本地存储

管理科学与工程学科
耿方方

主要内容

- 本地存储
- Web Storage
- IndexedDB
- Cookie
- 案例：使用本地数据提升服务器性能

1、本地存储

- 客户端本地存储数据即浏览器存储数据。
- 从HTML4中已经存在的Cookie存储机制到HTML5的Web Storage存储机制和本地数据库，目前本地存储技术主要由Web Storage、本地数据库和Cookie组成。
- 本地存储技术让永久数据管理这一完全由服务器端执行的工作也能够在客户端得以实现，从而大大减轻了服务器端的负担，加快访问速度。
-

1、本地存储

- Web Storage
- Web Storage把网站中有用的信息存储到本地，然后根据实际需要从本地读取信息。
- 主要分为Session Storage和Local Storage两种类型，其功能和用法基本上是相同的，只是保存数据的生存期限不同。
-

1、本地存储

- Storage对象的属性和方法具体如下所示。
- length：返回当前Storage对象里保存的键/值对数量。
- key(index)：该方法返回Storage中第index个键（key）的名称。
- getItem(key)：该方法返回指定key对应的当前值。
- setItem(key,value)：该方法首先检测指定的键/值对的键是否已存在于当前键/值对列表。
- removeItem(key)：该方法从Storage对象键/值对列表中删除指定键对应的数据项。
- clear()：当前Storage对象键/值对列表中有数据项时，清空键/值对列表。

1、本地存储

- 本地数据库
- 本地数据库是HTML提供的浏览器端数据库，可以在客户端存储大量结构化数据并直接通过JavaScript API高效检索，主要有Web SQL与IndexedDB两种类型。
- Web SQL是一种运行于浏览器的关系数据库。可以通过SQL语句执行数据的插入或检索等操作。
- IndexedDB是一种索引数据库，是一个不断发展中的网络标准，这个标准用于在浏览器中存储大量结构化的数据，并提供索引以保证高效率的查询。

1、本地存储

- Cookie
- Cookie是HTML4中已经存在的本地存储机制，多用于网站辨识用户身份、进行session跟踪而储存在用户本地终端上，以key-value形式进行存储数据，浏览器存储Cookie大小有限。

2、Web Storage

- 数据存储的实现
- sessionStorage是Storage对象的一个实例，浏览器中会话级别的WebStorage，对应Windows对象sessionStorage属性。
- W3C组织为sessionStorage制定的接口定义如下所示。

```
[NoInterfaceObject]
interface WindowSessionStorage {
   readonly attribute Storage sessionStorage;
};
Window implements WindowSessionStorage;
```

2、Web Storage

- 数据存储的实现
- 案例21-01：sessionStorage示例演示程序

```
<script type="text/javascript">
    //存储简单数据
    function saveSimpleString() {
                    var xs1=document.getElementById("xs");

        sessionStorage.setItem("course", "Web前端技术开发与实践");
                xs1.value=sessionStorage.getItem("course");
                document.write(xs1.value);
    }
    //存储结构化数据
    function saveStructuredData() {
        var studentInfo = [
        { "courseNum": "0", "name": "学号" },
        { "courseNum": "1", "name": "姓名" },
        { "courseNum": "2", "name": "性别" },
]
        sessionStorage.setItem("studentInfo", JSON.stringify(studentInfo));
    }
</script>
```

2、Web Storage

- 创建数据项
- sessionStorage和localStorage都将数据存储为项，项采用键/值组合的格式，每个值在存储前都要转化为字符串，sessionStorage继承Storage对象。
- sessionStorage创建数据项使用setItem()方法，具体如下所示。
- 创建数据项，存储简单字符串。

```
//存储简单数据
function saveSimpleString() {
    sessionStorage.setItem("course", "Web 前端技术开发与实践");
}
```

2、Web Storage

- 创建数据项，存储结构化数据。使用JSON.stringify()方法将复杂的JSON数据对象转换为字符串，再用setItem()方法保存到本地。

```
//存储结构化数据
function saveStructuredData() {
    var studentCourse = [
    { "courseNum": "0", "name": "离散数学" },
    { "courseNum": "1", "name": "高级语言程序设计" },
    { "courseNum": "2", "name": "算法与数据结构" },
    { "courseNum": "3", "name": "编译技术" },
    { "courseNum": "4", "name": "软件工程" },
    { "courseNum": "5", "name": "软件文档规范与标准" }
    ]
    sessionStorage.setItem("courseList", JSON.stringify(studentCourse));
}
```

