

Test di accettazione in framework MVC a confronto



Dibris

Relatore:
Prof. Maura Cerioli

Correlatore:
Prof. Davide Ancona

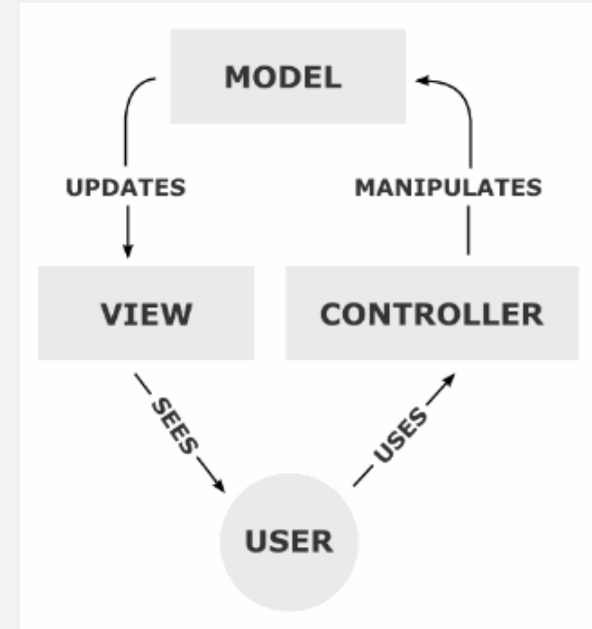
Candidato:
Mattia Barrasso
A.A. 2013/2014

Obiettivo

- "A comparative Study of Maintainability of Web Application on J2EE, .NET and Ruby on Rails", Look Fang Fang Stella, Stan Jarzabek and Bimlesh Wadhwa, 2008
 - Comparazione di RoR, Spring e .NET
 - Focus sulla manutenibilità, osservazione dello sviluppo introducendo una nuova funzionalità
- Sviluppo di tre applicazioni web
 - Utilizzando i framework MVC più diffusi e avanzati
 - Utilizzo di diverse tecnologie per lo sviluppo web
 - Comparazione delle potenzialità dei framework
- Focus sull'implementazione di test di accettazione e comparazione degli stack di strumenti utilizzati

