

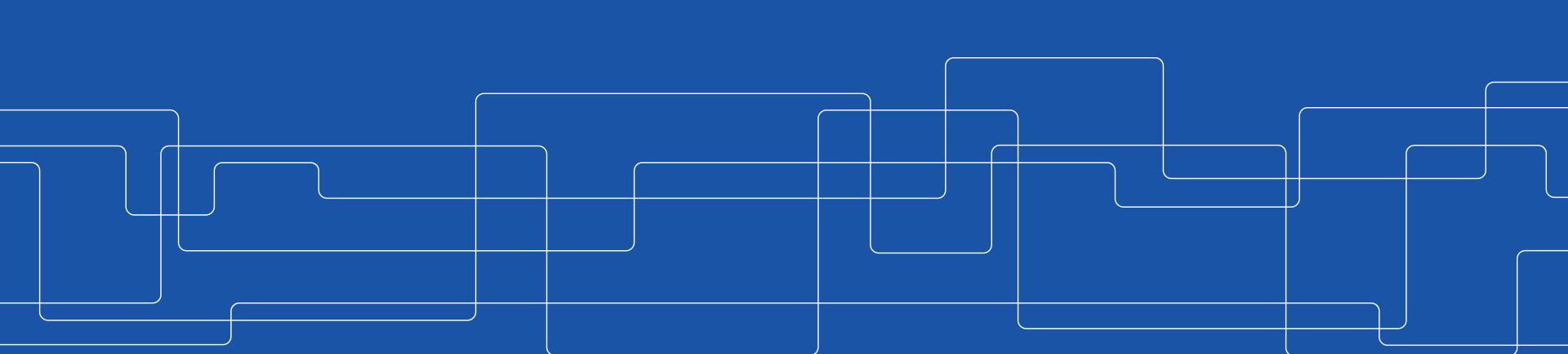


Self-healing Middleware Support for Python/Django Web Applications

Presenter: Yi-Pei Tu

Examiner: Martin Monperrus

Supervisor: Long Zhang





Outline

- Introduction
- Empirical Study
- Design & Implementation
- Experiment
- Results
- Discussion
- Conclusion



Outline

- Introduction
- Empirical Study
- Design & Implementation
- Experiment
- Results
- Discussion
- Conclusion



Outline

- Introduction
- Empirical Study
- Design & Implementation
- Experiment
- Results
- Discussion
- Conclusion



Outline

- Introduction
- Empirical Study
- Design & Implementation
- Experiment
- Results
- Discussion
- Conclusion



Introduction - Background

- Web framework
 - routing
 - web services
 - web resources
 - templating
 - session management
 - ...etc



Introduction - Background

- Web framework
 - routing
 - web services
 - web resources
 - templating
 - session management





Introduction - Background



This page isn't working

[REDACTED] is currently unable to handle this request.

HTTP ERROR 500

Reload



Introduction - Background & Related Work

- Automatic software repair
 - Monperrus, M. (2018). Automatic software repair: a bibliography. *ACM Computing Surveys (CSUR)*, 51(1), 1-24.



Introduction - Background & Related Work

- Automatic software repair
 - Monperrus, M. (2018). Automatic software repair: a bibliography. *ACM Computing Surveys (CSUR)*, 51(1), 1-24.
 - Behavioral repair
 - State repair



Introduction - Background & Related Work

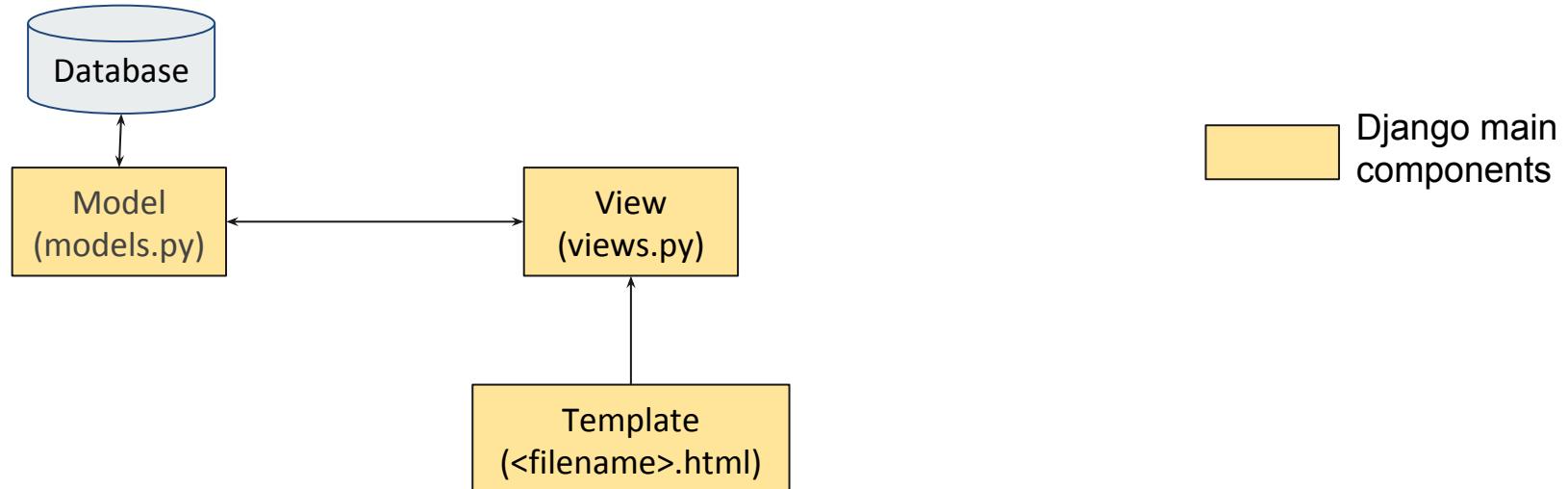
- Automatic software repair
 - Monperrus, M. (2018). Automatic software repair: a bibliography. *ACM Computing Surveys (CSUR)*, 51(1), 1-24.
 - Behavioral repair
 - State repair
 - Self-healing



