JavaCro

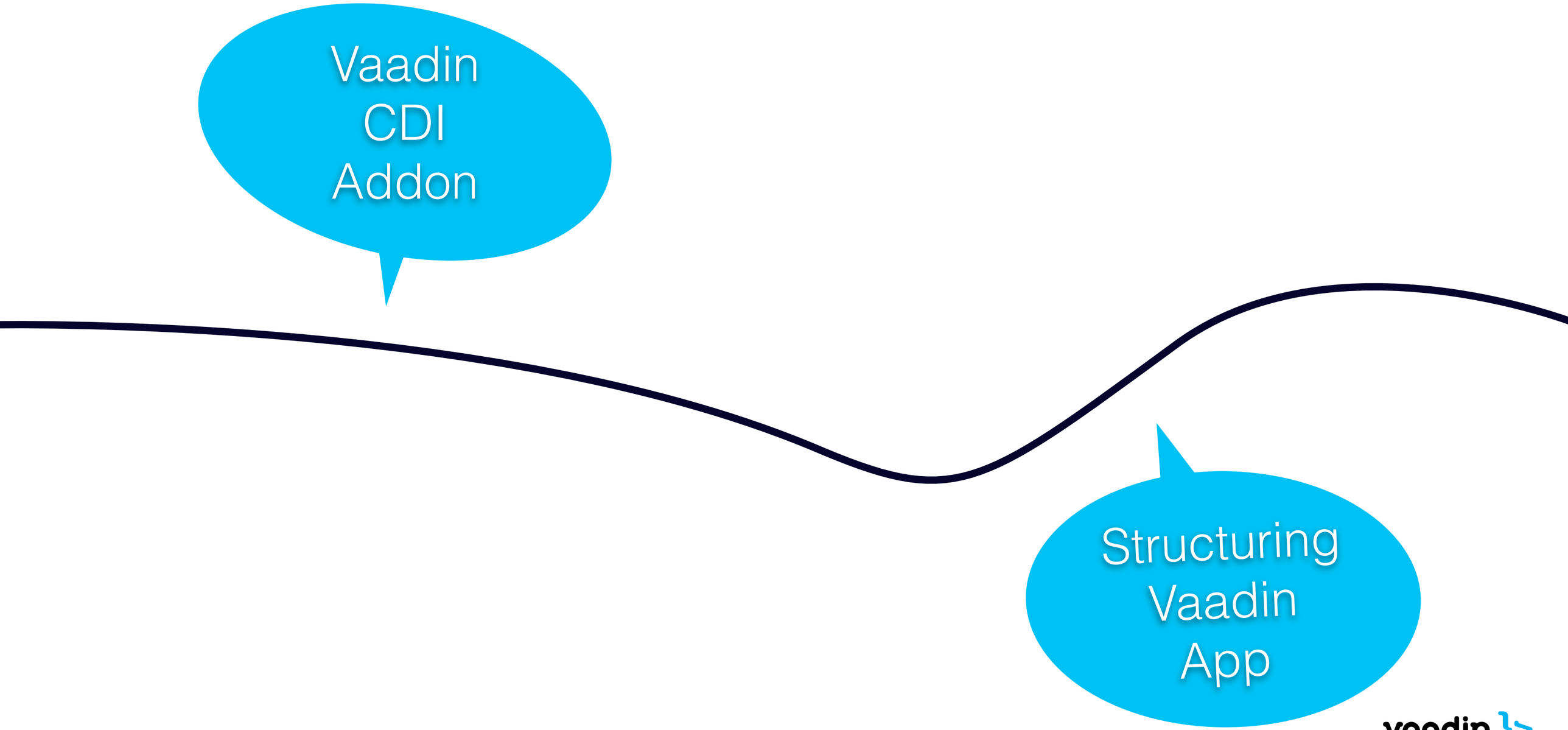
Web UI best practice integration with Java EE 7