2、Web Storage

- 读取数据
- 读取sessionStorage数据使用getItem()方法、key()方法和length属性，使用以及效果演示如下所示。
- 使用length属性获取当前数据项数目。

```
alert(sessionStorage.length);
```

- 使用key()方法输出指定编号数据项的键名称。

```
alert(sessionStorage.key(0));
alert(sessionStorage.key(1));
```

2、Web Storage

- 使用getItem()方法输出数据项的值。

```
alert(sessionStorage.getItem("course"));
alert(sessionStorage.getItem("sdudentInfo"));
```

- 控制台执行上述代码，依次输出已添加的两个数据项的值：

```
前端技术开发与实践
[{"courseNum":"0","name":"学号"},{"courseNum":"1","name":"姓名"},
{"courseNum":"2","name":"性别"},{"courseNum":"3","name":"专业"},
{"courseNum":"4","name":"Web 前端开发"},{"courseNum":"5","name":"操作系统"},
{"courseNum":"6","name":"网络"},{"courseNum":"7","name":"数据库"}]
```

2、Web Storage

- 删除数据
- 删除sessionStorage采用removeItem()方法和clear()方法，具体如下所示。
- 调用removeItem()方法删除一个数据项，并输出sessionStorage数据项数目和第一项的内容。

```
sessionStorage.removeItem("course");
alert(sessionStorage.length);
alert(sessionStorage.key(0));
```

- 调用clear()方法清空sessionStorage数据项列表，并输出结果。

```
sessionStorage.clear();
alert(sessionStorage.length);
```

2、Web Storage

- 数据存储的实现
- localStorage是Storage对象的一个实例，对应windows对象localStorage属性。
- W3C组织为localStorage制定的接口定义如下所示。

```
[NoInterfaceObject]
interface WindowLocalStorage {
  readonly attribute Storage localStorage;
};
Window implements WindowLocalStorage;
```

2、Web Storage

案例21-02：localStorage示例演示程序

```
<script type="text/javascript">
    //存储简单数据
    function saveSimpleString() {
        localStorage.setItem("course", "Web前端技术开发与实践");
    }
    //存储结构化数据
    function saveStructuredData() {
        var studentInfo = [
        { "courseNum": "0", "name": "学号" },
        { "courseNum": "1", "name": "姓名" },
        { "courseNum": "2", "name": "性别" },
        { "courseNum": "3", "name": "专业" },
        { "courseNum": "4", "name": "Web前端开发" },
        { "courseNum": "5", "name": "操作系统" },
        { "courseNum": "6", "name": "网络" },
        { "courseNum": "7", "name": "数据库" }
        ]
        localStorage.setItem("studentInfo", JSON.stringify(studentInfo));
    }
</script>
```

2、Web Storage

- 创建数据项
- 创建localStorage数据项使用setItem()方法，具体如下。
- 创建数据项，存储简单字符串；
- 创建数据项，存储结构化数据。

2、Web Storage

- 读取数据
- 与读取localStorage数据项有关的方法属性有getItem()方法、key()方法和length属性，具体如下。
- 使用length属性获取当前数据项数目。
- 使用key()方法输出指定编号数据项的键名称。
- 使用getItem()方法输出数据项的值。

2、Web Storage

- 删除数据
- 删除localStorage数据项采用removeItem()方法和clear()方法，具体如下。
- 调用removeItem()方法删除一个数据项，并输出sessionStorage数据项数目和第一项的内容。
- 调用clear()方法清空localStorage数据项列表，并输出相应的内容。

2、Web Storage

- 区别
- Session Storage是将数据保存在会话中，在会话期间内有效，浏览器关闭后，数据消失。
- LocalStorage是将数据存储在磁盘中，除非用户或程序进行主动清除，否则仅仅关闭浏览器是不会造成数据丢失的。
- 联系
- SessionStorage和LocalStorage都是Storage对象的实例，使用相同的API，数据操作方式基本相同，都是以键/值对存储数据。

2、Web Storage

案例21-03：

```
<script type="text/javascript">
    saveSimpleString();
    saveStructuredData();
    //存储简单数据
    function saveSimpleString() {
        sessionStorage.setItem("course", "Web前端技术开发与实践");
    }
    //存储结构化数据
    function saveStructuredData() {
        var studentInfo = [
        { "courseNum": "0", "name": "学号" },
        { "courseNum": "1", "name": "姓名" },
        { "courseNum": "2", "name": "性别" },
]
        sessionStorage.setItem("studentInfo", JSON.stringify(studentInfo));
        console.log(studentInfo);
    }
</script>
```

