UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

Tesi di Laurea

Esperimenti sul Testing di Rich Internet Applications

Anno Accademico 2010 / 2011

relatore
Ch.mo prof. Porfirio Tramontana

candidato
Roberto Niro
Matr. 885/177

Problematica

Analisi dinamica del comportamento di un'applicazione web di tipo “rich”, implementata utilizzando framework di programmazione

Obiettivi principali

- Individuazione di una strategia di testing di RIA sviluppate mediante framework
- Valutazioni sull'adozione di framework nel contesto di un processo produttivo aziendale

Framework: perchè ?

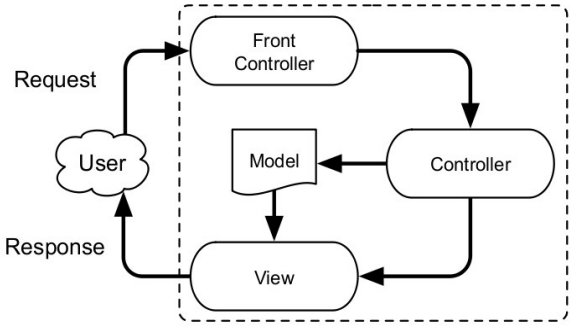
Un ***framework*** è un'architettura generica, un insieme di classi ed interfacce di base, che costituisce l'infrastruttura per lo sviluppo di applicazioni in una determinata area tecnologica.

Il codice applicativo non è direttamente invocato dall'intervento dell'utente sul sistema ma il flusso elaborativo passa attraverso il codice del framework

Vantaggi

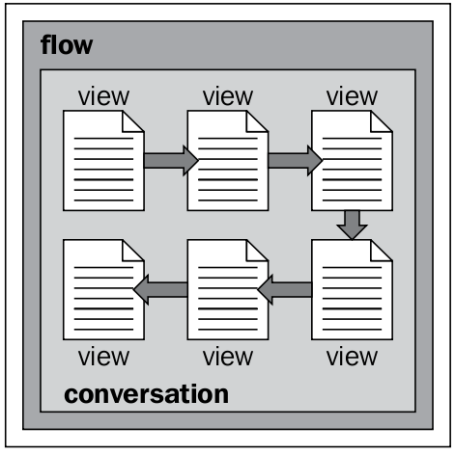
- semplificazione dello sviluppo → riduzione tempi di progetto
- disegno architetturale solido
- standard nell'implementazione dei servizi applicativi, le cui interfacce sono ben definite

Spring Web Flow



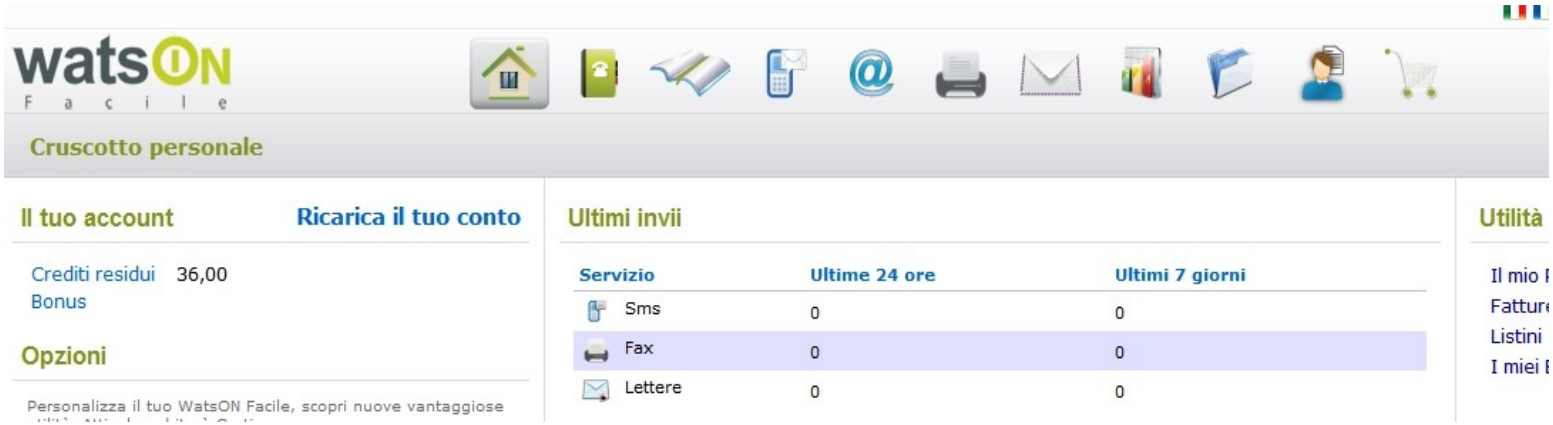
- risolve il problema della definizione e dell'esecuzione di conversazioni di un determinato livello di complessità all'interno di una web application