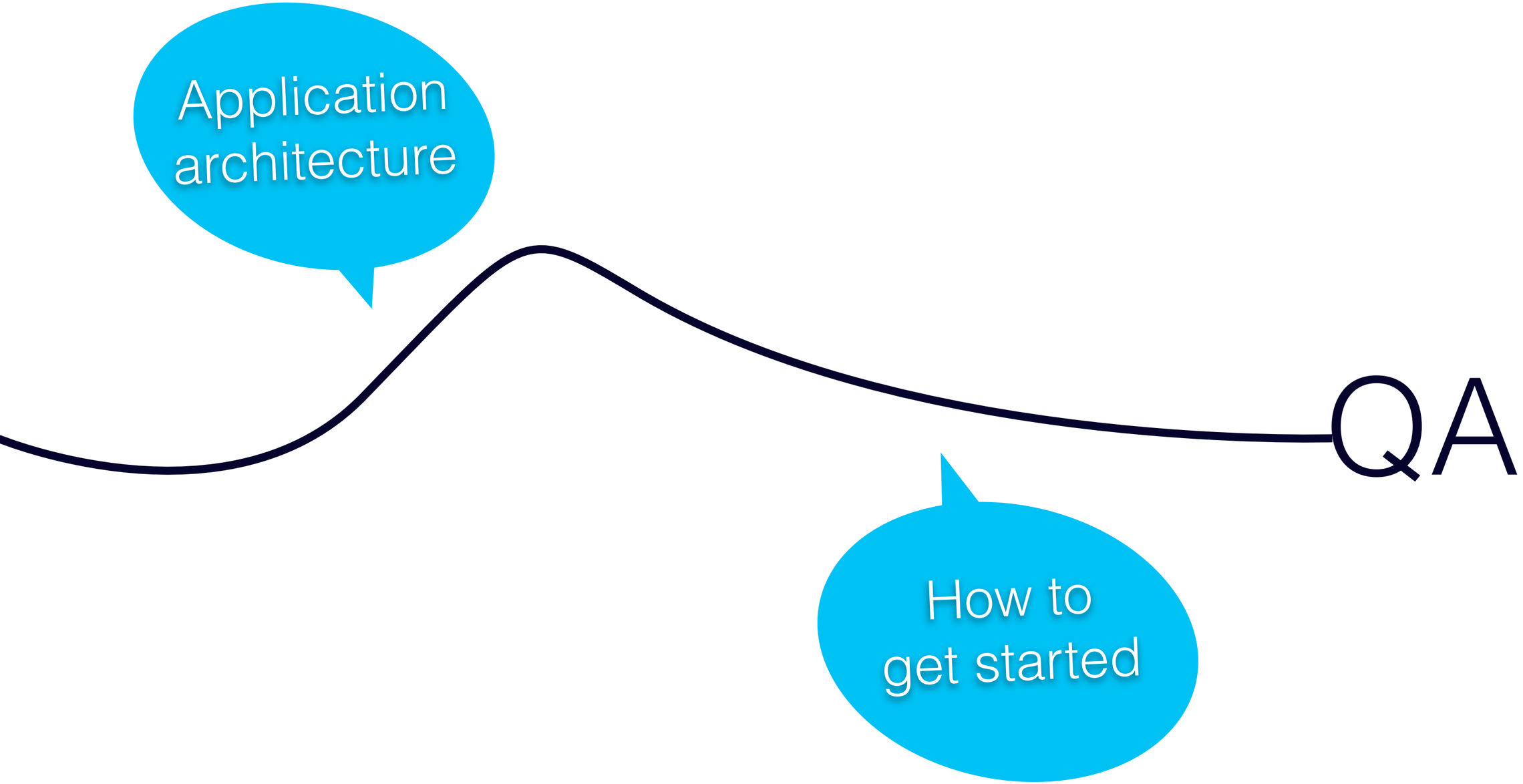
Peter Lehto
@peter_lehto
expert & trainer
vaadin

What is Vaadin?