3、IndexedDB

- 数据库
- 数据库本身简单，不需要考虑用户关联或其他形式的访问限制，只需要指定名称和版本，数据库就处于就绪状态。
- 数据库对象用来管理数据库中的对象。
- IndexedDB仅具有在同一个源中执行的程序所共享的空间。
- 在一个源所拥有的空间中可以创建多个数据库，而在一个数据库中又可以创建多个对象存储（ObjectStore）。

3、IndexedDB

- 对象与对象库
- 关系数据库中称为记录的内容，在IndexedDB中称为对象。对象带有属性，用来存储键值和标识值。
- 属性的个数与对象的结构无关。对象的唯一要求是至少包含一个声明为索引的属性。
- 对象库，又叫对象存储，对应关系数据库中的数据表，没有固定的结构，可以保存任意类型的JavaScript对象。

3、IndexedDB

- IndexedDB提供的操作对象库的方法如下：
- createObjectStore(name, keyPath, autoIncrement)：用于属性指定名称和配置集合来新建一个对象库。
- objectStore(name)：要访问对象库中的对象，必须启动一个事务，并为这个事务打开对象库，该方法打开name属性名称指定的对象库。
- deleteObjectStore(name)：这个方法删除name属性声明名称指定的对象库。

3、IndexedDB

- IndexedDB操作对象的方法如下：
- add(object)：该方法接受关键字/值组合或包含多个关键字/值组合的对象。
- put(object)：该方法与前一个方法类似，在对象库中已经存在与索引相同的对象时，覆盖索引相同的对象。
- get(key)：该方法可以从对象库中获取指定对象，Key属性是要获取的对象索引值。
- delete(key)：要在选中的对象库中删除某个对象，用该对象的索引值作为属性调用该方法。

3、IndexedDB

- 索引
- 要在对象库中寻找对象，需要将这些对象的某些属性设置为索引，简单的做法如下：
- 在createObjectStore()方法中声明keyPath属性。
- 声明为keyPath的属性就成为对象库中存储的每个对象的公共索引。
- 设置keyPath时，要确保每个对象中都有这个属性。
- 可以为保存的JavaScript对象的任意属性创建索引，且可以同时创建多个索引。通过索引的创建，能够对相应的属性进行指定范围的高速检索。

3、IndexedDB

- 事务
- 在浏览器上工作的数据库系统必须面对一些其他平台上没有的特殊情况。
- IndexedDB可以使用事务功能。在进行数据库的版本更改等处理时，将会在内部自动地创建事务。
- IndexedDB事务拥有数据库操作的一个中止和提交工具。

3、IndexedDB

3.1存储原理

案例21-04：IndexedDB示例演示程序

```
        <script type="text/javascript">//连接/创建数据库
            var request = indexedDB.open('myDatabase');
            //连接数据库成功
            request.onsuccess = function (event) {
                //使之可以通过全局变量引db引用
                var db = event.target.result;
                //数据库操作程序
                var veisionSet = db.setVersion('1.0');
                veisionSet.onsuccess = function (event) {
                    //创建对象存储
                    var store = db.createObjectStore('course', {
                        keyPath: '_id',
                        autoIncrement:true
                    })                                                  alert('数据库创建成功');
                }
            };
request.onerror = function (event) {
                alert('连接数据库失败！');
            }


        </script>
```

3、IndexedDB

- 数据操作
- 连接数据库的方法和更新版本的方法
- IndexedDB使用异步API，并不是这条指令执行完毕，就可以使用request.result来获取indexedDB对象，语句执行完并不代表已经获取到了对象，所以一般在其回调函数中处理。
- 数据操作的过程具体如下：
- 创建对象存储
- 数据写入
- 数据读取
- 数据修改
- 数据删除
- 数据检索

3、IndexedDB

3.1存储原理

- 创建对象存储

```
function openDB(name, version, arrStoreName) {
    var version = version || 1;
    var request = window.indexedDB.open(name, version);
    request.onerror = function (e) {
        console.log(e.currentTarget.error.message);
    };
    request.onsuccess = function (e) {
        db = e.target.result;
        console.log('DB version changed to ' + version);
    };
```

```
request.onupgradeneeded = function (e) {
    db = e.target.result;
    for (var i = 0; i < arrStoreName.length; i++) {
      var storeName = arrStoreName[i];
      if (!db.objectStoreNames.contains(storeName)) {
        var store = db.createObjectStore(storeName, {keyPath: "id"});
        store.createIndex('nameIndex', 'name', {unique: false});
      }
    }
    console.log('DB version changed to ' + version);
  };
}
```

