服务端 拦截器

拦截器应用于终结点行为上

拦截器代码

public class LoginInspector : IDispatchMessageInspector

{

public object AfterReceiveRequest(ref Message request, IClientChannel channel, InstanceContext instanceContext)

{

int n = OperationContext.Current.IncomingMessageHeaders.FindHeader("AuthTicket", "System");

if (n >= 0)

{

string ticket = OperationContext.Current.IncomingMessageHeaders.GetHeader<string>("AuthTicket", "System");

GeneratePrincipal.Generate(ticket);

}

return DateTime.Now;

}

public void BeforeSendReply(ref Message reply, object correlationState)

{

}

}

终结点行为上绑定拦截器

public class BaseEndpointBehavior : IEndpointBehavior

{

public void AddBindingParameters(ServiceEndpoint endpoint, BindingParameterCollection bindingParameters)

{

}

public void ApplyClientBehavior(ServiceEndpoint endpoint, ClientRuntime clientRuntime)

{

}

public void ApplyDispatchBehavior(ServiceEndpoint endpoint, EndpointDispatcher endpointDispatcher)

{

endpointDispatcher.DispatchRuntime.MessageInspectors.Add(new LoginInspector());

}

public void Validate(ServiceEndpoint endpoint)

{

}

}

服务行为上绑定终结点行为

public class BaseServiceBehavior: Attribute, IServiceBehavior

{

public void AddBindingParameters(ServiceDescription serviceDescription,

ServiceHostBase serviceHostBase,

Collection<ServiceEndpoint> endpoints,

BindingParameterCollection bindingParameters)

{

}

public void ApplyDispatchBehavior(ServiceDescription serviceDescription,

ServiceHostBase serviceHostBase)

{

foreach (var item in serviceDescription.Endpoints) {

item.EndpointBehaviors.Add(new BaseEndpointBehavior());

}

}

public void Validate(ServiceDescription serviceDescription,

ServiceHostBase serviceHostBase)

{

}

}

客户端 拦截器

拦截器代码

public class LoginInspector : IClientMessageInspector

{

void IClientMessageInspector.AfterReceiveReply(ref Message reply, object correlationState)

{

}

object IClientMessageInspector.BeforeSendRequest(ref Message request, IClientChannel channel)

{

MessageHeader<string> ticket = new MessageHeader<string>(AuthTicket.GetTicket());

OperationContext.Current.OutgoingMessageHeaders.Add(ticket.GetUntypedHeader(MessageHeaderConfig.TicketName, MessageHeaderConfig.TicketNs));

return null;

}

}

终结点行为上添加拦截器

public class BaseEndpointBehavior: IEndpointBehavior

{

public void AddBindingParameters(ServiceEndpoint endpoint, BindingParameterCollection bindingParameters)

{

// 不需要

return;

}

public void ApplyClientBehavior(ServiceEndpoint endpoint, ClientRuntime clientRuntime)

{

// 植入“偷听器”客户端

clientRuntime.ClientMessageInspectors.Add(new LoginInspector());

}

public void ApplyDispatchBehavior(ServiceEndpoint endpoint, EndpointDispatcher endpointDispatcher)

{

// 不需要

return;

}

public void Validate(ServiceEndpoint endpoint)

{

// 不需要

return;

}

}

代理上添加中结点行为

proxy.Endpoint.EndpointBehaviors.Add(new BaseEndpointBehavior());