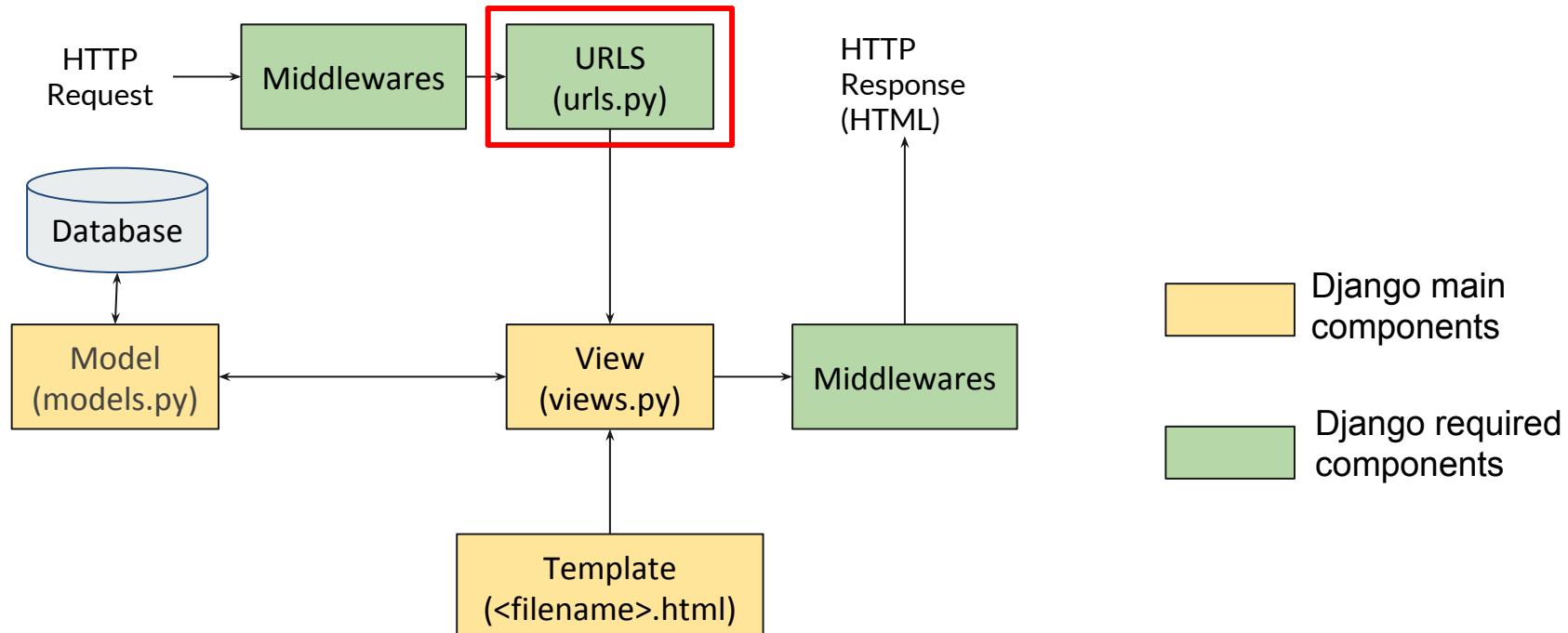
Introduction - Background & Related Work

- Self-healing
- Durieux, T., Hamadi, Y., & Monperrus, M. (2020). Fully automated HTML and JavaScript rewriting for constructing a self-healing web proxy. *Software Testing, Verification and Reliability*, 30(2), e1731.