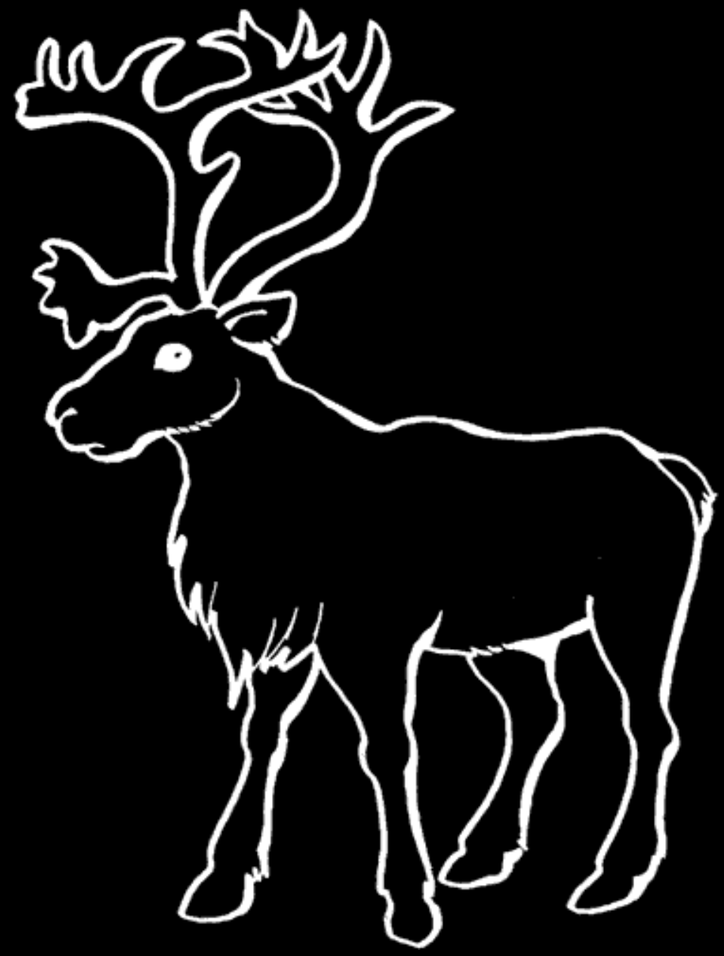
Java EE 7

vaadin


Vaadin
CDI
Addon
Structuring
Vaadin
App


Application architecture
QA
How to get started

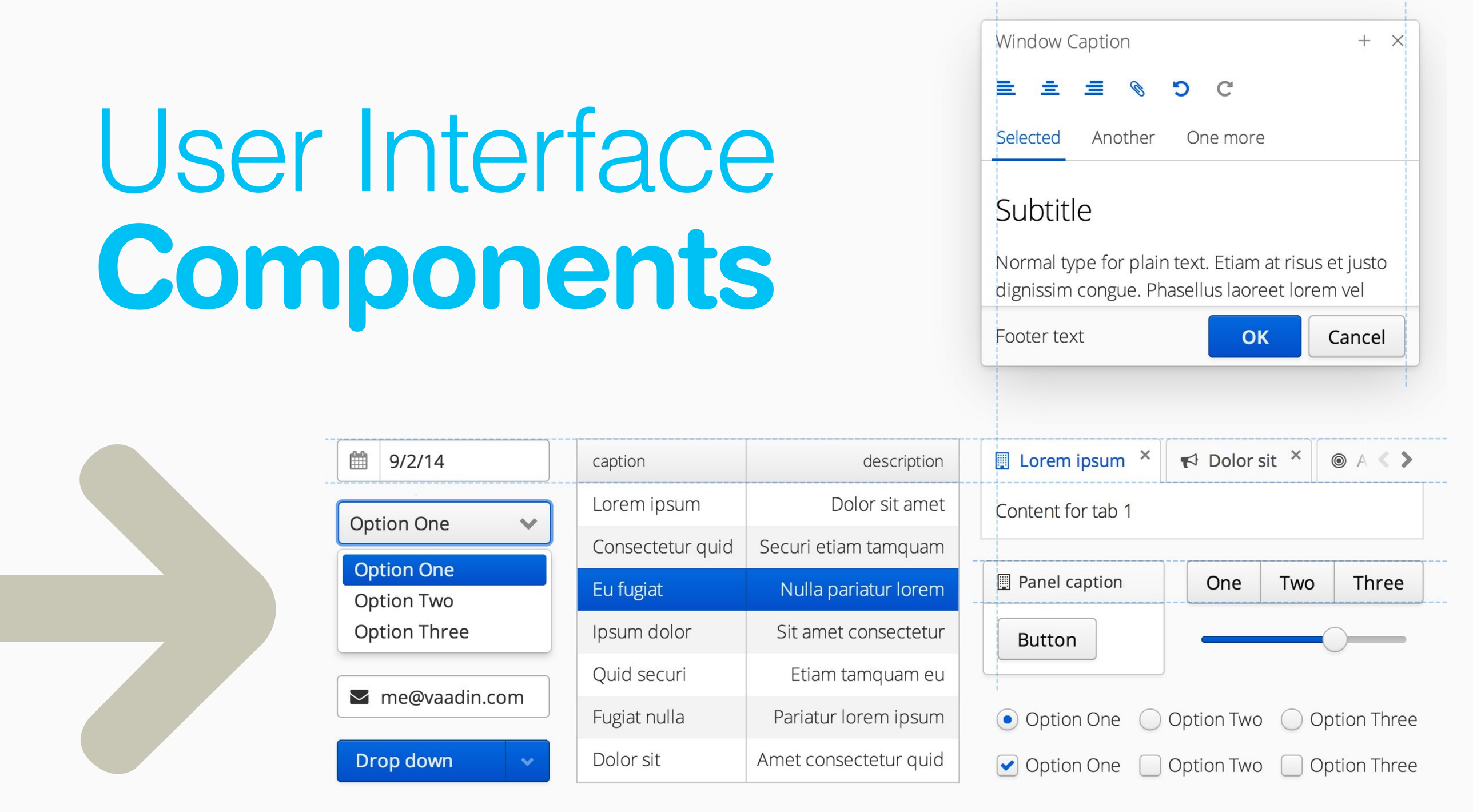
vaadin }>



Server driven UI framework for rich web applications

vaadin }>

User Interface **Components**