- funge da *controller* che orchestra le interazioni con i componenti di business, prepara i dati del model, e seleziona le view appropriate per la loro visualizzazione.



- basato sul concetto di **flow**, particolarizzazione di una FSM (*Finite State Machine*), consistente di un numero di stati (**view-state**) che definiscono le attività da eseguire durante il processing della navigazione e le relative **transitions** tra più stati.

Applicazione Target: WatsON Facile



- Fornisce la possibilità di comunicare in maniera immediata sfruttando canali di comunicazione diversi mediante un unico strumento: una RIA dall'elevata accessibilità ed immediatezza nell'utilizzo

- Permette l'invio di sms, l'invio e la ricezione di fax sulla propria casella di posta elettronica (servizio mail2fax) e l'invio di lettere di posta ordinaria dal proprio computer, oltre a gestire una rubrica contatti e un calendario eventi con reminder via email ed sms

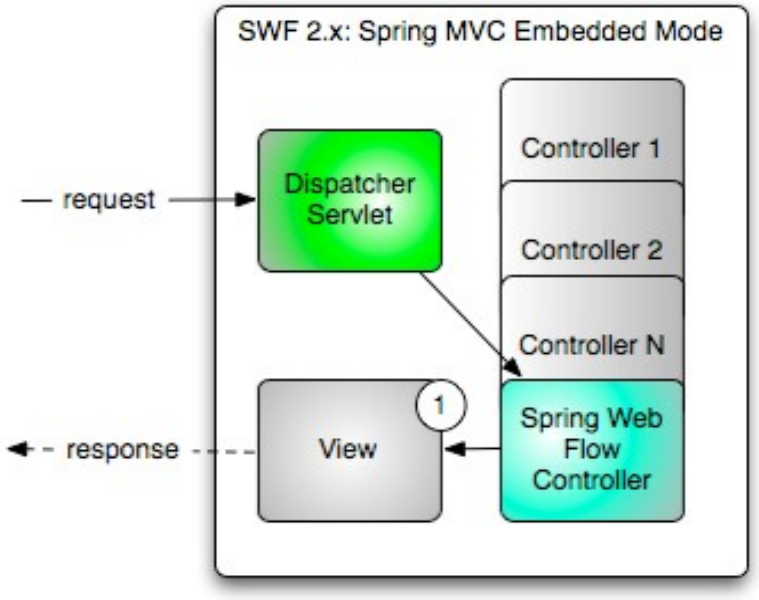
Applicazione Target: WatsON Facile

Framework utilizzati:

- ***JSF/RichFaces*** per le pagine web dinamiche
- ***ORM Hibernate*** per l'accesso e la gestione dei dati persistenti
- ***Spring Web Flow*** per la logica di controllo e di navigazione, moduli ***JAVA*** per le funzioni di business

Principio di funzionamento

Ogni view corrisponde a uno stato dell'applicazione. Le transizioni inter-view e intra-view sono mappate dal **flow descriptor**, un file xml che inoltra la richiesta all'**SWF Controller** che ha il compito di invocare i metodi di business richiesti e restituirne l'output alla view



