

Использование JSON с Flex 2

Автор: [Майк Чемберз](#)

[Оригинал статьи](#)

Перевод и отсебятина: [Горбатов Андрей](#) ([блог](#))

Что такое JSON?

[JSON](#) (JavaScript Object Notation) – процесс сериализации данных с помощью JavaScript. Конкурент XML.

Сравним эти формата:

XML:

```
<classinfo>
  <students>
    <student>
      <name>Michael Smith</name>
      <average>99.5</average>
      <age>17</age>
      <graduating>true</graduating>
    </student>
    <student>
      <name>Steve Johnson</name>
      <average>34.87</average>
      <age>17</age>
      <graduating>>false</graduating>

    </student>
    <student>
      <name>Rebecca Young</name>
      <average>89.6</average>
      <age>18</age>
      <graduating>true</graduating>
    </student>
  </students>
</classinfo>
```

JSON:

```
{ "classinfo" :
  {
    "students" : [
      {
        "name" : "Michael Smith",
        "average" : 99.5,
        "age" : 17,
        "graduating" : true
      },
      {
        "name" : "Steve Johnson",
        "average" : 34.87,
        "age" : 17,
        "graduating" : false
      }
    ]
  }
}
```

```

    },
    {
        "name" : "Rebecca Young",
        "average" : 89.6,
        "age" : 18,
        "graduating" : true
    }
]
}
}

```

Как можно заметить, в случае с JSON, отсутствует много лишней информации. Однако минусом является пониженная читаемость кода. Но расстраиваться не стоит - существует множество инструментов перевода из/в JSON формат. Вот некоторые ссылки:

- **C#/.NET:** www.crockford.com/JSON/cs/.
- **ColdFusion:** <http://jehiah.com/projects/cfjson/>.
- **Java:** www.crockford.com/JSON/java/.
- **Perl:** <http://search.cpan.org/dist/JSON/>.
- **PHP:** www.aurore.net/projects/php-json/, <http://mike.teczno.com/json.html>
- **Python:** <https://sourceforge.net/projects/json-py/>.

Нас интересует использование JSON и Flex 2.

Класс JSON можно найти в библиотеке [corelib](#). Класс, написанный Дэроном Шеллом ([Darron Schall](#)), облегчает сериализацию и десериализацию JSON данных.

Итак, для демонстрации использования JSON внутри Flex приложения, мы будем загружать [JSON фид](#), который периодически генерируется PHP файлом, в приложение и отображать в его в datagrid.



Service	Title
digg	Tutorial : Using JSON with Flex 2 and ActionScript 3
delicious	Windows Is So Slow, but Why? - New York Times
delicious	WindowsDevCenter.com -- Create Podcasts Using Your PC
digg	Flex is Free!
delicious	Dax Pandhi's Digital Exile - The Future of WPF / Flash vs WPF
delicious	Microsoft Gadgets
delicious	Synergy
delicious	Microsoft XNA
digg	Oh My God! The Kid Just Drew On My LCD With a Ballpoint Pen!
digg	Microsoft to offer cheap Xbox 360 game development kit
digg	Google Finance with open source Flash / Ajax integration!
delicious	popurls.com popular urls to the latest web buzz
delicious	Bona tempora voluntur: by Guy Kawasaki: The Art of Sucking Down
delicious	Microsoft Typography - Font properties extension, version 2.1
delicious	Trunks and branches in subversion
delicious	SWCDOC: Audio Interface Adapter

Должно быть установлено следующее:

- [Flex Builder](#) или [MXMLC компилятор](#)
- [Flash Player 9](#)
- [библиотека corelib](#)

В архиве corelib находятся следующие папки:

- src : исходники
- bin : SWC файл
- docs : содержит API библиотеки

Мы будем использовать SWC файл, так как исходники правим не надо в данном случае.

Итак, открываем Flex Builder (я делал в бесплатном [FlashDevelop](#) – прим. пер.), убеждаемся, что открыта перспектива:

Window > Open Perspective > Flex Development

Далее в навигаторе (Window > Show View > Navigator) кликаем правой кнопкой мыши (New > Flex Project), называем "JSONExample".

Затем нажмите "Next" (не Finish!), выберите Add SWC и добавьте файл corelib.swc, который находится в папке bin пакета corelib. Жмем Finish.

Сначала необходимо добавить datagrid для отображения данных. Она должна содержать две колонки: для заголовков и типов сервисов (delicious, digg, tada и т.д.).

Откройте JSONExample.mxml в редакторе, перейдите в режим Design. Добавьте из вида Components (Window > Show View > Components) элемент DataGrid. Выберите его в редакторе и в виде Flex Properties (Window > Show Views > Flex Properties) добавьте:

1. В поле ID - grid
2. Установите якоря для компонента: отметьте чекбоксы для
 - Top Right
 - Top Left
 - Left Top
 - Left Bottom

Перейдите в режим Code:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*"
layout="absolute">
    <mx:DataGrid id="grid" right="10" left="10" top="10" bottom="10">
        <mx:columns>
            <mx:DataGridColumn headerText="Column 1"
dataField="col1"/>
            <mx:DataGridColumn headerText="Column 2"
dataField="col2"/>
            <mx:DataGridColumn headerText="Column 3"
dataField="col3"/>
        </mx:columns>
    </mx:DataGrid>
</mx:Application>
```

Добавим тег HTTPService под тегом Application:

```
<mx:HTTPService id="service" resultFormat="text"
url="http://weblogs.macromedia.com/mesh/mashedpotato.json"
result="onJSONLoad(event)" />
```

Вероятно, возникнет ошибка (Window > Show Views > Problems):

```
Call to a possibly undefined method 'onJSONLoad'
```

Значит мы должны добавить обработчик события onJSONLoad. Но сначала рассмотрим код, который мы написали:

- id - идентификатор элемента, который будет использован в дальнейшем.
- url – ссылка на загружаемые JSON данные
- resultFormat – формат возвращаемых данных (В данном случае, просто текст).
- result – обработчик события, вызываемый при загрузке данных.

Код теперь выглядит так:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*"
layout="absolute">

    <mx:HTTPService id="service" resultFormat="text"

        url="http://weblogs.macromedia.com/mesh/mashedpotato.json"
        result="onJSONLoad(event)" />

    <mx:DataGrid id="grid" right="10" left="10" top="10" bottom="10">
        <mx:columns>
            <mx:DataGridColumn headerText="Column 1" dataField="col1"/>
            <mx:DataGridColumn headerText="Column 2" dataField="col2"/>
            <mx:DataGridColumn headerText="Column 3" dataField="col3"/>
        </mx:columns>
    </mx:DataGrid>
</mx:Application>
```

Добавим блок скрипта:

```
<mx:Script>
    <![CDATA[

    ]]>
</mx:Script>
```

И определим обработчик события:

```
<mx:Script>
    <![CDATA[
        import mx.rpc.events.ResultEvent;

        private function onJSONLoad(event:ResultEvent):void
        {
        }

    ]]>
</mx:Script>
```

Нам осталось:

- Получить JSON данные
- Десериализовать JSON в ActionScript
- Установить dataProvider для datagrid

Вот готовая функция с комментариями:

```
<mx:Script>
    <![CDATA[
```

```

import mx.collections.ArrayCollection;
import mx.rpc.events.ResultEvent;
import com.macromedia.serialization.json.JSON;

private function onJSONLoad(event:ResultEvent):void
{
//получаем JSON данные и приводим их к типу String
var rawData:String = String(event.result);

//переводим данные в ActionScript с помощью JSON API
//в данном случае, JSON данные - массив объектов.
var arr:Array = (JSON.decode(rawData) as Array);

//создаем ArrayCollection
//ArrayCollection лучше подходит для DataProvider
//так как может отслеживать изменения.
var dp:ArrayCollection = new ArrayCollection(arr);

//передаем ArrayCollection в качестве dataProvider DataGrid.
grid.dataProvider = dp;

}
]]>
</mx:Script>

```

Десериализация JSON состоит из одной строки:

```
var arr:Array = (JSON.decode(rawData) as Array);
```

JSON.decode возвращает Object, потом мы приводим его к типу Array с помощью оператора "as".

Осталось:

1. Сообщить сервису о необходимости загрузки данных.
2. Отредактировать DataGrid для отображения данных.

Первое сделать просто – добавим событие в тег Application:

```
creationComplete="service.send() "
```

Затем, удалим третий тег DataGridColumn – нам понадобятся только две колонки.

Теперь добавим заголовки колонок с помощью атрибута headerText и свяжем каждую колонку с загружаемыми данными. Если посмотреть на JSON данные, то можно выделить четыре поля:

- title
- src
- url
- date

Для примера необходимы только source и title. Зададим их для атрибутов dataField:

```

<mx:DataGridColumn headerText="Service" dataField="src"/>
<mx:DataGridColumn headerText="Title" dataField="title"/>

```

Полностью код выглядит так:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*"
layout="absolute"
creationComplete="service.send()">

  <mx:Script>
  <![CDATA[
    import mx.collections.ArrayCollection;
    import mx.rpc.events.ResultEvent;
    import com.macromedia.serialization.json.JSON;

    private function onJSONLoad(event:ResultEvent):void
    {
      //получаем JSON данные и приводим их к типу String
      var rawData:String = String(event.result);

      //переводим данные в ActionScript с помощью JSON API
      //в данном случае, JSON данные - массив объектов.
      var arr:Array = (JSON.decode(rawData) as Array);

      //создаем ArrayCollection
      //ArrayCollection лучше подходит для DataProvider
      //так как может отслеживать изменения.
      var dp:ArrayCollection = new ArrayCollection(arr);

      //передаем ArrayCollection в качестве dataProvider DataGrid.
      grid.dataProvider = dp;
    }
  ]]>
</mx:Script>

  <mx:HTTPService id="service" resultFormat="text"

    url="http://weblogs.macromedia.com/mesh/mashedpotato.json"
    result="onJSONLoad(event)" />

  <mx:DataGrid id="grid" right="10" left="10" top="10" bottom="10">
    <mx:columns>
      <mx:DataGridColumn headerText="Service" dataField="src"/>
      <mx:DataGridColumn headerText="Title" dataField="title"/>
    </mx:columns>
  </mx:DataGrid>
</mx:Application>
```

Запустите приложение, данные из JSON фида отобразятся в таблице.

Можно посмотреть [скринкаст урока](#).