Introduction - Django

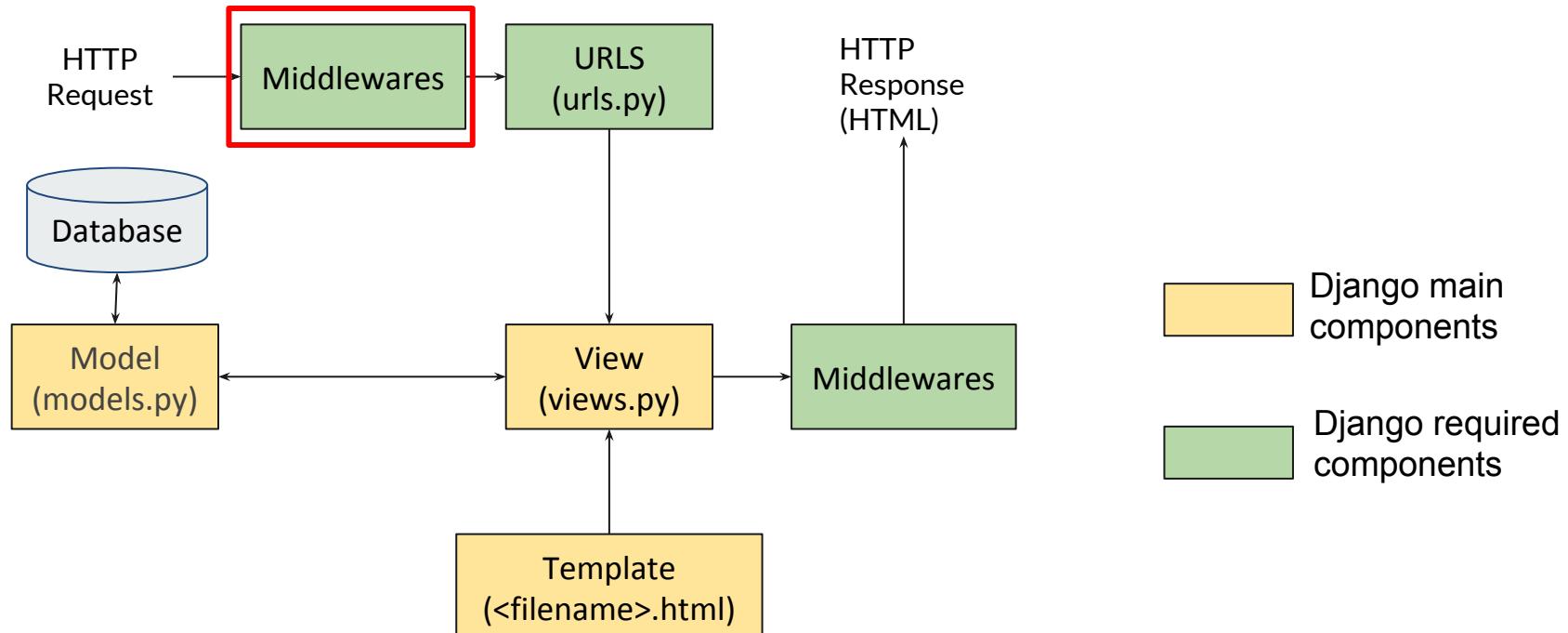


Introduction - Django



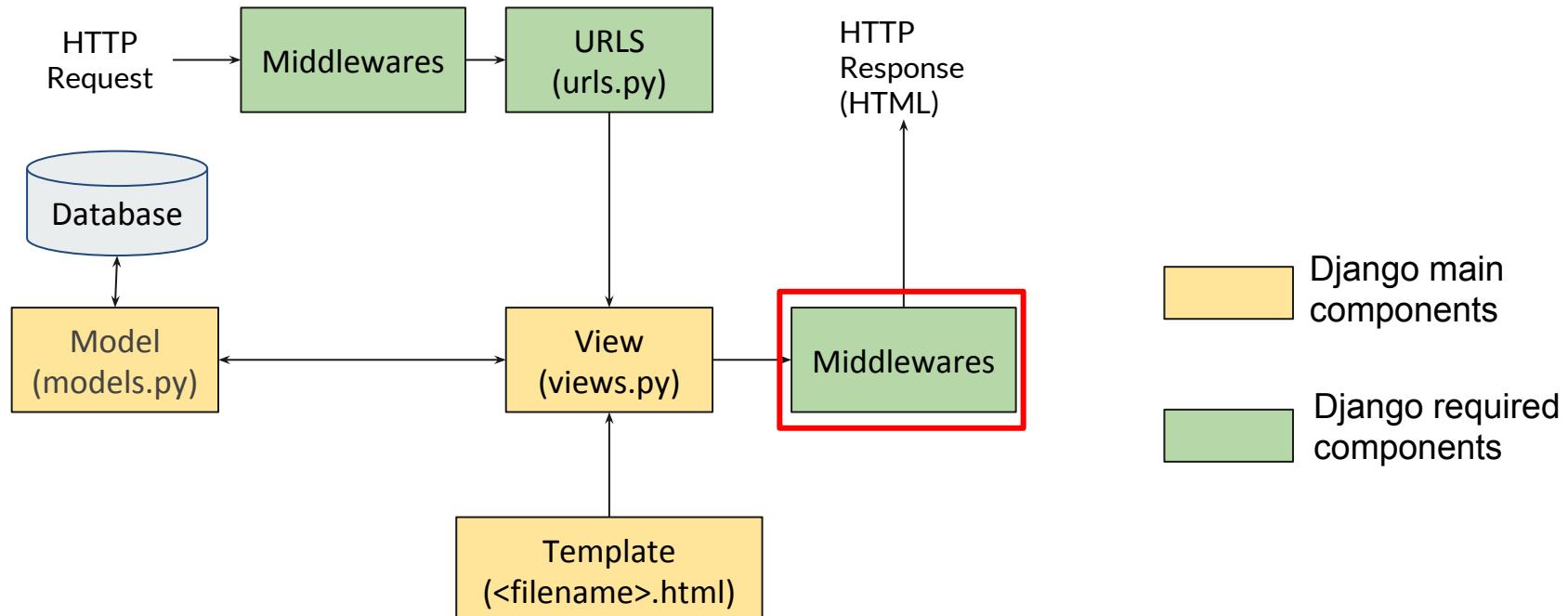


Introduction - Django





Introduction - Django





Introduction - Research Questions

1. What are the common errors in the current Django web framework?



Introduction - Research Questions

1. What are the common errors in the current Django web framework?
2. How to design and implement self-healing strategies for Django web applications?



Introduction - Research Questions

1. What are the common errors in the current Django web framework?
2. How to design and implement self-healing strategies for Django web applications?
3. How to evaluate the self-healing prototype with respect to effectiveness?



Introduction - Research Questions

1. What are the common errors in the current Django web framework?
2. How to design and implement self-healing strategies for Django web applications?
3. How to evaluate the self-healing prototype with respect to effectiveness?
4. How to evaluate the self-healing prototype with respect to overhead?



Empirical Study - Protocol

- 16 Django popular projects (374~6,297 stars)





Empirical Study - Protocol

- 16 Django popular projects (374~6,297 stars)
- 19,442 issues



◀ The no-conflict mode should be the default behaviour #12395

Open thewebdreamer opened this issue 3 days ago · 10 comments

thewebdreamer commented 3 days ago
The no-conflict mode should be the default behaviour. Why would a Bootstrap client need to implement this?

cvrebert commented 3 days ago
I believe no-conflict-is-not-the-default is the norm for jQuery plugins?

thewebdreamer commented 3 days ago
It is true that it is the norm for jQuery plugins.
Couldn't there be a clash with other jQuery plugins with the current implementation of Bootstrap though?