Analisi dinamica della RIA

Obiettivo principale: completamento di un set di tasks individuati come fondamentali per una valutazione della qualità del software, in particolare, dal punto di vista della manutenibilità

Metodi: Analisi dinamica dell'esecuzione dell'applicazione

Strumenti: *DynaRIA*, tool sviluppato dal DIS dell'Università degli Studi di Napoli Federico II

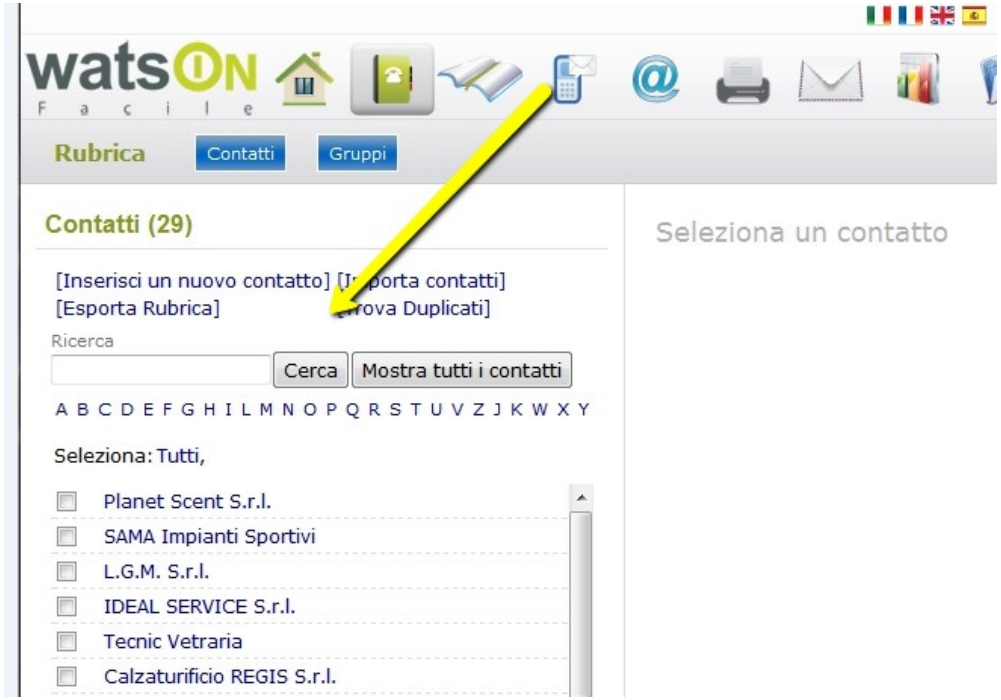
	Task Description
T1	Analisi tracciabilità codice del server → codice generato (mapping)
T2	Analisi tracciabilità codice generato → codice del server (manutenibilità)
T3	Profilazione
T4	Accoppiamento Dinamico

Task T1: Mapping codice del server → codice generato

Obiettivo: analizzare le corrispondenze tra codice del server e codice client generato dal server.

- Le pagine XHTML integrano codice HTML puro con tags richFaces tradotti runtime con il codice HTML/JavaScript corrispondente al componente rich.

- In che maniera viene effettuato il rendering ?



