```
        $server_freshness_limit = $default_cache_min_lifetime;
        $heuristic = 1;
    }

    if ($heuristic)
    {
        if ($server_freshness_limit > $default_cache_max_lifetime)
        { $server_freshness_limit = $default_cache_max_lifetime; }
        if ($server_freshness_limit < $default_cache_min_lifetime)
        { $server_freshness_limit = $default_cache_min_lifetime; }
    }

    return($server_freshness_limit);
}
```

193

现在，我们来看看客户端怎样修正服务器为文档指定的使用期限制。例 7-3 显示了一个 Perl 算法，此算法获取了服务器的新鲜度限制并根据客户端的限制对其进行修改。它会返回一个最大使用期，这是在无需再次验证，仍由缓存提供文档的前提下，文档的最大生存时间。

例 7-3　客户端新鲜度限制的计算

```
sub client_modified_freshness_limit
{
    $age_limit = server_freshness_limit( ); ## From Example 7-2

    if ($Max_Stale_value_set)
    {
        if ($Max_Stale_value == $INT_MAX)
        { $age_limit = $INT_MAX; }
        else
        { $age_limit = server_freshness_limit( ) + $Max_Stale_value; }
    }

    if ($Min_Fresh_value_set)
    {
    $age_limit = min($age_limit, server_freshness_limit( ) -
           $Min_Fresh_value_set);
    }

    if ($Max_Age_value_set)
    {
        $age_limit = min($age_limit, $Max_Age_value);
    }
}
```

整个进程中包含两个变量：文档的使用期及其新鲜度限制。如果使用期小于新鲜度限制，就说明文档“足够新鲜”。例 7-3 中的算法只是考虑了服务器的新鲜度限制，并根据附加的客户端限制对其进行了调整。希望通过本节的介绍能使在 HTTP 规范中描述的比较微妙的过期算法更清晰一些。