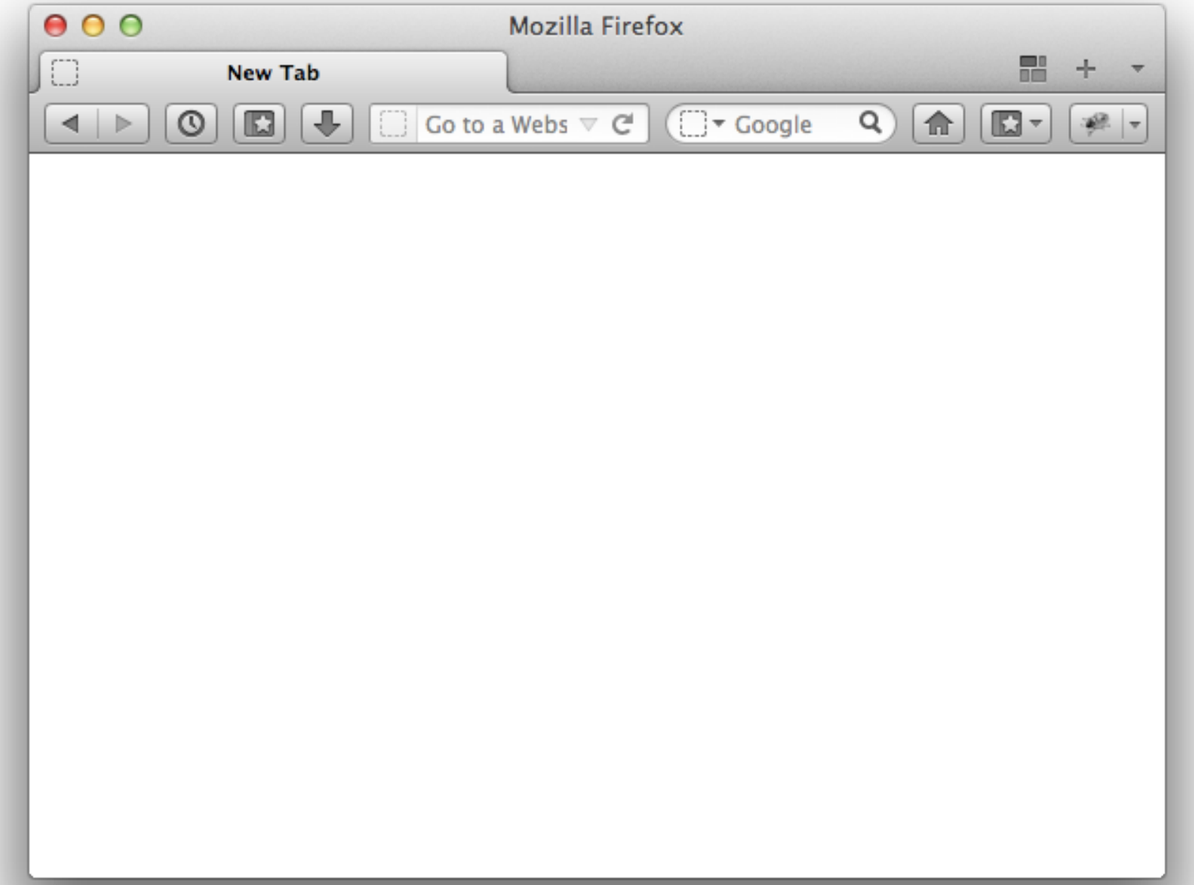


Developer Productivity + Rich UX

Server side UI How?

vaadin

```java
TextField name = new TextField("Name");
Button greetButton = new Button("Greet");

layout.addComponents(name, greetButton);

greetButton.addClickListener(
        e -> Notification.show("Hi " + name.getValue()));
```