Labels js

Milestone No milestone

Assignee No one assigned

Notifications Subscribe

3 participants



Empirical Study - Analysis

- 5 types of Django exception
 - URL Resolver exceptions
 - Http exceptions
 - Database exceptions
 - Transaction exceptions
 - Core exceptions



Empirical Study - Results

- Top 4 covers 57.35% issues

Exception Type	Exception	Frequency	Percentage
URL Resolver Exceptions	NoReverseMatch	44/265	16.6%
Database Exceptions	OperationalError	42/265	15.85%
Core Exceptions	ValidationError	38/265	14.33%
Database Exceptions	IntegrityError	28/265	10.57%



Design & Implementation - Architecture

1. Request monitor
2. Controller
3. Strategy executor
4. Strategy verifier



Design & Implementation - Architecture

1. Request monitor
2. Controller
3. Strategy executor
4. Strategy verifier



Design & Implementation - Architecture

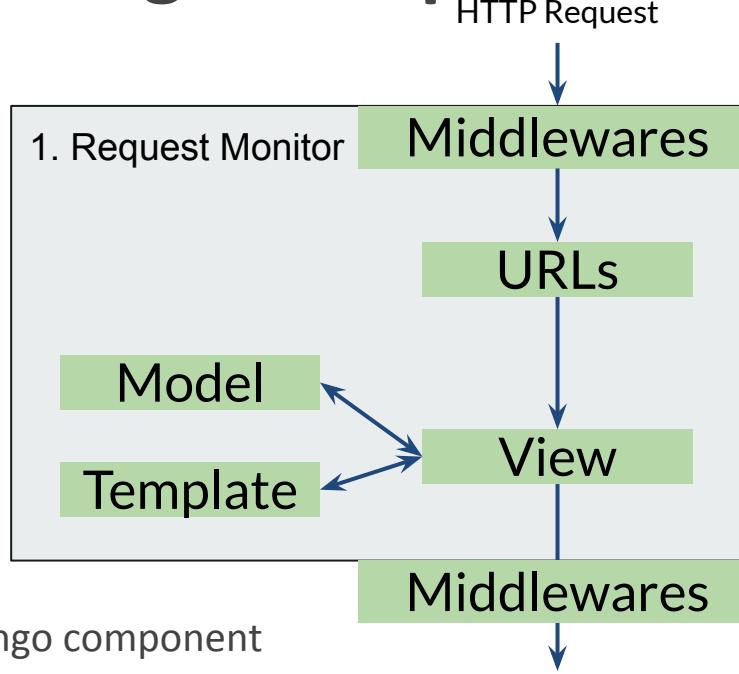
1. Request monitor
2. Controller
3. Strategy executor
4. Strategy verifier