Task T1: Mapping codice del server → codice generato

Codice del Server →

```
<!-- PANNELLO FILTRO RICERCA -->
<a4j:outputPanel layout="block" style="margin:10px 0 20px 0;">
    <h:panelGrid columns="1" cellpadding="0" cellspacing="0">

        <h:panelGroup>
            <h:outputText value="#{msg['title.filtro_di_ricerca']}" styleClass="lightText"/><br/>
            <h:inputText id="searchVal" value="#{ps.stringaRicerca}" size="20">
                <rich:toolTip followMouse="true" direction="top-right"
                    showDelay="500">
                    <span style="white-space: nowrap">
                        <h:outputText value="#{msg['message.cerca']}" />
                    </span>
                </rich:toolTip>
            </h:
            <a4j:commandButton value="#{msg['button.cerca']}" action="cerca" id="cercaButton"
                forceId="true" reRender="sxPanel, mainPanel" />
            <a4j:commandButton value="#{msg['link.visualizza_tutti']}"
                action="visualizza_tutti" reRender="sxPanel" />
        </h:panelGroup>
```

```
<!-- PANNELLO FILTRO RICERCA -->
<div id="selContattiForm:j_id504" style="margin: 5px 0 15px 0;">
    <table cellpadding="0" cellspacing="0">
        <tbody>
        <tr>
            <td>
                <span class="lightText">Ricerca</span>
                <br />
                <span class="rich-tool-tip " id="selContattiForm:j_id509" style="z-index:99; ">
                    <span id="selContattiForm:j_id509content">
                        <span style="white-space: nowrap">
                            Inserisci il testo da ricercare nel nome, cognome oppure e-mail
                        </span>
                    </span>
                </span>
                <span id="selContattiForm:j_id509script" style="display:none">
                    <script type="text/javascript" id ="scriptselContattiForm:j_id509">
                        new ToolTip("selContattiForm:j_id509","selContattiForm:searchVal",
                            {'showEvent':'mouseover','direction':'top\x2Dright','delay':500,'followMouse':true} );
                    </script>
                </span>
                </span>
                <input id="selContattiForm:searchVal" type="text" name="selContattiForm:searchVal" size="20" />
                <input id="selContattiForm:cercaButton" name="selContattiForm:cercaButton"
                    onclick="A4J.AJAX.Submit('selContattiForm',event,
                        {'similarityGroupingId':'selContattiForm:cercaButton','parameters':
                            {'selContattiForm:cercaButton':'selContattiForm:cercaButton'} } );return false;"
                    value="Cerca" type="button" />
                <input id="selContattiForm:j_id513" name="selContattiForm:j_id513"
                onclick="A4J.AJAX.Submit('selContattiForm',event,
                        {'similarityGroupingId':'selContattiForm:j_id513','parameters':
                        {'selContattiForm:j_id513':'selContattiForm:j_id513'} } );return false;"
                        value="Mostra tutti i contatti" type="button" />
            </td>
        </tr>
```

← Codice Generato

Task T1: Mapping codice del server → codice generato

Analisi

- Tutti gli elementi rich sono stati tradotti nei relativi elementi HTML e JavaScript (toolTip, button di submit)
- Comprensione del codice generato:
  - naming non del tutto intuitivo
  - Introduzione di funzioni non note allo sviluppatore

Conseguenze

- scarso controllo del codice generato
- nessun tipo di operazione sul codice in fase di modifica se non la variazione del tipo di componenti o dei parametri del componente utlizzato

Possibile soluzione: Ridurre l'uso di componenti del framework, limitandosi ad impiegarne componenti per funzioni di core business (es. submit) ed elementi HTML/JavaScript per funzioni non strettamente correlate ad esso (es. toolTip)

Task T2: Analisi tracciabilità codice generato → codice del server

Obiettivo: in caso di difetto, comprendere fino a che livello è possibile risalire alla causa che ha provocato il malfunzionamento, dal punto di vista del codice applicativo

Due tipologie di difetti iniettati:

- **primo tipo**: alterazione del comportamento lato client, mediante l'inserimento di un alert JavaScript di una variabile mai definita

- **secondo tipo**: alterazione della business logic provocando un crash mediante l'inserimento di una non-caught exception

Task T2: Analisi tracciabilità codice generato → codice del server

- ***Primo tipo:***

```
</rich:panel>

<rich:panel id="mainPanel" styleClass="main">

    <script type="text/javascript">alert(neverDefined);</script>

    <rich:panel id="defPanel" rendered="#{((empty selectedContatto.id) and (empty viewScope.selezionati
        <div style="font-size: 150%; color: #ababab;">
            <h:outputText value="#{msg['text.seleziona_contatto']}" />
        </div>
    </rich:panel>
```

