Html5 WebSQL

本章目标

掌握htm15的常用标签

掌握htm15的常用事件

Why? & What?

在HTML5中，大大丰富了客户端本地可以存储的内容，添加了很多功能来将原本必须保存在服务器上的数据转为保存在客户端本地，从而大大提高了Web应用程序的性能，减轻了服务器端的负担，使Web时代重新回到了“客户端为重，服务器为轻”的时代。

在这其中，一项非常重要的功能就是数据库的本地存储功能。在HTML5中内置了一个可以通过SQL语言来访问的数据库。在HTML4中，数据库只能放在服务器端，只能通过服务器来访问数据库，但是在HTML5中，可以就像访问本地文件那样轻松的对内置的数据库进行直接访问。现在，像这种不需要存储在服务器上的，被称为“SQLite”的文件型SQL数据库已经得到了很广泛的利用，所以HTML5中也采用了这种数据库来作为本地数据库。

对于 Web 应用的存储，相信大家都接触过 Cookie。Cookie 用于弥补 HTTP 协议的无状态性，服务器可以使用 Cookie 中包含的信息来判断 HTTP 传输中的状态。但 Cookie 有自己固有的缺点：它的大小受限，大多数浏览器对 Cookie 大小限制为 4K；Cookie 机制可以在浏览器中被禁用；Cookie 需要在客户端和服务器端来回地传送，繁琐且消耗带宽；存在安全风险，Cookie 是以明文存放，可能被恶意客户修改，当然可以手动加密和解密 Cookie，但这需要额外的编码，并且因为加密和解密需要消耗一定的时间而影响应用程序的性能。

对于 HTML5，也许很有用的就是它新推出的“Web Storage”（Web 存储）API，它包括 localStorage 和 sessionStorage，对简单的键值对（比如应用程序设置）或简单对象（如应用程序状态）进行存储，使用本地和会话存储能够很好地完成，对于存储少量的数据非常有用，但是对大量的结构化数据进行处理时，它就力所不及了，而这正是 HTML5 的“Web SQL Database” API 接口的应用所在。

Web SQL Database API 实际上并不包含在 HTML5 规范之中。它是一个独立的规范，它引入了一套使用 SQL 操作客户端数据库的 API。最新版本的 Chrome，Safari 和 Opera 浏览器都支持 Web SQL Database。

Html5数据库API是以一个独立规范形式出现，它包含三个核心方法：

1、openDatabase：这个方法使用现有数据库或创建新数据库创建数据库对象。

2、transaction：这个方法允许我们根据情况控制事务提交或回滚。

3、executeSql：这个方法用于执行真实的SQL查询。

打开数据库

openDatabase方法可以打开已经存在的数据库，不存在则创建：

```
var db = openDatabase('mydatabase', '2.0', my db', 2 * 1024);
```

openDatabasek中五个参数分别为：数据库名、版本号、描述、数据库大小、创建回调。创建回调没有也可以创建数据库。

执行查询

database.transaction()函数用来查询，下面将在mydatabase数据库中创建表t1：

```
    var db = openDatabase(' mydatabase ', '1.0', 'Test DB', 2 * 1024 * 1024);
    db.transaction(function (tx) {
        tx.executeSql('CREATE TABLE IF NOT EXISTS t1 (id unique, log)');
    });
```

插入操作

```
var db = openDatabase('mydatabase', '2.0', my db', 2 * 1024);

db.transaction(function (tx) {
      tx.executeSql('CREATE TABLE IF NOT EXISTS t1 (id
unique, log)');
      tx.executeSql('INSERT INTO t1 (id, log) VALUES (1,
"foobar")');
      tx.executeSql('INSERT INTO t1 (id, log) VALUES (2,
"logmsg")');
});
```

在插入新记录时，我们还可以传递动态值，如：

```
   var db = openDatabase(' mydatabase ', '2.0', 'my db', 2 *
1024);
   db.transaction(function (tx) {
     tx.executeSql('CREATE TABLE IF NOT EXISTS t1 (id unique,
log)');
     tx.executeSql('INSERT INTO t1
                            (id,log) VALUES (?, ?) ', [e_id,
e_log];  //e_id和e_log是外部变量
   });
```

读操作

如果要读取已经存在的记录，我们使用一个回调捕获结果，代码如下：

```
   var db = openDatabase(mydatabase, '2.0', 'my db', 2*1024);     db.transaction(function
(tx) {
      tx.executeSql('CREATE TABLE IF NOT EXISTS t1 (id unique, log)');
      tx.executeSql('INSERT INTO t1 (id, log) VALUES (1, "foobar")');
      tx.executeSql('INSERT INTO t1 (id, log) VALUES (2, "logmsg")');
   });
   db.transaction(function (tx) {
      tx.executeSql('SELECT * FROM t1', [], function (tx, results) {
      var len = results.rows.length, i;
      msg = "<p>Found rows: " + len + "</p>";
      document.querySelector('#status').innerHTML +=  msg;
      for (i = 0; i < len; i++){
         alert(results.rows.item(i).log );
      }
    }, null);
   });
```

result

result：查询出来的数据集。其数据类型为 SQLResultSet ，就如同C#中的DataTable。
SQLResultSet 的定义为：

```
interface SQLResultSet {
        readonly attribute long insertId;
        readonly attribute long rowsAffected;
        readonly attribute SQLResultSetRowList rows;
};
```

其中最重要的属性—SQLResultSetRowList 类型的 rows 是数据集的“行” 。
rows 有两个属性：length、item 。
故，获取查询结果的第一行列名为name的值 ：result.rows.item(0).name 。

更新数据

```
this.update = function (id, name) {
 dataBase.transaction(function (tx) {
 tx.executeSql( "update stu set name = ? where id= ?", [name, id],
function (tx, result) { },
function (tx, error) { alert('更新失败: ' + error.message);
});
});
}
```

删除数据

```
this.del = function (id) {
dataBase.transaction(function (tx) {
tx.executeSql(
"delete from stu where id= ?",
[id],
function (tx, result) {
},
function (tx, error) {
alert('删除失败: ' + error.message);
});
});
}
```

删除数据表

```
this.dropTable = function () {
      dataBase.transaction(function (tx) {
       tx.executeSql('drop table stu');
      });
}
```

常见问题

Q1：数据存储在哪儿？

Web Storage / Web SQL Database / Indexed Database 的数据都存储在浏览器对应的用户配置文件目录(user profile directory)下，以 Windows 7 为例，Chrome 的数据存储在" C:\Users*your-account-name*\AppData\Local\Google\Chrome\User Data\Default\"下，而 Firefox 的数据存储在" C:\Users*your-account-name*\AppData\Local\Mozilla\Firefox\Profiles\"目录下。

Q2：卸载浏览器之后数据还在不在？

如果你在卸载浏览器时**主动**勾选了同时删除个人数据的选项（如下图所示），那么用户配置文件目录就会被整个删除。当然所有的数据也就不存在了。

如果没有勾选这项的话，下次安装此浏览器后，会发现存储的数据还在。

Q3：存储的数据是否安全？

我觉得很难去界定这些新的存储技术是"安全的"还是"不安全的"，只能说这些新的存储技术并没有增加更多的安全隐患，他们并没有比传统的 cookies 更安全，但也不会更危险。跨站攻击依然是一个隐患。而 Sandbox 也不会这些数据进行保护。所以我们也希望 Web Application 的开发者们在开发的时候就考虑到这样的问题。