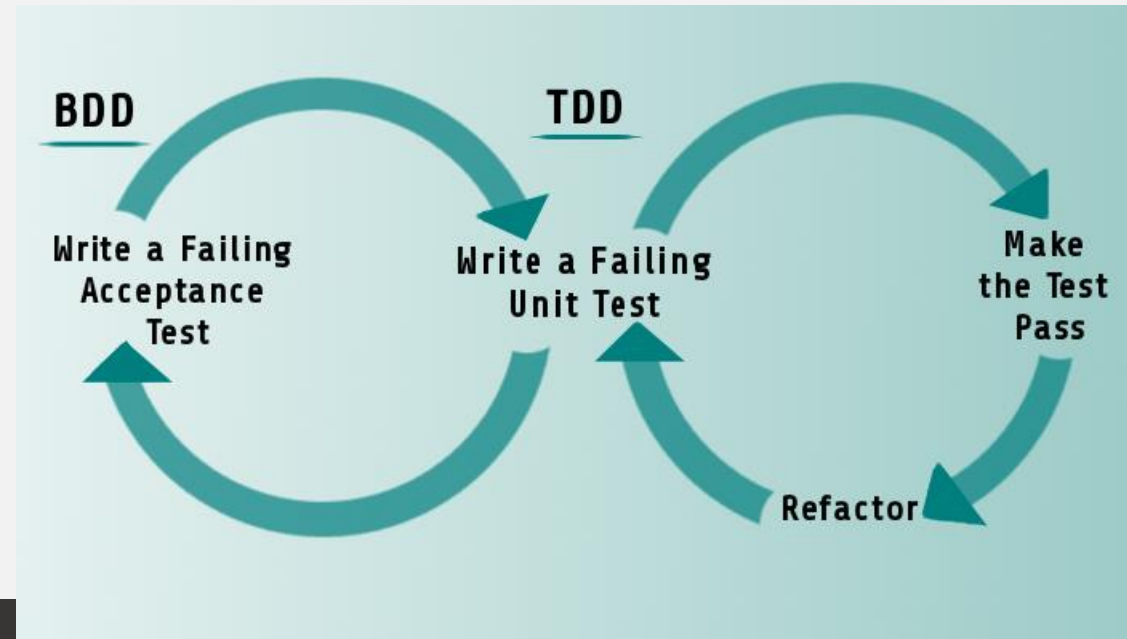
Model View Controller Pattern

- Utilizzato per la definizione di interfacce utente
- Tre componenti:
 - Il **modello** descrive il dominio e permette la persistenza delle informazioni
 - Le **viste** sono l'interfaccia dell'applicazione utilizzabili dall'utente e reagiscono in funzione dei cambiamenti sul modello
 - I **controlli** ricevono ed interpretano le richieste effettuate dall'utente
- In pratica sono utilizzate variazioni architetturali come il **Model View Presenter**, con componenti che esplicitamente interagiscono fra loro.



Behavior-driven Development

- Metodo di sviluppo le cui iterazioni producono una funzionalità funzionante e testata, descritta da un insieme di test di accettazione (o scenari)
- Utilizzo della tecnica ATDD
- Gli scenari sviluppati rappresentano la documentazione del progetto – Live Documentation



Caso di studio

- Sviluppo e test di tre applicazioni web, identiche per:
 - Funzionalità
 - Interfaccia e layout
 - Tecnologie «web» utilizzate
- Sviluppate tramite ATDD, utilizzando una libreria di test di accettazione comune
- Ogni iterazione e relativa storia coincide con l'introduzione di un numero ridotto di tecnologie



RBlog

- RoR
- Ruby



SBlog

- Spring
- Java



CSBlog

- ASP.NET MVC5
- C#

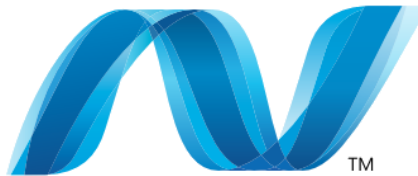
{R, S, CS}Blog MVC - Modello



PostgreSQL
Active Record



PostgreSQL
JPA + EclipseLink



Microsoft Express SQL
EF6 + ADO.NET
LINQ

{R, S, CS}Blog MVC - Viste



ERB

HTML, Sass, JavaScript, JQuery, AJAX



Thymeleaf

HTML, CSS, ...



Razor

HTML, Sass, ...

{R, S, CS}Blog MVC - Controlli



Convention over Configuration



Configuration over Configuration



Convention & Configuration

Confronto fra i framework

- ↑ Convenzioni
- ↑ Ricca libreria di Helper
- ↑ Semplicità e versatilità di Ruby



- ↑ Convenzioni e minima configurazione
- ↑ Integrazione con VS
- ↑ Strumenti molto avanzati