L'eccezione è rilevata da DynaRIA, ma non si concretizza in un malfunzionamento

Error messages

Line	Type	Message
9	EXCEPTION	neverDefined is not defined

```
        } else {
            window.eval(newscrip
        }
      } catch (e) {
        LOG.error("ERROR Evalu
      }
    }
    newscripts = null;
    if (isLast) {
```

Task T2: Analisi tracciabilità codice generato → codice del server

- ***Secondo tipo (codice):***

```
<transition on="cerca">
    <set name="ps.selLettera" value="null" />
    <set name="selectedContatto" value="rubricaViewService.nuovoContatto()" />

    <set name="ps.gruppoSel" value="'-'" />
    <set name="ps.page" value="0" />
    <evaluate
        expression="rubricaViewService.trovaContattoVO(currentUser.name,
            ps.stringaRicerca, ps.gruppoSelezionatoRicerca, ps.page, ps.row_pages)"
        result="contattiList" />
    <evaluate
```

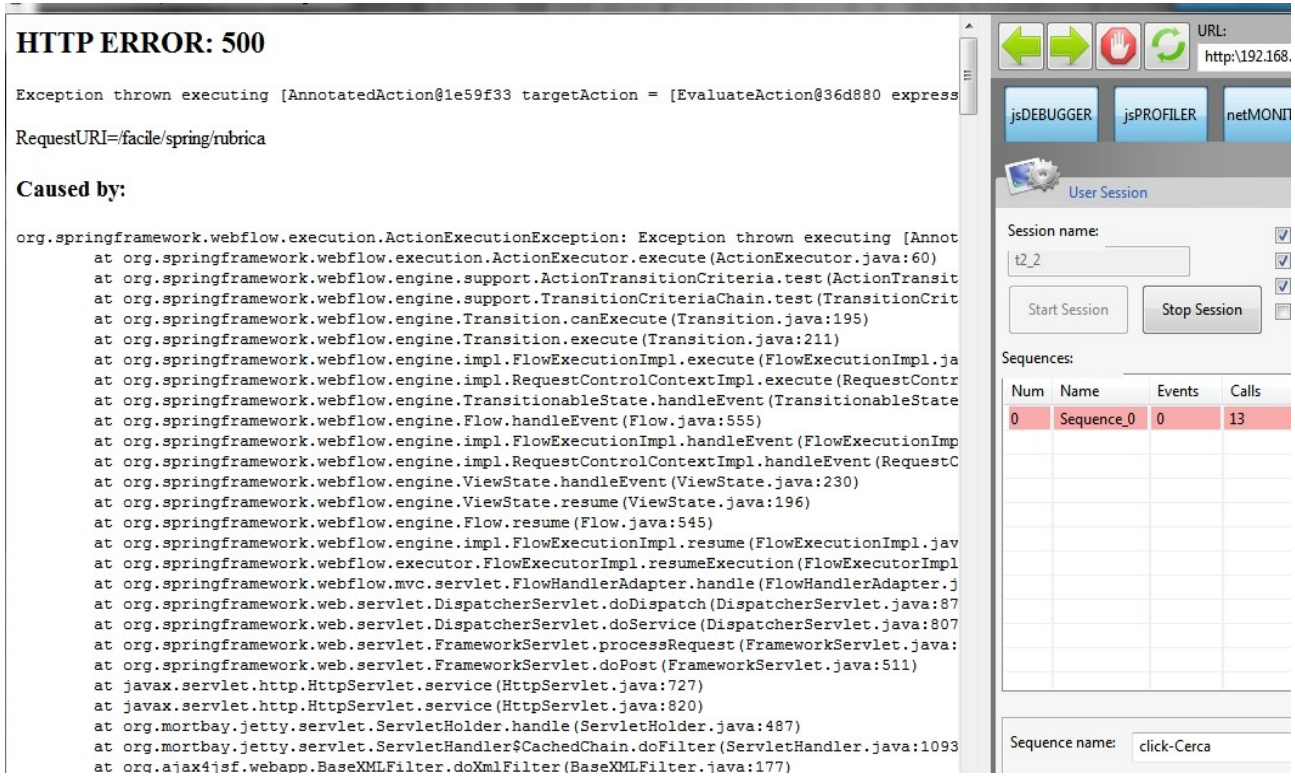
Codice Flow Descriptor (XML)

```
public List<ContattoVO> trovaContattoVO(String username,
    String nome_cognome_email, String gruppoId, int start, int size) {
try {
    throw new IllegalStateException ( );
}
    catch (NullPointerException objA){
        System.out.println ("Exception not caught.....");
    }
    return new ArrayList<ContattoVO>();
}
```