Design & Implementation - Architecture

1. Request monitor
2. Controller
3. Strategy executor
4. Strategy verifier

Design & Implementation - Workflow

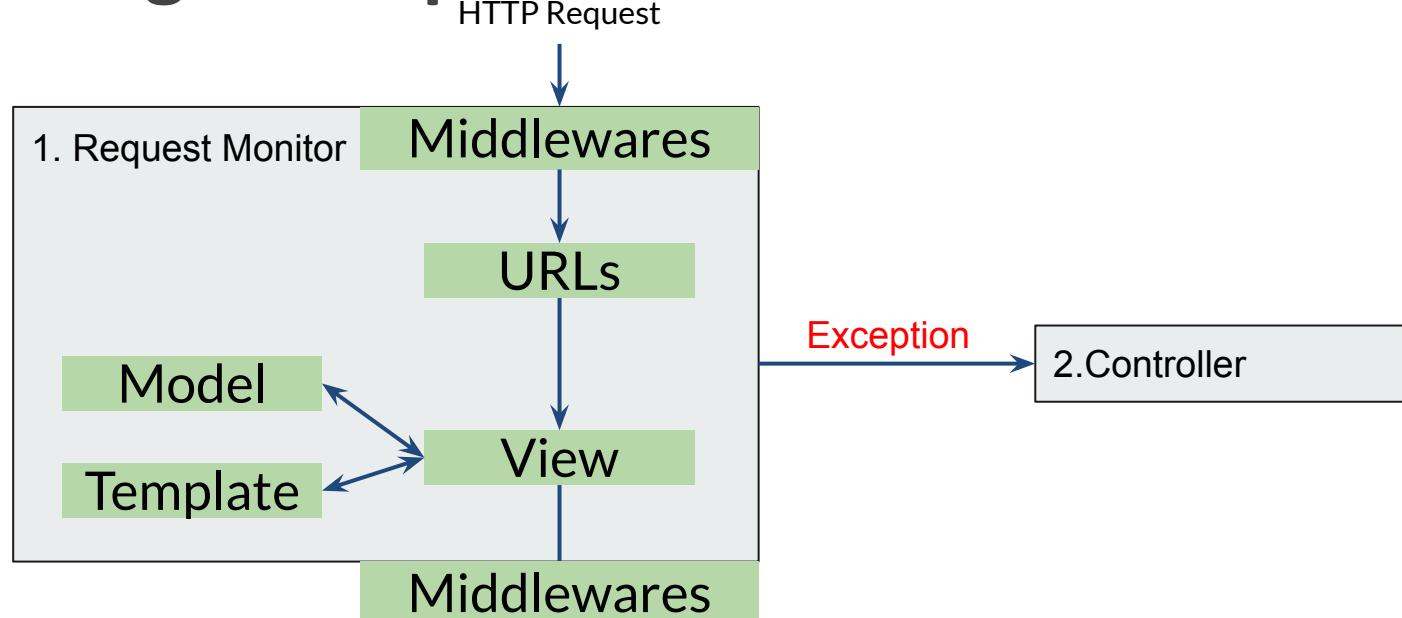


django component

self-healing middleware
component

HTTP Response

Design & Implementation - Workflow

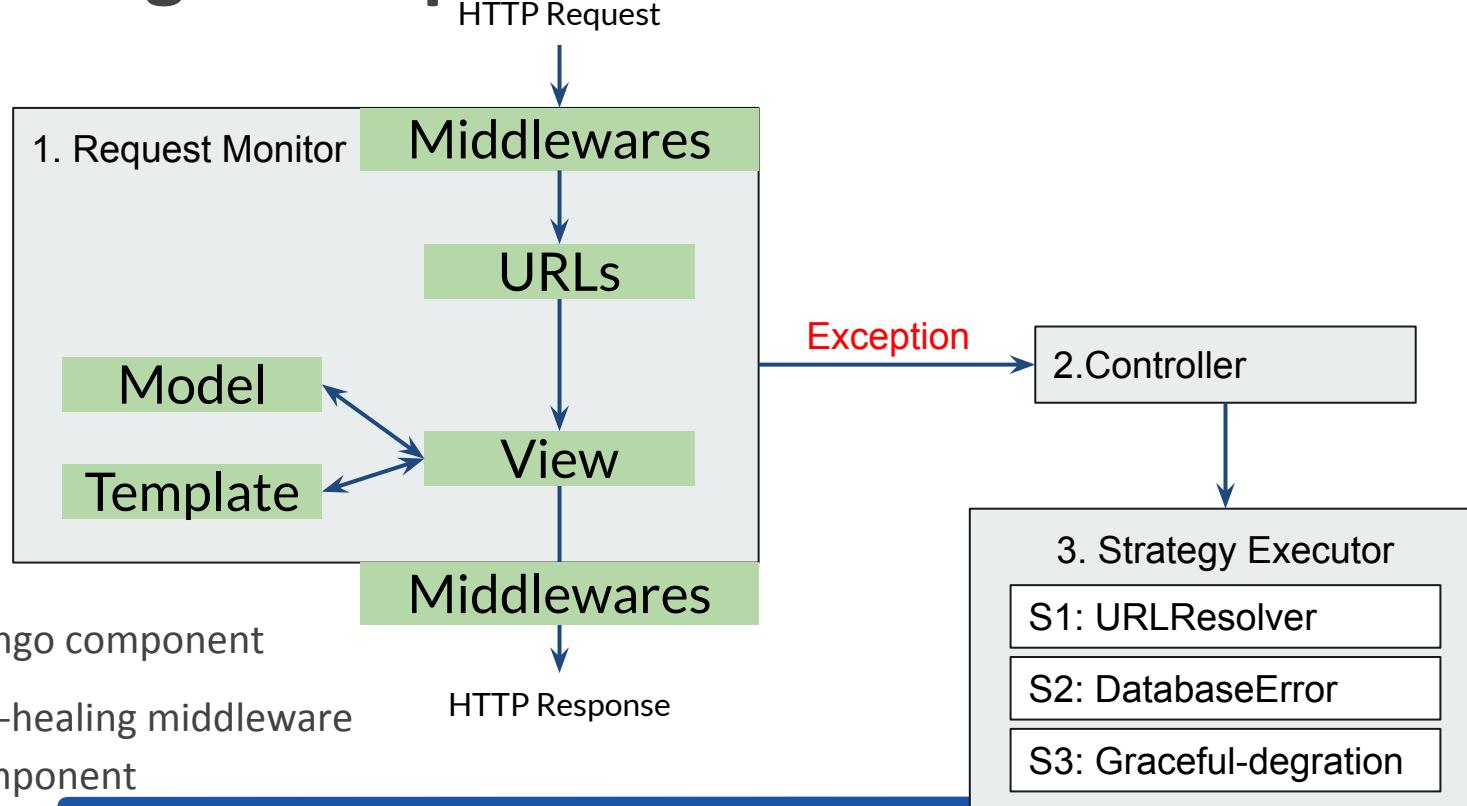


django component

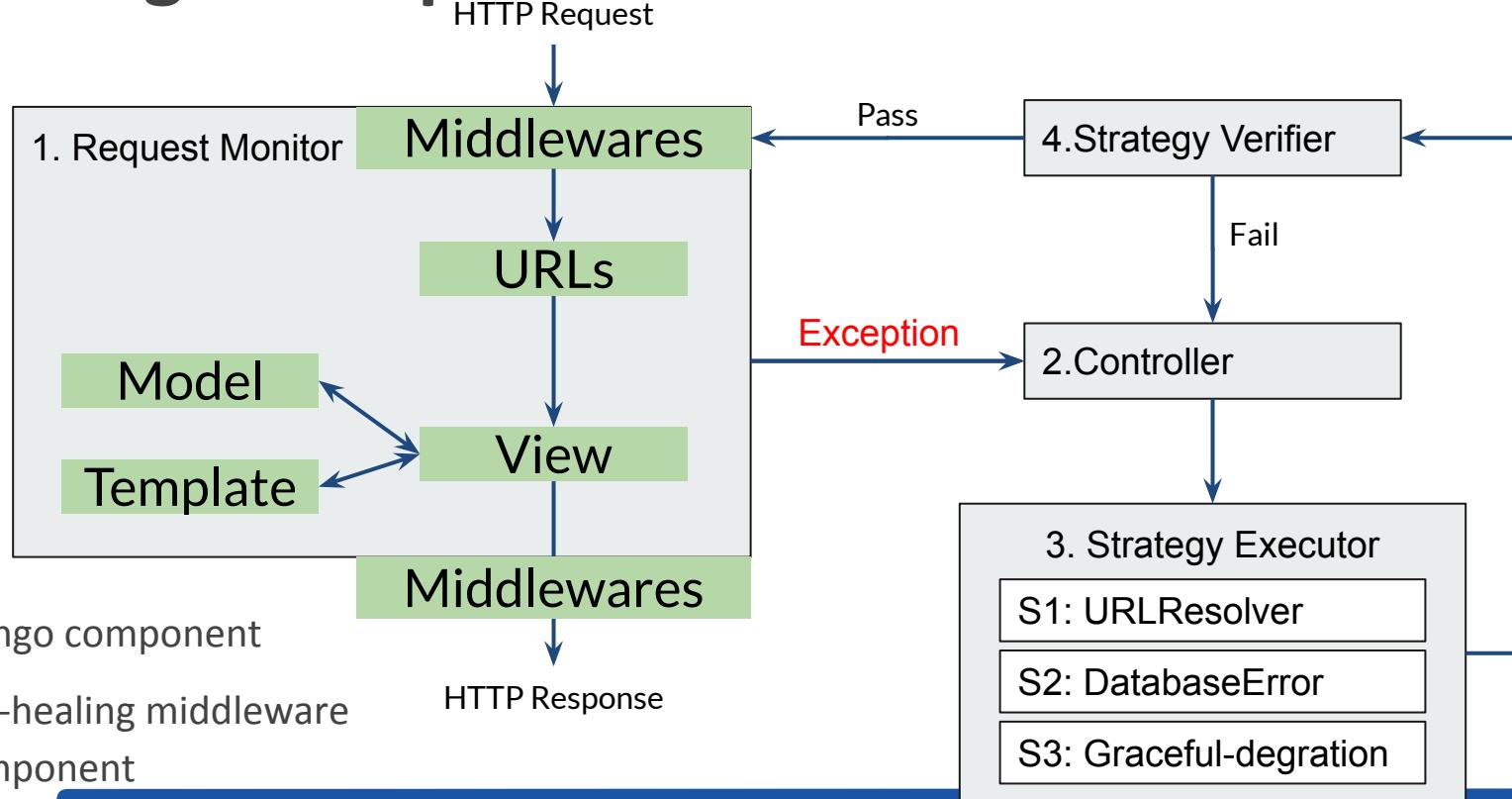
self-healing middleware
