Programowanie Defensywne – Laboratorium

Walidacja oraz obsługa błędnych danych.

W progamowaniu defensywnym ważne jest by zdefiniować granice zaufania. Wszystkie dane, które pochodzą spoza tej granicy, powinny być dokładnie sprawdzone. Jeśli proces walidacji wykryje nieprawidłowość, wtedy należy odpowiednio obsłużyć taki przypadek. Najczęściej, rzuca się wyjątek (Exception), który najlepiej opisuje daną sytuację (np. ArgumentException, InvalidOperationException, itd.)

Zadanie

Serwis (VotingService.svc.cs) ma bezpośredni kontakt z zewnętrznymi danymi, którym nie można ufać.

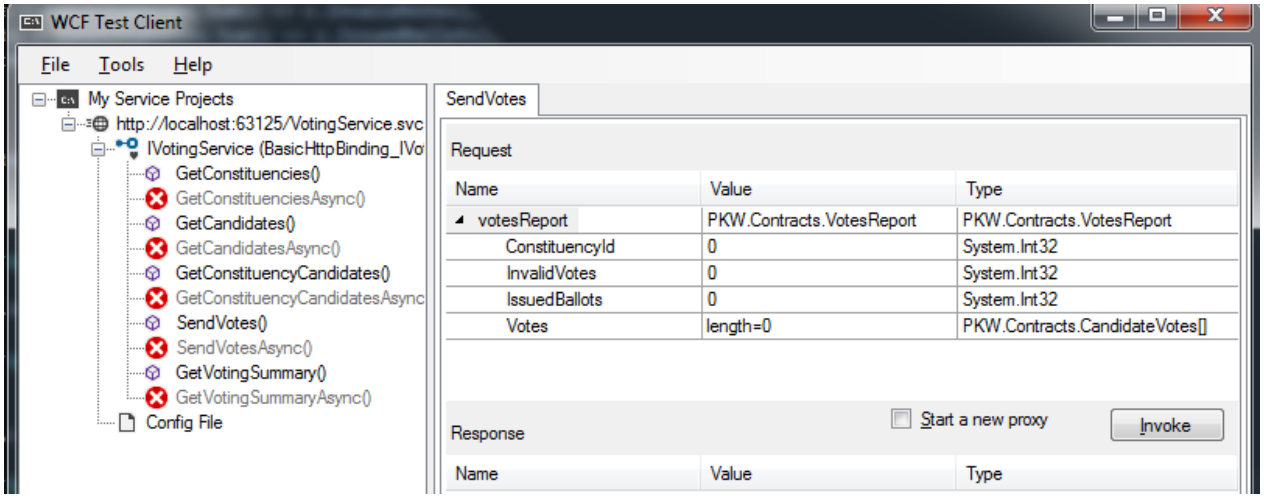
Napisz kod , który będzie wykonywał walidaję argumentów przesłanych w metodzie VotingService.SendVotes(VotesReport votesReport).
Postaraj się obsłużyć jak najwięcej błędnych sytuacji, np.:
- obiekty nigdy nie powinien być null
- id okręgu wyborczego wskazuje na nieistniejący obiekt.

W przypadku niepowodzenia walidacji odpowiedni wyjątek (typ wyjątku) powinien być rzucony.

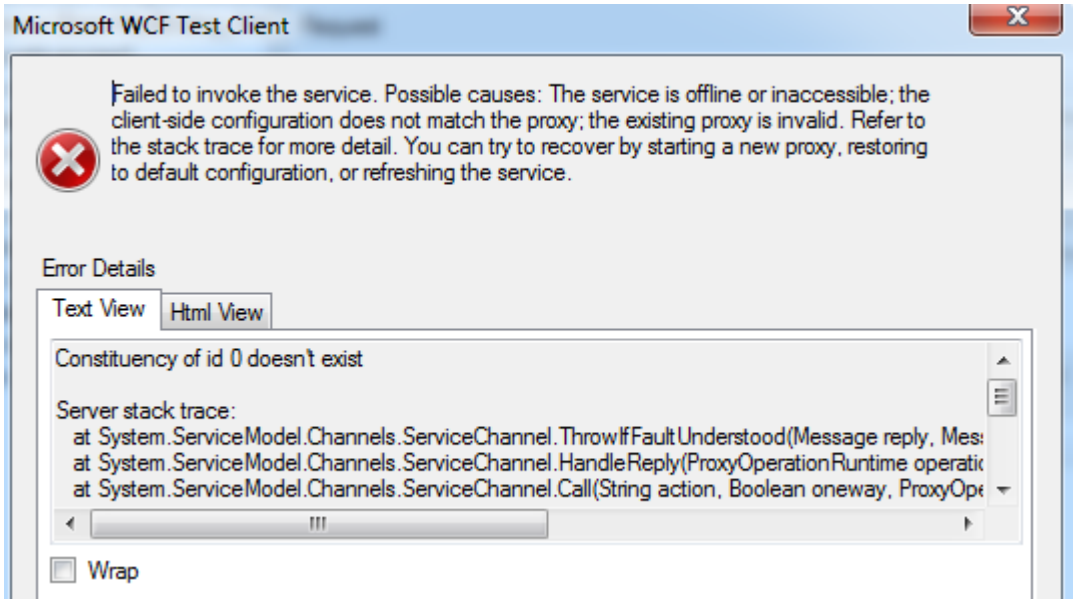
Uwaga: zostały wymienione tylko dwie przykładowe błędne sytuacje, niedopuszczalnych przypadków jest znacznie więcej.