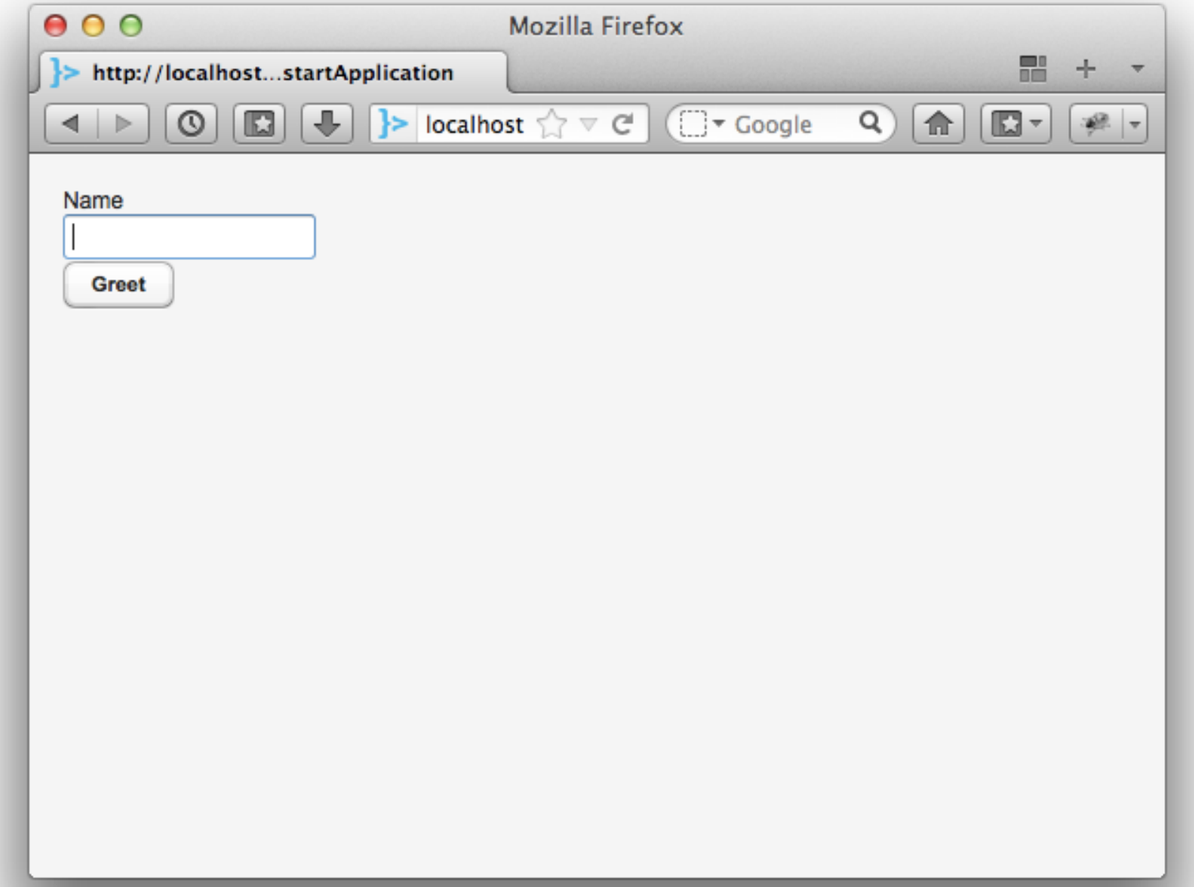


- Loader page
- CSS Theme
- Images
- JavaScript

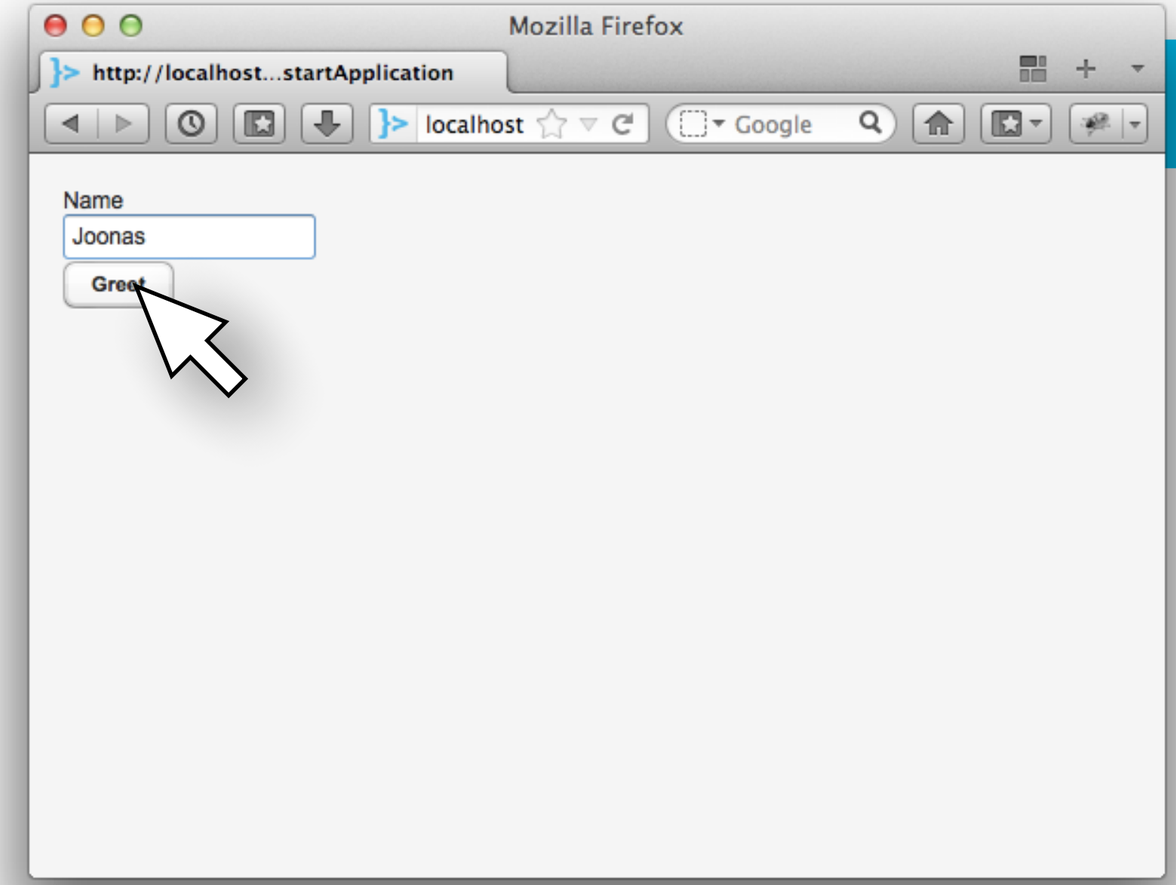
Compressed & reduced thin client

135k

vaadin


Mozilla Firefox
