https://www.cnblogs.com/zhaoyihao/p/5345784.html

[Js参数RSA加密传输，jsencrypt.js的使用](https://www.cnblogs.com/zhaoyihao/p/5345784.html)

### 注意几点：

　　1、参数传递的+号处理，在传输时会把+变成空格，不处理后端就报错了。

### 1、前端代码

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Login</title>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script src="http://passport.cnblogs.com/scripts/jsencrypt.min.js"></script>

<script type="text/javascript">

$(function () {

var encrypt = new JSEncrypt();

encrypt.setPublicKey($("#tra").val());

var data = encrypt.encrypt("123456789");

alert(data);

$("#btn").click(function () {

$.ajax({

url: '@Url.Action("Login")',

data: "pwd=" + encodeURI(data).replace(/\+/g, '%2B'), //+号的处理：因为数据在网络上传输时，非字母数字字符都将被替换成百分号（%）后跟两位十六进制数，而base64编码在传输到后端的时候，+会变成空格，因此先替换掉。后端再替换回来

type: 'post',

success: function (msg) {

alert(msg);

}

});

});

});

</script>

</head>

<body>

<div>

<input type="button" id="btn" value="点我" />

<textarea id="tra" rows="15" cols="65">

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCa4KHNwDX44gGmmIAtRu4gjVYt

GWZzcm4t+1wjUD4dn7fMLPvuK7ai4UrfDeEJE1RPwudJw+lJ6crql8wSIg7/DbTl

G3ihsCT6dT9H5B9OoeR7K9VWUesaW/iyVL6HXiYOANabW14pvJATDmdq91Tfgp6P

SQyvdfiRdV4r07crpQIDAQAB

</textarea>

<hr/>

注意+好的处理

</div>

</body>

</html>

[复制代码](javascript:void(0);)

### 2、后端代码

[复制代码](javascript:void(0);)

public class IndexController : Controller

{

public ActionResult Login()

{

return View();

}

[HttpPost]

public ActionResult Login(string pwd)

{

//密钥格式要生成pkcs#1格式的 而不是pkcs#8格式的

string privateKey = @"MIICWwIBAAKBgQCa4KHNwDX44gGmmIAtRu4gjVYtGWZzcm4t+1wjUD4dn7fMLPvuK7ai4UrfDeEJE1RPwudJw+lJ6crql8wSIg7/DbTlG3ihsCT6dT9H5B9OoeR7K9VW

UesaW/iyVL6HXiYOANabW14pvJATDmdq91Tfgp6PSQyvdfiRdV4r07crpQIDAQAB

AoGABb+3gdb+qeG0b1CogVsT/7//UOaTzPk/FGneKQQTf4SsN+H7lVhTYTG9ARFC

JyoWg8IXqmn2ljhywHPTWWD2RCZIn2sYT1sVkGb70EgHGQLBraFHElmw+DsVJ+nD

fBCfMrJ1TYXlwigjRkaueaoGgG8LdR8XD+Xs5LersPLjZgECQQCguSB7C4wF6oSw

EDmwNF8ffT5cQc1U2OIq6NBG8rafrjb7LsjhOd03pmY7i4LbW3Vvq4AhQpJEdF1C

vd+Sk/BBAkEA9rBhqnyumV09zFEomSX3zZu+bdhTzM4bJDfEa95swp1gANCVvF/t

DCnlBf51EhCWdeGSpARPUkQnXrYfFUDiZQJAAZEshuaa6+fYeVr/JP+tucHf3Mhr

dxtSQTbZ6QcuzqnFMXfIT6HfzU4bCxOWKAthPsB+VFSw1mgIDMGLL4OvwQJAJlVy

V9PYLezXVZCnBmVoBINXLCqZmxHMFey0kS6XKAbcjEPdgNBHPcSk2jGYb540Q00y

RFqHGPmORKF4Yw0aIQJAd5JRtD3z2MgP/vPoKHJNHqY8bboVcmwqVAm6xCZoTCZz

jNV1Cnsdf4wBV3LCDzYBy+xR4qYNUy5CFXN+8WzzAA==";

try

{

RSACryptoServiceProvider rsaCryptoServiceProvider = CreateRsaProviderFromPrivateKey(privateKey);

//把+号，再替换回来

byte[] res = rsaCryptoServiceProvider.Decrypt(Convert.FromBase64String(pwd.Replace("%2B","+")), false);

return Content(Encoding.UTF8.GetString(res));

}

catch (Exception exception)

{

}

return Content("");

}

private RSACryptoServiceProvider CreateRsaProviderFromPrivateKey(string privateKey)

{

var privateKeyBits = System.Convert.FromBase64String(privateKey);

var RSA = new RSACryptoServiceProvider();

var RSAparams = new RSAParameters();

using (BinaryReader binr = new BinaryReader(new MemoryStream(privateKeyBits)))

{

byte bt = 0;

ushort twobytes = 0;

twobytes = binr.ReadUInt16();

if (twobytes == 0x8130)

binr.ReadByte();

else if (twobytes == 0x8230)

binr.ReadInt16();

else

throw new Exception("Unexpected value read binr.ReadUInt16()");

twobytes = binr.ReadUInt16();

if (twobytes != 0x0102)

throw new Exception("Unexpected version");

bt = binr.ReadByte();

if (bt != 0x00)

throw new Exception("Unexpected value read binr.ReadByte()");

RSAparams.Modulus = binr.ReadBytes(GetIntegerSize(binr));

RSAparams.Exponent = binr.ReadBytes(GetIntegerSize(binr));

RSAparams.D = binr.ReadBytes(GetIntegerSize(binr));

RSAparams.P = binr.ReadBytes(GetIntegerSize(binr));

RSAparams.Q = binr.ReadBytes(GetIntegerSize(binr));

RSAparams.DP = binr.ReadBytes(GetIntegerSize(binr));

RSAparams.DQ = binr.ReadBytes(GetIntegerSize(binr));

RSAparams.InverseQ = binr.ReadBytes(GetIntegerSize(binr));

}

RSA.ImportParameters(RSAparams);

return RSA;

}

private int GetIntegerSize(BinaryReader binr)

{

byte bt = 0;

byte lowbyte = 0x00;

byte highbyte = 0x00;

int count = 0;

bt = binr.ReadByte();

if (bt != 0x02)

return 0;

bt = binr.ReadByte();

if (bt == 0x81)

count = binr.ReadByte();

else

if (bt == 0x82)

{

highbyte = binr.ReadByte();

lowbyte = binr.ReadByte();

byte[] modint = { lowbyte, highbyte, 0x00, 0x00 };

count = BitConverter.ToInt32(modint, 0);

}

else

{

count = bt;

}

while (binr.ReadByte() == 0x00)

{

count -= 1;

}

binr.BaseStream.Seek(-1, SeekOrigin.Current);

return count;

}

}