Strategia obsługi błedów

W aplikacji WCF Test Client uruchom metodę SendVotes z błędnym argumentem ConstituencyId=0



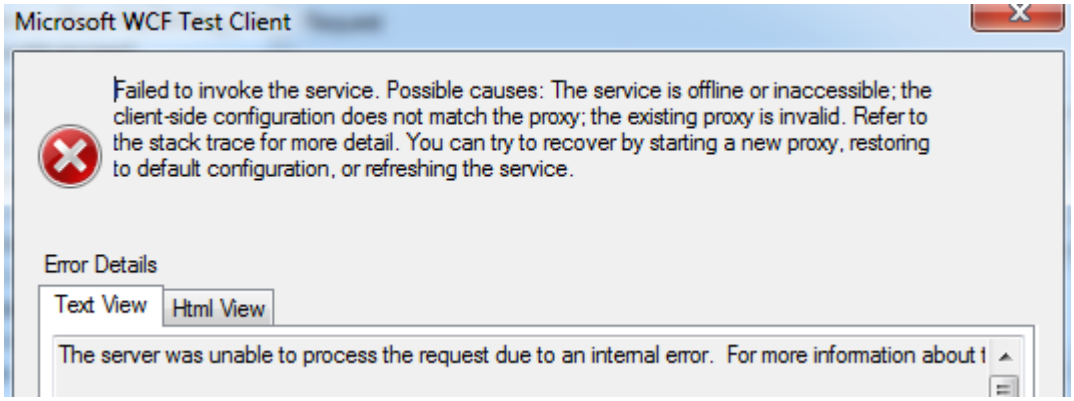
Ponieważ nie istnieje okręg wyborczy o ID=0, wyjątek zostaje zwrócony wraz z opisem.



W aplikacji produkcyjnej ze względów bezpieczeństwa nie powinny być ujawniane szczegóły występujących błędów (Exception).
W pliku web.config zmień wartość includeExceptionDetailInFaults na false.

<serviceDebug includeExceptionDetailInFaults="false" />

Wywołaj jeszcze raz SendVotes() z takimi samymi argumentami.
Tym razem serwis nie ujawnił żadnych szczegółów opisujących błąd.



System jest bardziej bezpieczny, ale też nie ujawnia użytkownikowi (nie przysyła do klienta) informacji jakie dane zostały źle wprowadzone.

WCF (SOAP) zawiera mechanizm obsługi błędów.
Przykładowo, klasa FaultException umożliwia przekazanie obisu błędu.

Zadanie
Zaprogramuj dla serwisu strategię chwytania wyjątku (try catch), która:
- Jeśli wystąpi wyjątek ArgumentException, wtedy rzucony jest nowy wyjątek FaultException. Opis błędu (Message) w FaultException powinien być taki sam jak w ArgumentException.
- Jeśli wystąpi inny typ wyjątku niż ArgumentException, wtedy należy rzucić wyjątek FaultException z generycznym opisem błędu (Message), np. „Wystąpił błąd w systemie”
- Taka sama strategia obsługi błędów powinna być zastsowana dla wszystkich metod, które znajdują się w interfejsie IVotingService .
- Implementacja nie powinna zawierać jak najmniej duplikującego się kodu.

W przypadku poprawnej implementacji, wywołania metody SendVotes (ConstituencyId=0) powinniśmy znów zobaczyć opis błędu.