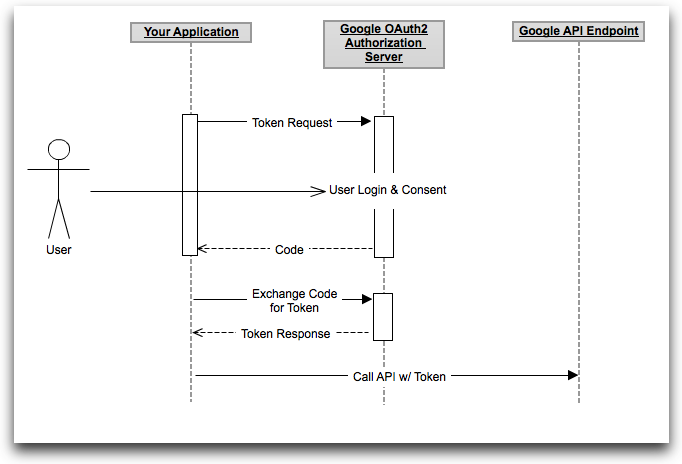
# GmailContacts

Esta aplicação, feita em ASP.NET MVC 3, irá mostrar contactos de um determinado utilizador do Google, existe um input na aplicação para indicar o email do qual pretendemos ver os contactos, é **obrigatório** que este email seja o mesmo da conta que está autenticada e irá autorizar a visualização dos contactos.

A aplicação usa OAuth2 na comunicação com a API da Google para obter autorização para aceder ao recurso, o processo de autorização segue o seguinte esquema:



Tudo acontece no ContactController, começando na acção ObtainAll, que irá fazer um redirect do dono do recurso para uma página da Google onde este irá ser perguntado se autoriza que a aplicação aceda ao recurso. Após a responsa a Google irá redireccionar de volta, para um URI que é indicado no primeiro redirect, caso o dono do recurso tenha autorizado iremos obter um *Autorization Grant* que iremos usar para obter um *Access Token* fazendo um novo pedido à Google, neste pedido é necessário indicar um redirect\_uri (embora não seja usado para fazer redirect) que tem que ser o mesmo que o indicado no primeiro pedido.

Obtendo o *Access Token* podemos então fazer o pedido para obter os contactos, enviando também o *Access Token*, sendo que a resposta irá vir em XML bastando então fazer o *parsing* destes dados e mostrar ao dono do recurso.

## Usar HTTPS

Para facilitar o processo foi trocado o servidor de testes pelo IIS Express, isto pode ser feito fazendo right-click no projecto e escolhendo a opção "Use IIS Express...". Começamos então a usar o IIS Express, agora é preciso que este aceite ligações SSL e para isso abre-se as propriedades do projecto e na opção "SSL Enabled" escolhe-se true.

Agora é preciso instalar o certificado na máquina, garantindo que temos a chave primária deste, e adicionar o certificado root como trusted. Estando os certificados instalados precisamos então de escolher o certificado que o IIS Express vai usar, para isso usamos os seguinte comandos:

* netsh http delete sslcert ipport=0.0.0.0:<SSLPort>
  + Este comando irá apagar o certificado que se encontra de momento na porta <SSLPort>.
* netsh http add sslcert ipport=0.0.0.0:44300 certstorename=MY certhash=<certificate hash> appid=<appid>
  + Este comando irá adicionar um novo certificado a essa porta, o valor appid pode ser um qualquer identificador único (por ex, {214124cd-d05b-4309-9af9-9cac44b2b74a}).

Por fim, é preciso adicionar o *callback* para *https* no painel de controlo das APIs do Google. E depois disso é preciso que a aplicação escolha este novo *callback* como *callback* a usar, a partir deste momento iremos estar a comunicar sobre um canal seguro.

# PolicyDecisionPoint utilizando RBAC1

No desenvolvimento deste componente .NET foi definido um modelo interno para representar os diversos intermediários:

* User
* Role
* Permission
* Session

A classe PolicyDecisionPoint é então onde tudo acontece, esta tem dois construtores um sem parâmetros e outro que recebe 6 enumeráveis de string que representam a seguinte informação:

* Users
* Roles
* Permissions
* Roles Heritage
* User Assignment
* Permission Assignment

No construtor sem parâmetros esta irá carregar a informação presente no ficheiro de configuração da aplicação, secção PDPPolicy, no construtor com parâmetros ele irá obter a informação que precisa dos enumeráveis de string.

Para interagir com o PolicyDecisionPoint tem que se criar uma sessão, existem duas formas, uma basta indicar o utilizador para o qual é a sessão e este irá atribuir todas as permissões associadas aos roles desse utilizador, na segunda para além do utilizador também se indica um role e as permissões associadas serão apenas as desse role. Para saber se uma determinada sessão tem permissões basta chamar o método HasPermission que recebe a sessão e as permissões necessárias, sendo que este irá então avaliar se esta sessão pode aceder a um recurso que necessite das permissões indicadas.