在控制台中调用一下

```
openDB('studentScore',1,['scoreList']);
```

3、IndexedDB

3.1存储原理

- 数据写入
- 在对新数据库做任何事情之前，需要开始一个事务。事务中需要指定该事务跨越哪些object store。调用数据库实例的transaction方法打开事务，通过事务获取对象存储实例，调用对象存储实例的add方法添加数据。具体如下所示。

```
function addData(db, storeName, data) {
    var transaction = db.transaction(storeName, 'readwrite');
    //读写方式打开事务
    var store = transaction.objectStore(storeName);
    //获取对象存储
    for (var i = 0; i < data.length; i++) {
        var request = store.add(data[i]);
        //调用add方法添加数据
        request.onsuccess = function (event) {
            console.log('addData success! key', event.target.result);
        }
    }
}
```

3、IndexedDB

3.1存储原理

- 数据读取
- 通过游标遍历存储对象，输出对象到控制台日志。调用数据库实例的transaction方法打开事务，通过事务获取对象存储实例，调用对象存储实例的openCursor方法打开游标，cursor.contine()会使游标下移，直到没有数据时返回undefined。具体如下所示。

```
function fetchStoreByCursor(db, storeName) {
    var transaction = db.transaction(storeName);
    var store = transaction.objectStore(storeName);
    var request = store.openCursor();
    request.onsuccess = function (e) {
        var cursor = e.target.result;
        if (cursor) {
            var currentStudent = cursor.value;
            console.log(currentStudent);
            cursor.continue();
        }
    };
}
```

3、IndexedDB

- 数据修改
- 调用数据库实例的transaction方法打开事务，通过事务获取对象存储实例，调用对象存储实例的get方法读取数据，修改处理后，调用put方法返回对象存储。具体如下所示。

```
function updateDataByKey(db, storeName, key, value) {
    var transaction = db.transaction(storeName, 'readwrite');
    var store = transaction.objectStore(storeName);
    var request = store.get(key);
    request.onsuccess = function (e) {
        var student = e.target.result;
        student = value;
        store.put(student);
    };
}
```

3、IndexedDB

- 数据删除
- 通过键值删除数据对象
- 调用数据库实例的transaction方法打开事务，通过事务获取对象存储实例，调用对象存储实例的delete方法根据key删除对象。具体如下所示。

```
function deleteDataByKey(db, storeName, key) {
    var transaction = db.transaction(storeName, 'readwrite');
    var store = transaction.objectStore(storeName);
    store.delete(key);
}
```

3、IndexedDB

- 数据检索
- ①通过键值检索对象
- 调用数据库实例的transaction方法打开事务，通过事务获取对象存储实例，调用对象存储实例的get方法读取数据。具体如下所示。

```
function getDataByKey(db, storeName, value) {
    var transaction = db.transaction(storeName, 'readwrite');
    var store = transaction.objectStore(storeName);
    var request = store.get(value);
    request.onsuccess = function (e) {
        var student = e.target.result;
        console.log(student);
    };
}
```

3、IndexedDB

- 数据检索
- 通过索引检索对象
- 调用数据库实例的transaction方法打开事务，通过事务获取对象存储实例，调用对象存储实例的index方法读取索引对象，最后通过索引对象的get方法读取数据。具体如下所示。

```
function getDataByIndex(db, storeName, indexName) {
    var transaction = db.transaction(storeName);
    var store = transaction.objectStore(storeName);
    var index = store.index("nameIndex");
    index.get(indexName).onsuccess = function (e) {
        var student = e.target.result;
        console.log(student);
    };
}
```

4、Cookie

- 什么是Cookie
- Cookie是由服务器端生成，发送给User-Agent，通常是浏览器。
- Cookie的名称和值可以由服务器端开发者自己定义，这样服务器可以知道该用户是否是合法用户以及是否需要重新登录等，服务器可以设置或读取Cookies中包含的信息，借此维护用户跟服务器会话中的状态。