component

HTTP Response

Design & Implementation - Workflow

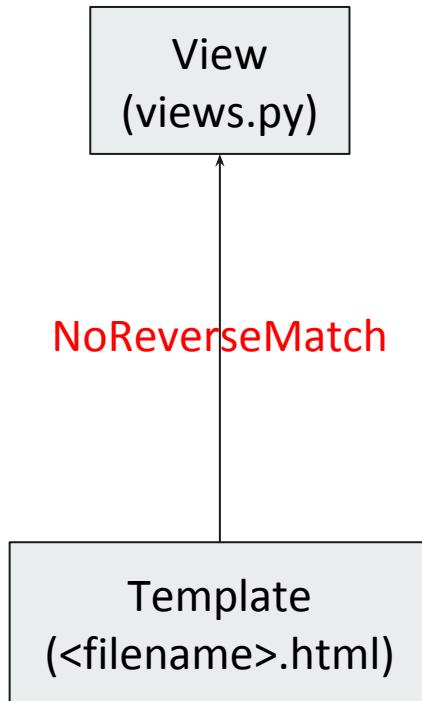


Design & Implementation - Workflow



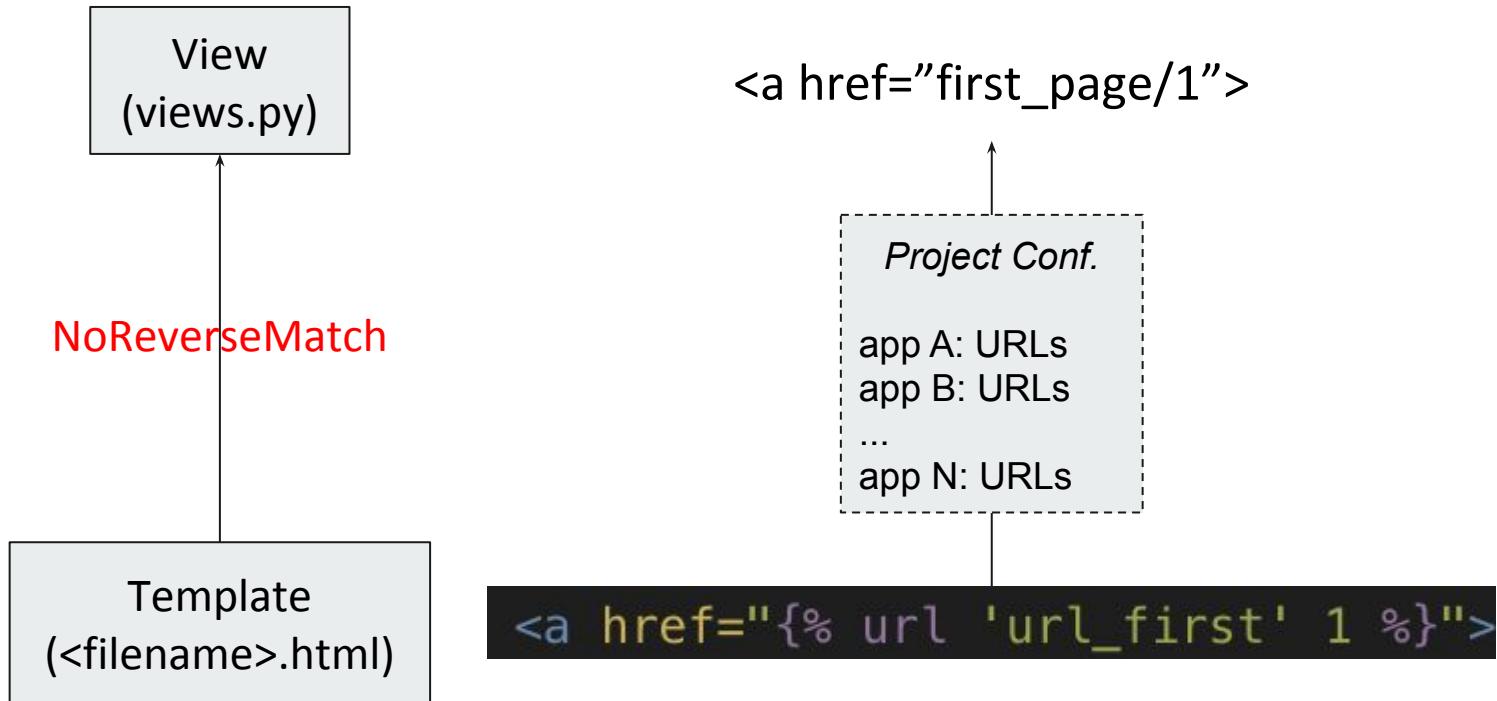


Design & Implementation - S1-URLResolver





Design & Implementation - S1-URLResolver





Design & Implementation - S1-URLResolver

Template
Modification

Template
(<filename>.html)

Configuration
Modification

Project Conf.