Codice Metodo Business (JAVA)

Task T2: Analisi tracciabilità codice generato → codice del server

Secondo tipo (crash):



Task T2: Analisi tracciabilità codice generato → codice del server

- **Considerazioni**

  - il framework effettua un **error masking** nel primo caso, in cui l'utente finale non percepisce l'errore grazie alla gestione effettuata a runtime dell'eccezione iniettata

  - l'individuazione della porzione di codice che ha generato l'eccezione sul client è sostanzialmente complicata, essendo il tester all'oscuro del codice JavaScript generato in maniera automatica dal framework

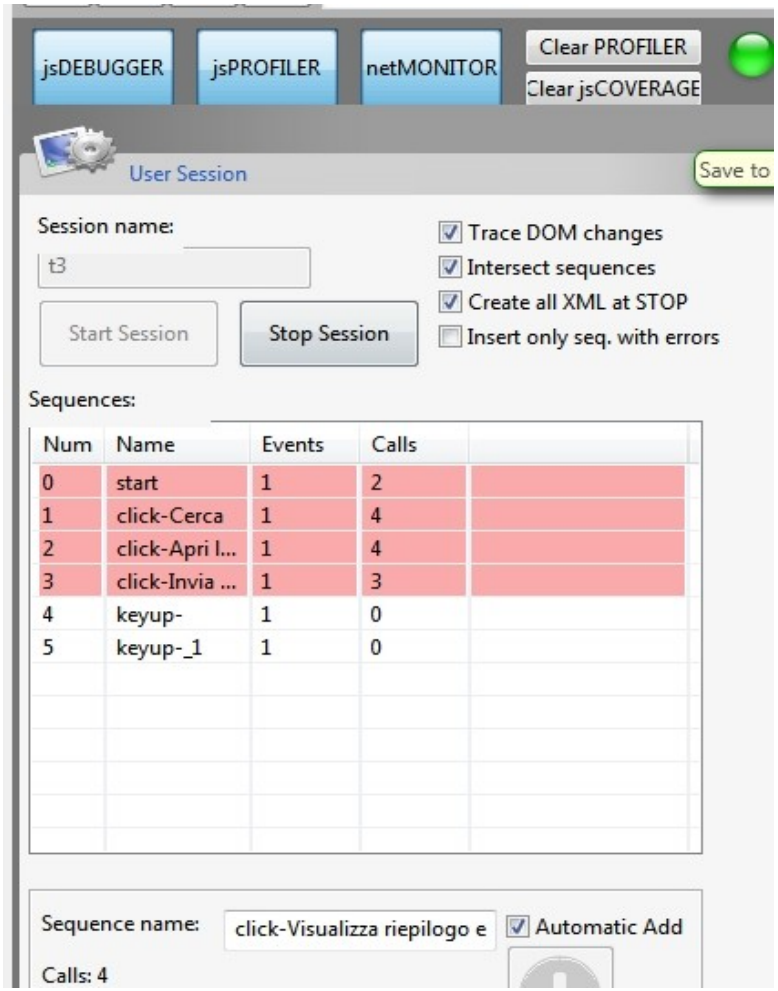
  - un difetto sul server che genera un crash sul client, è rilevato dal tester ma le informazioni che l'analisi dinamica e DynaRIA ci danno sono insufficienti per localizzarlo con precisione