- ↓ Verbosità
- ↓ Eccessiva configurazione
- ↓ Template engine macchinoso

Scelta degli strumenti per l'ATDD

Prodotto

- Gratuito
- Preferibilmente Open Source

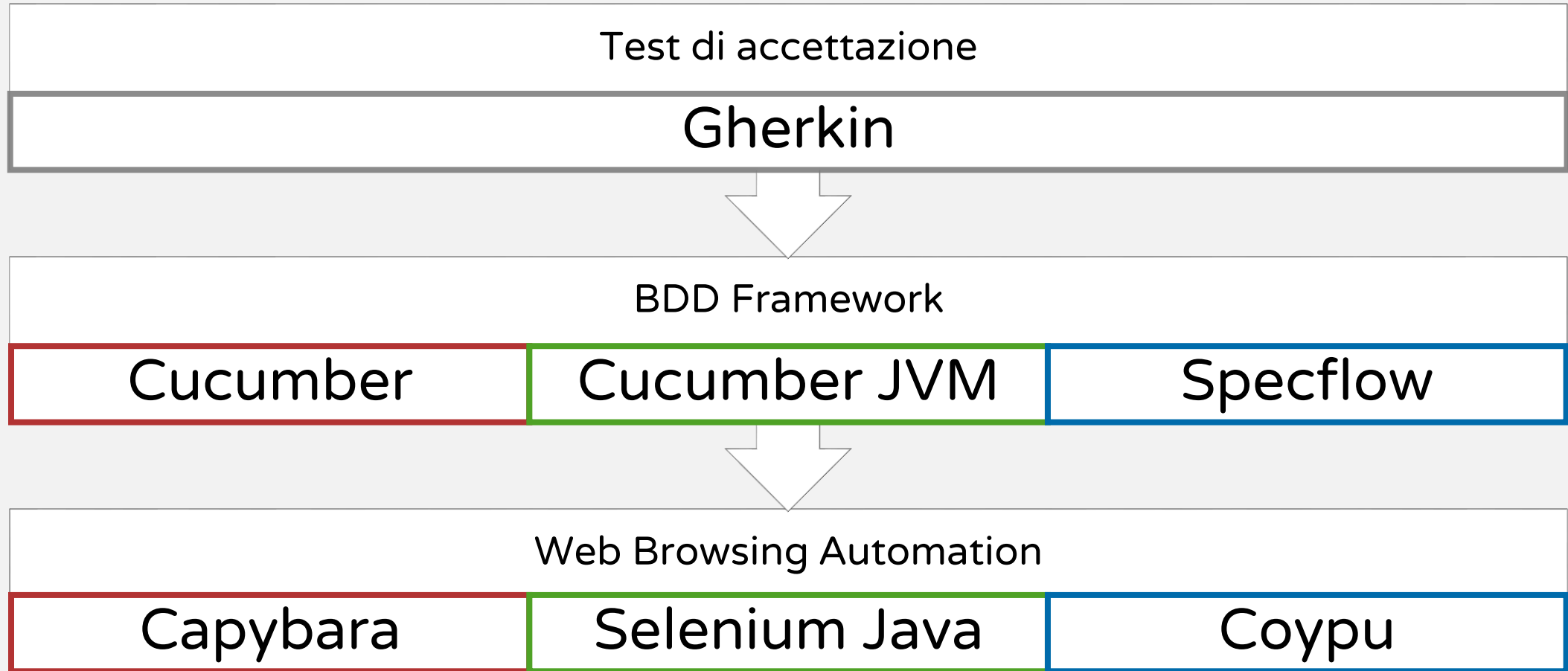
Sviluppo

- Correntemente attivo ed in evoluzione
- Al passo con i tempi e le nuove tecnologie