app A: URLs
app B: URLs
...
app N: URLs



Design & Implementation - S2-DatabaseError





Design & Implementation - S2-DatabaseError



Name *Unique key	Age *Cannot be null
Peter	12
Kim	23

Name	Age
Peter	



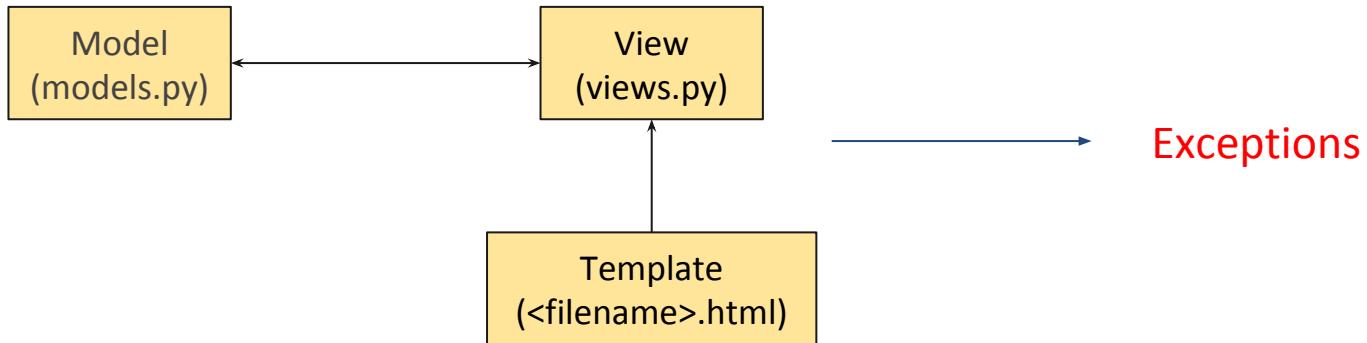
Design & Implementation - S2-DatabaseError



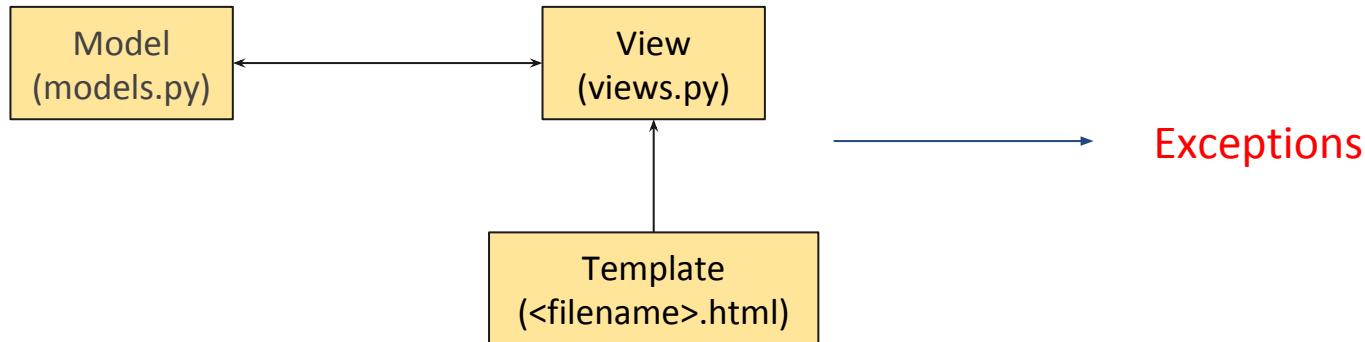
- Database migration
- Show error message page



Design & Impl. - S3-Graceful-degradation



Design & Impl. - S3-Graceful-degradation



- Show home page
- Show last visited page
- Show error message page



Experiment - Projects

- blog-zinnia (1,897 stars)

The screenshot shows the homepage of "Zinnia's Blog". At the top, there is a navigation bar with links to Sitemap, RSS Feed, Planet, Documentation / Covers, Contact, and a GitHub link. Below the header, there is a large image of several red flowers. A dark blue overlay box contains the text "Hello World!". Below the image, there is a navigation bar with arrows and a "Welcome!" message. The main content area displays a single blog post titled "Hello World!" by admin on Jan. 1, 2010 in Uncategorized. The post content is: "Welcome to the Zinnia blogging application written in Django. This is your first entry, don't hesitate to archive or remove it." A note at the bottom states: "The site will reset every two hours."

- django-oscar (3,860 stars)

The screenshot shows the "Profile" page of the Oscar Sandbox. At the top, there is a header with the text "Oscar Sandbox" and a "Basket total: £15.00" message. Below the header, there is a search bar and a "Search" button. The main content area has a "Profile" tab selected, showing the following user information:

Name	-
Email address	superuser@example.com
Date registered	12 Sep 2012, 6 p.m.

Below the table, there are "Change password" and "Edit profile" buttons. To the left of the table, there is a sidebar with links to Order History, Address Book, Email History, Product Alerts, Notifications, and Wish Lists.



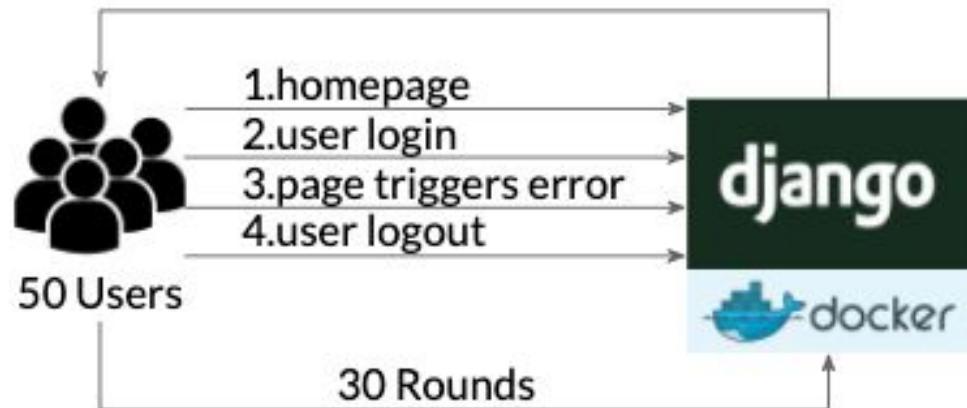
Experiment - Workload

- 50 Users
- 4 Action requests
 - home page
 - login
 - exception
 - logout
- 30 Rounds

Experiment - Workload

- Workload
 - 50 Users
 - 4 Requests
 - 30 Rounds
- 3 Experiments

- One experiment





Experiment - Metrics

Metrics	Indicator	Measured By
Effectiveness	Correctness	JMeter 
Overhead	Response time	JMeter 
	CPU Usage	cAdvisor 
	Memory Usage	cAdvisor 

Durieux, T., Hamadi, Y., & Monperrus, M. (2020). Fully automated HTML and JavaScript rewriting for constructing a self-healing web proxy. *Software Testing, Verification and Reliability*, 30(2), e1731.



