Rack 的#1748 PR（合并的PR）

**Clarify streaming response body behavior in SPEC(在SPEC中阐明流响应内容体的行为)**

From wjordan:

这个PR来自#1745的讨论关于如何在Rack接口中区分流和缓存的内容体(streaming and buffered bodies)。在这里提议了为这个标准增加两句话：

字符串一定要单独处理。

然而，如果内容体响应为to\_ary，它会被暗含的强制的转化为数组，然后再被一次性处理。

我认为这对接口来说更像是‘阐明’（clarification）而不是一个大的改变，正如以前的讨论和实现所建议的，把Array内容体（或者通过#to\_ary强制转化为Array的对象）看作一个固定长度的内容，与之相反的是其它的响应主体可能是内容长度不确定的。

比如，看来自2011年5月的#166(评论)：

尽管to\_ary没有在rack spec提及，一个内容体是否是一个数组取决与内容体的大小是已知的还是未知的。在chunked和non-chunked转化编码之间会被考虑。

这个PR也更新了ContentLength和ETag中间件。它展示了这个标准是如何在实践中指导对’buffering’中间件/服务端进行一个更加一致的实现。有三个步骤：

1. 检查body.respond\_to?(:to\_ary),如果响应没有强制转化为Array,就跳过buffering/transformation逻辑。
2. 调用body = body.to\_ary,强制转化响应内容体来确保它是一个字符串数组。
3. 按需操作这个内容体（如计算SHA256 digest，计算字符串字节大小）。

一些注释：

* 注意到使用这个模式，在中间件中BodyProxy不再需要，因为Array对象从来不会响应close。（如果#close被一个可以被转化为Array的对象需要，它应该作为to\_ary的实现的一部分来关闭。比如，看这个更新’close bodies that need to be close。）
* Web服务器可能也会采用这个模式来作为一个优化，在写进socket之前把内容缓存为一个字符串（或者通过调用#write来写入一个字符串数组），来减少socket写入的次数。
* 它不会影响并且和hijack APIs（#481）是无关的，

TODO:待翻译

以下是修改的代码：

总结：

这个PR更新了下rack标准，还有ContentLength和Etag中间件。即文件SPEC.rdoc,lib/rack/content\_length.rb,lib/rack/etag.rb,test/spec\_content\_length.rb,test/spec\_etag.rb。

没看明白这个PR改动的意义。