطها مع أي موديول آخر، كذلك يمكن في الموديول الرئيسي وضع كل الكائنات أو المتغيرات المشتركة التي يحتاجها البرنامج في كل النداءات.

قياس الأداء

يمكن الرجوع لصفحة الـ Performance ضمن صفحة فري سبادير في الموقع لمعرفة تفاصيل قياس الأداء الذي تمت تجربته مع برامج فري سبايدر.
وكملخص لهذه القياسات التي تمت في مخدم ثنائي النواة 2.16 غيغا هيرتز، وذاكرة 2 غيغا بايت، ومع إستخدام قاعدة بيانات وجدنا أنه يمكن تلبية 2500 طلب في الدقيقة في حالة عدم وجود طلبات في نفس الوقت (غير وقت الذروة).
أما في وقت الذروة عند وجود طلبات في نفس الزمن فإن الأداء سوف يتراجع بسبب قاعدة البيانات ليخدم فقط 1000 طلب في الدقيقة. وهذه القياسات تختلف بإختلاف ظروف ساعة الذروة وتعقيد الإجراءات المرتبطة بقاعدة البيانات.
فإذا افترضنا أننا قمنا بإنزال برنامج ويب مصمم بواسطة فري سبايدر وهو مربوط بقاعدة بيانات في شركة ما، وكان موظفو هذه الشركة يطلبون متوسط صفحة واحدة فقط في الدقيقة، فيمكن لهذا البرنامج خدمة 1000 موظف في ساعات الذروة، أو 2500 موظف في الساعات العادية أو عندما لاتكون هناك ساعة ذروة (عندما تتوزع الطلبات خلال ساعات اليوم بصورة طبيعية وعدم وجود ساعة تكثر فيها الطلبات أكثر من غيرها).

معتز عبدالعظيم
www.code.sd