4、Cookie

- Cookie数据存储位置与格式
- Cookie是个存储在浏览器目录的文本文件，当浏览器运行时，存储在RAM中。
- 一旦用户从该网站或网络服务器中退出，Cookie也可存储在计算机的硬件驱动上。
- cookie是以键值对的形式保存的，即key=value的格式。
- cookie语法如下：

```
set-Cookie:name=value;[expirse=date];[path=dir];[domain=domainname];[secure]
```

- 各个cookie之间一般是以“;”分隔。

4、Cookie

- Cookie的主要用途：
- 记录访客的某些信息
- 例如：访问网页的次数，或者记录访客曾经输入过的信息。
- 跟踪用户行为
- 可以根据用户的IP地址报告当前此用户所在地的天气情况等。
- 创建购物车
- 记录用户曾经浏览过的相关信息，最后在结账时一起提交。

4、Cookie

- Cookie的工作流程：
- Cookie是在浏览器访问Web服务器的某个资源时，由Web服务器在HTTP响应消息头文件中附带传递给浏览器的一些数据。如果浏览器保存了这些数据，当它每次访问该Web服务器时，都应在HTTP请求头文件中将这些数据传回给Web服务器。Web服务器将这些数据在HTTP请求头文件中使用set-Cookie响应头字段将Cookie信息发送给浏览器，浏览器则通过在HTTP请求中增加Cookie请求字段将Cookie回传给Web服务器。

4、Cookie

- 创建Cookie
- 创建Cookie方法具体如下：
- setcookie(name,value,expire,path,domain,secure)：该方法有6个参数。
- name规定cookie的名称，是必需参数。
- value规定cookie的值，是必需参数。
- expire规定cookie有效期。
- path规定cookie的服务器路径，是可选参数。
- domain规定cookie的域名，是可选参数。
- secure规定是否通过安全的HTTPS连接来传输cookie，是可选参数。

4、Cookie

- 创建简单Cookie：

```
setcookie('username', 'sdsd');
```

- PHP创建一个5分钟过期的cookie：

```
setcookie('cookietest', '5 分钟有效', time() + 5 * 60);
exit();
```

4、Cookie

- 浏览器中的Cookie
- 访问该网页打开浏览器调试界面，存储界面下Cookie存储列表中出现“’username’”项，最后访问时间是当前时间，过期时间为“会话”；cookietest项，最后访问时间为当前时间，过期时间为当前时间后延5分钟。
- JavaScript创建简单cookie，其使用的方法为：document.cookie=″name=″+value;。

```
document.cookie="user=demo";
```

4、Cookie

- JavaScript创建一个5分钟过期的cookie：

```
/*设置 cookie*/
function addCookie(name, value, expireHours) {
    var cookieString = name + "=" + escape(value);
    //判断是否设置过期时间
    if (expireHours) {
        var date = new Date();
        date.setTime(date.getTime() + expireHours * 3600 * 1000);
        cookieString = cookieString + ";    expire=" + date.toGMTString();
    }
    document.cookie = cookieString;
}
```

4、Cookie

- 数据修改
- setcookie()，当cookie名称已存在，且新值与旧值不同时，更新cookie：

```
setcookie('username', 'change');
exit();
```

- JavaScript修改cookie：

```
document.cookie="user=demo1";
```

4、Cookie

- Cookie读取
- PHP通过 $HTTP_COOKIE_VARS[“user”] 或 $_COOKIE[“user”] 来访问名为“user” 的 cookie 的值:

```
//读取并输出 cookie
echo $_COOKIE['username'];
echo '<br/>';
echo $_COOKIE['cookietest'];
exit();
```

- JavaScript读取Cookie:

```
//获取指定名称的 cookie 值
function getCookie(name) {
    var arr = document.cookie.match(new RegExp("(^| )" + name + "=([^;]*)(;|$)"));
    if (arr != null)
    return unescape(arr[2]);
    return null;
}
```

4、Cookie

- 清除Cookie
- PHP清除Cookie:

```
//清除 cookie
setcookie('cookietest', '5 分钟有效', time());
echo $_COOKIE['cookietest'];
```

- JavaScript清除Cookie:

```
//删除 cookies
function delCookie(name) {
      var exp = new Date();
      exp.setTime(exp.getTime();
      var cval = getCookie(name);
      if (cval != null){
       document.cookie = name + "=" + cval + ";expires=" + exp.toGMTString();
      }
}
```

- 案例21-05：页面重定向的问题
- 案例21-06：广告弹出问题