Task T3: Profilazione

Obiettivo: analizzare l'efficienza del codice finale prodotto dal framework e le eventuali possibili vie di ottimizzazione

Caso d'uso: ricerca di un contatto in rubrica, visualizzazione della relativa scheda contatto, modifica di un dato, invio di sms

Metriche:
- LOC JS Generate/Eseguite
- Funzioni JS Generate/Eseguite



Task T3: Profilazione

Module	JS Found	JS Executed	Total LOC	Executed LOC	JS %	Exec %
prototypeScript	541	76	843	293	14,00%	34,75%
swfObject.js	40	8	224	49	20,00%	21,87%
tree-item.js	27	1	77	3	3,70%	3,90%
json-dom.js	10	2	17	7	20,00%	41,17%
jquery.js	340	59	1364	403	17,35%	29,54%
jquery.utils.js	5	1	15	5	20,00%	33,33%
rubrica?e=e2s1	6	2	8	4	33,33%	50,00%
utils.js	30	2	85	22	6,66%	25,88%
modalPanelBorders.js	32	13	46	42	40,62%	91,30%
modalPanel.js	48	23	145	79	47,91%	54,48%
rubrica	4	1	4	3	25,00%	75,00%
browser_info.js	3	2	15	10	66,66%	66,66%
tooltip.js	25	3	92	48	12,00%	52,17%
skinning.js	4	3	20	6	75,00%	30,00%

Task T3: Profilazione

- **Considerazioni**

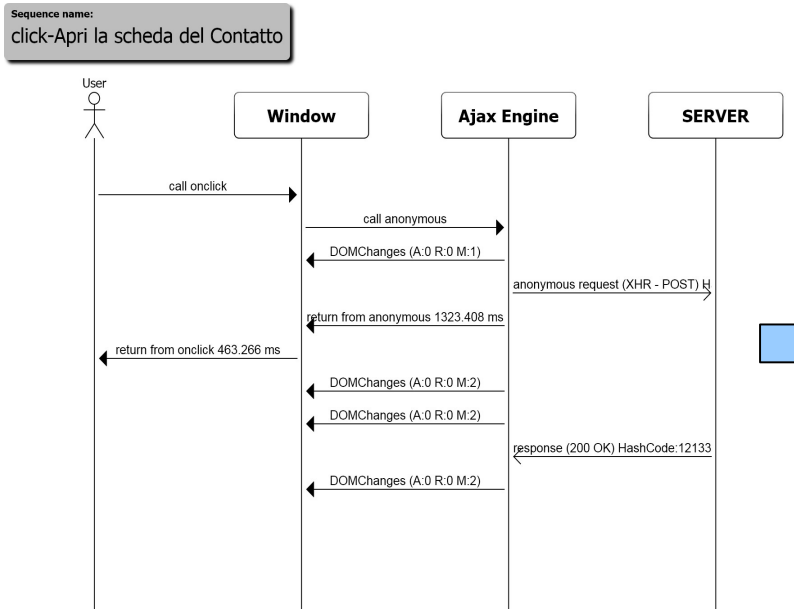
- Elaborazione dei dati che grava sul client: caricamento asincrono di porzioni di pagina