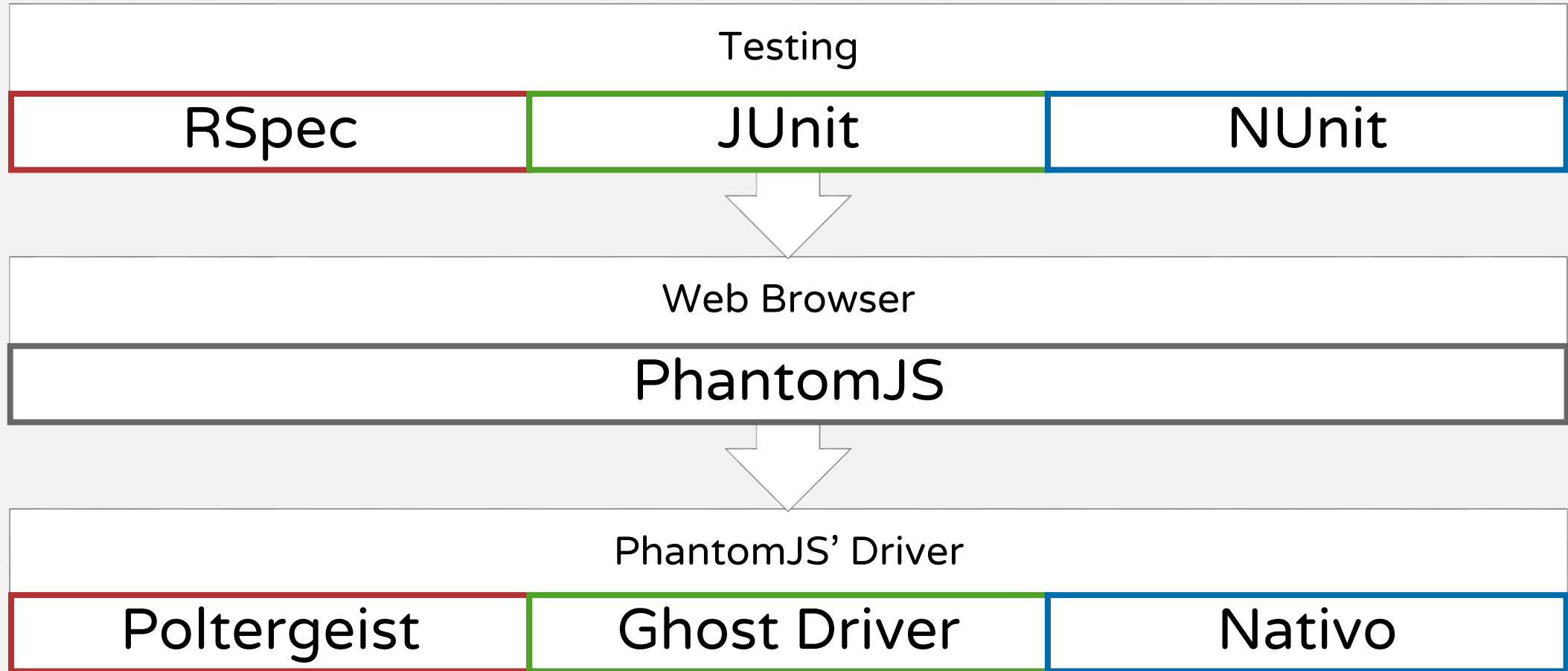
Comunità

- Esistente ed attiva
- Partecipa allo sviluppo

Testing Stack



Testing Stack (2)



Gherkin

Gherkin è un **Business Readable DSL** (Martin Fowler), utilizzato per la definizione di funzionalità e scenari e sviluppato dal team di Cucumber.

Diversi elementi sintattici:

- Scenari parametrici
- Utilizzo di tabelle per esprimere in maniera analitica molti dati
- Tag per scenari e funzionalità
- Background
- ...

Schema dello scenario: tramite l'intestazione è possibile navigare alle pagine dell'autore e dell'abstract

Dato è presente l'intestazione
E l'intestazione permette la navigazione

Allora posso navigare verso
"<nome della pagina>"

Esempi:

	nome della pagina	
	Abstract	
	Autore	

BDD Framework

- Supporto a Gherkin da parte di Cucumber, Cucumber JVM e SpecFlow:
 - Implementazione di tutte le funzionalità di Gherkin
 - Minori differenze nel meccanismo di visibilità delle implementazioni dei passi (SpecFlow è più flessibile)
 - Ruby favorisce nettamente Cucumber tramite una sintassi più immediata e leggibile
 - SpecFlow e Cucumber sono meglio integrati negli IDE rispetto a Cucumber JVM



Web Automation

Funzionalità: Hello *Blog!

Per leggere i post e visitare il blog

Come Lettore

Vorrei che *Blog permettesse la navigazione



	Open Source	Gratuito	Wrapper	Mobile	Linguaggi
Capybara	✓	✓	✗	✗	Ruby
Selenium	✗	✓	✗	✓	C#, Java, Ruby, Python, PHP, Perl, JS
Coypu	✓	✓	✓	✓ tramite Selenium	C#, Java

CRUD

Funzionalità: Gestione dei post

Come Autore

Vorrei poter inserire, modificare e
rimuovere dei post su *Blog

Per poter documentare la mia tesi

Funzionalità: Navigazione dei post

Come Lettore

Vorrei che nel blog fossero
presenti dei post

Per potermi informare



	Selettori	Page Object Pattern	Azioni	Supporto ai form
Capybara	✓ ad alto livello	✗	✓	✓
Selenium	✓	✓	✓ browser	✓
Coypu	✓ ad alto livello	✓ tramite Selenium	✓ browser	✓

Gestione del CSS

Funzionalità: Introducendo il (S)CSS

Per rendere l'esperienza di navigazione gradevole
Come Lettore

Vorrei che il sito esponesse una grafica omogenea



	Attributi inline	Fogli di stile
Capybara	Valore serializzato	Tramite JQuery, valore serializzato
Selenium	Singolo valore, solo CSS2	Singolo valore, solo CSS2
Coypu	Singolo valore, tramite Selenium	Singolo valore, tramite Selenium

Asincronia

Funzionalità: Easter Egging

Come Sviluppatore

Vorrei che nel blog fosse
presente un mio logo
Per firmare il mio lavoro

Funzionalità: Ricerca fra i post

Come Lettore

Vorrei poter ricercare i post su Rblog
Per poter navigare fra i contenuti più
velocemente



JavaScript



	Attese esplicite	Attese Implicite	Semplicità d'uso
Capybara	✗	✓	↑
Selenium	✓	✓	↓ con le attese esplicite ↑ con le attese implicite
Coypu	Tramite Selenium	✓	↑↑

Conclusione – Lo stack ideale