Experiment - Metrics

Metrics	Indicator	Measured By
Effectiveness	Correctness	JMeter 
Overhead	Response time	JMeter 
	CPU Usage	cAdvisor 
	Memory Usage	cAdvisor 

Durieux, T., Hamadi, Y., & Monperrus, M. (2020). Fully automated HTML and JavaScript rewriting for constructing a self-healing web proxy. *Software Testing, Verification and Reliability*, 30(2), e1731.



Results - Correctness

SHM: self-healing middleware

Exception Type	Projects	Action Requests			
		Admin	Login	Exception	Logout
URL Resolver	blog-zinnia without SHM	100%	100%	0%	100%
DatabaseError	django-oscar without SHM	100%	29.65%	0%	88.34%



Results - Correctness

SHM: self-healing middleware

Exception Type	Projects	Action Requests			
		Admin	Login	Exception	Logout
URL Resolver	blog-zinnia without SHM	100%	100%	0%	100%
	blog-zinnia with SHM	100%	100%	100%	100%
DatabaseError	django-oscar without SHM	100%	29.65%	0%	88.34%
	django-oscar with SHM	100%	27.11%	11.87%	88.73%



Results - Response Time (seconds)

SHM: self-healing middleware

Exception Type	Projects	Action Requests				Total Time
		Admin	Login	Exception	Logout	
URL Resolver	blog-zinnia without SHM	1.09	5.98	9.14	3.34	19.57
DatabaseError	django-oscar without SHM	3.74	8.02	2.28	10.79	24.83



Results - Response Time (seconds)

SHM: self-healing middleware

Exception Type	Projects	Action Requests					Total Time
		Admin	Login	Exception	Logout		
URL Resolver	blog-zinnia without SHM	1.09	5.98	9.14	3.34		19.57
	blog-zinnia with SHM	2.71	10.66	28.06	6.11		47.55
DatabaseError	django-oscar without SHM	3.74	8.02	2.28	10.79		24.83
	django-oscar with SHM	8.19	18.22	4.97	21.52		52.9



Results - CPU & Memory Usage

SHM: self-healing middleware

Exception	Projects	CPU (%)	Memory (%)
URL Resolver	blog-zinnia without SHM	64.5	10.31
	blog-zinnia with SHM	49.34	11.05
DatabaseError	django-oscar without SHM	73.23	11.37



Results - CPU & Memory Usage

SHM: self-healing middleware

Exception	Projects	CPU (%)	Memory (%)
URL Resolver	blog-zinnia without SHM	64.5	10.31
	blog-zinnia with SHM	49.34	11.05
DatabaseError	django-oscar without SHM	73.23	11.37
	django-oscar with SHM	99.58	26.76



Discussion - Research Questions

- RQ1: What are the common errors in the current Django web framework? [Empirical study](#)
- RQ2: How to design and implement self-healing strategies for Django web applications? [Design & Implementation](#)
- RQ3: How to evaluate the self-healing prototype with respect to effectiveness? [Correctness](#)
- RQ4: How to evaluate the self-healing prototype with respect to overhead? [Resp. time, CPU, and memory usage](#)



Discussion - Research Questions

- RQ1: What are the common errors in the current Django web framework? [Empirical study](#)
- RQ2: How to design and implement self-healing strategies for Django web applications? [Design & Implementation](#)
- RQ3: How to evaluate the self-healing prototype with respect to effectiveness? [Correctness](#)
- **RQ4:** How to evaluate the self-healing prototype with respect to overhead? [Resp. time, CPU, and memory usage](#)



Discussion - Limitation & Future Work

- Middleware position
- Self-healing strategies
- Correctness measurement

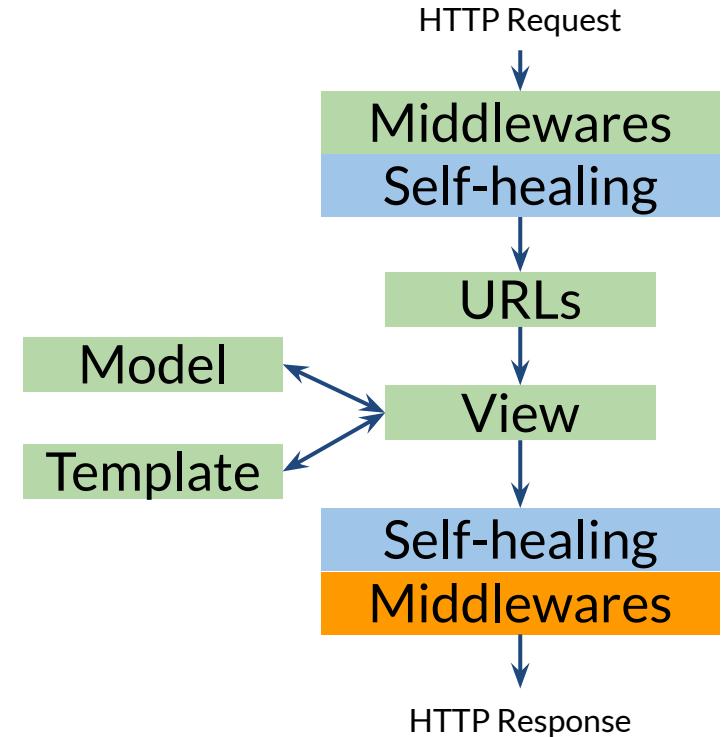
Discussion - Limitation & Future Work

- Middleware position
- Self-healing strategies
- Correctness measurement

 self-healing middleware

 django components

 cannot be caught





Conclusion

- First common Django errors
- First self-healing middleware for Django
- Open source