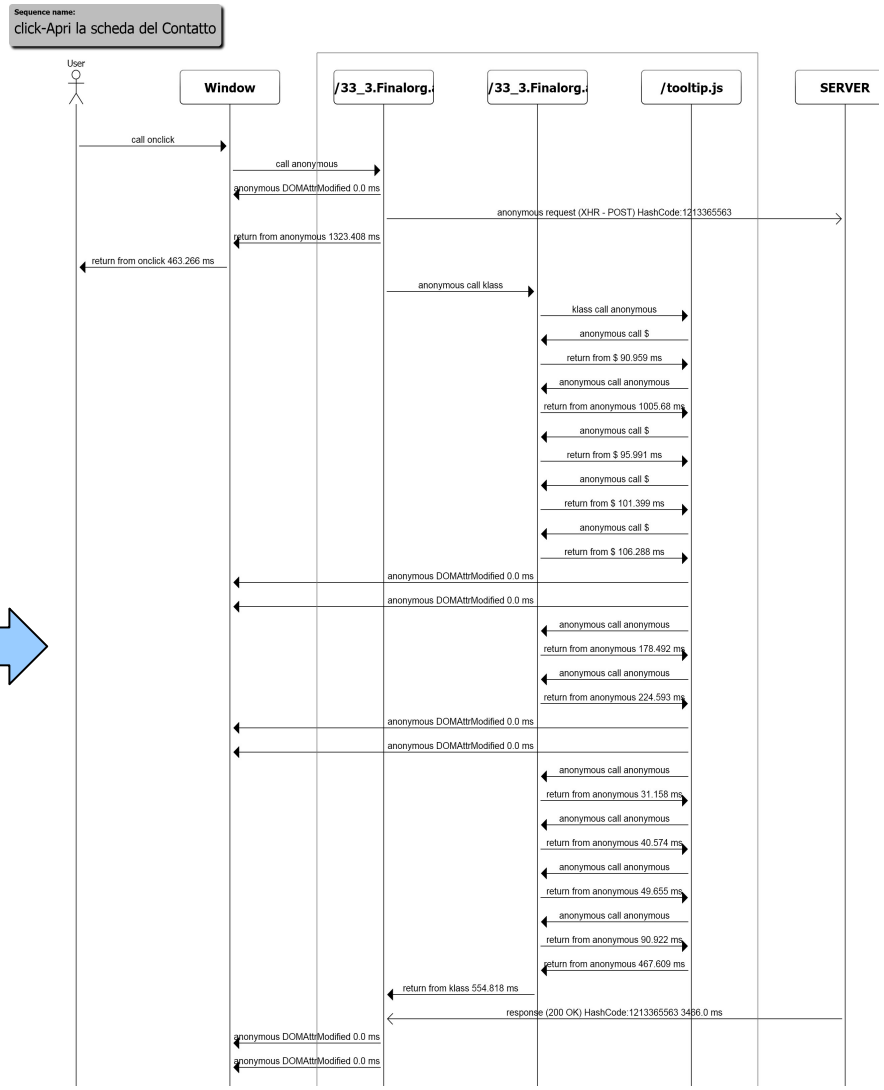
  \+ **pro**: riduzione dei cicli request-response e del traffico da e verso il server
  - **contro**: il codice generato è in quantità maggiore a quello effettivamente utilizzato

- Lo scarso controllo sul codice prodotto rende difficile perseguire obiettivi legati al miglioramento del livello di copertura del codice

Task T4: Accoppiamento dinamico

Obiettivo: comprendere in che maniera interagiscono i vari componenti dell'applicazione





Task T4: Accoppiamento dinamico

- **Considerazioni**

  \+ **pro**: richiesta soddisfatta in tempi brevi, pagina non ricaricata in toto ma solo parzialmente
  \- **contro**: numero di interazioni tra i componenti AJAX elevato e non chiaro

- In un'ottica di rimodularizzazione, lo sviluppo di componenti JS che effettuano esplicitamente le operazioni svolte a scatola chiusa risulta abbastanza complesso

Conclusioni

Vantaggi

- ***Miglioramento dei tempi di produzione del software***
  - in un contesto aziendale, la fase di produzione del codice è notevolmente ridotta, non dovendo lo sviluppatore preoccuparsi di implementare la logica interna dei componenti utilizzati

- ***Produzione "guidata" di codice***
  - il codice scritto è sostanzialmente "safe" essendo costituito da componenti testati e facilmente configurabili

Svantaggi

- ***Scarso controllo sul codice prodotto***
- ***Parziale inefficienza del codice generato***

Sviluppi futuri

Ottimizzazione del codice prodotto da un framework basate su tecniche di caching (lato client) del codice più frequentemente utilizzato