http://localhost...startApplication
localhost
Google
Name
Greet

vaadin

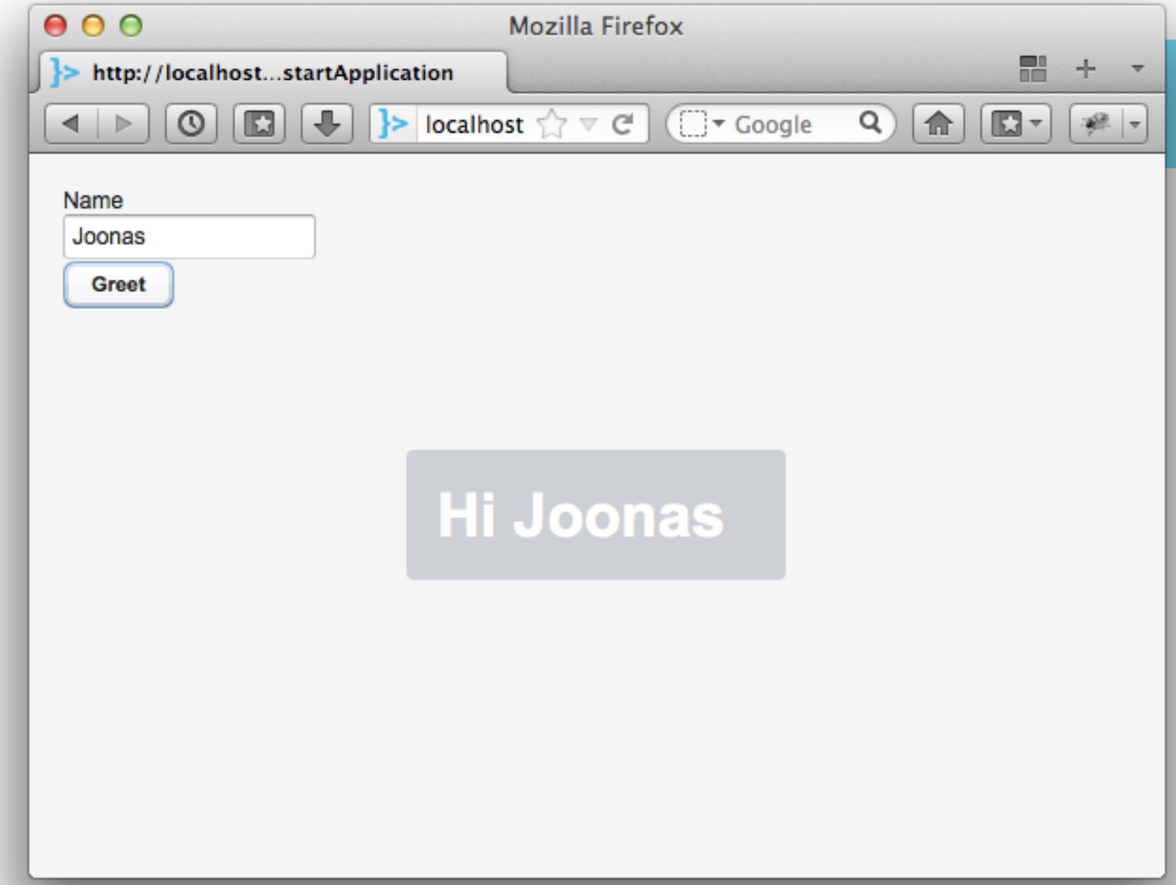


- name=”Joonas”
- button clicked

261 bytes

vaadin

```
TextField name = new TextField("Name");
Button greetButton = new Button("Greet");

layout.addComponents(name, greetButton);

greetButton.addClickListener(
        e -> Notification.show("Hi " + name.getValue()));
```



- name=”Joonas”
- button clicked

261 bytes

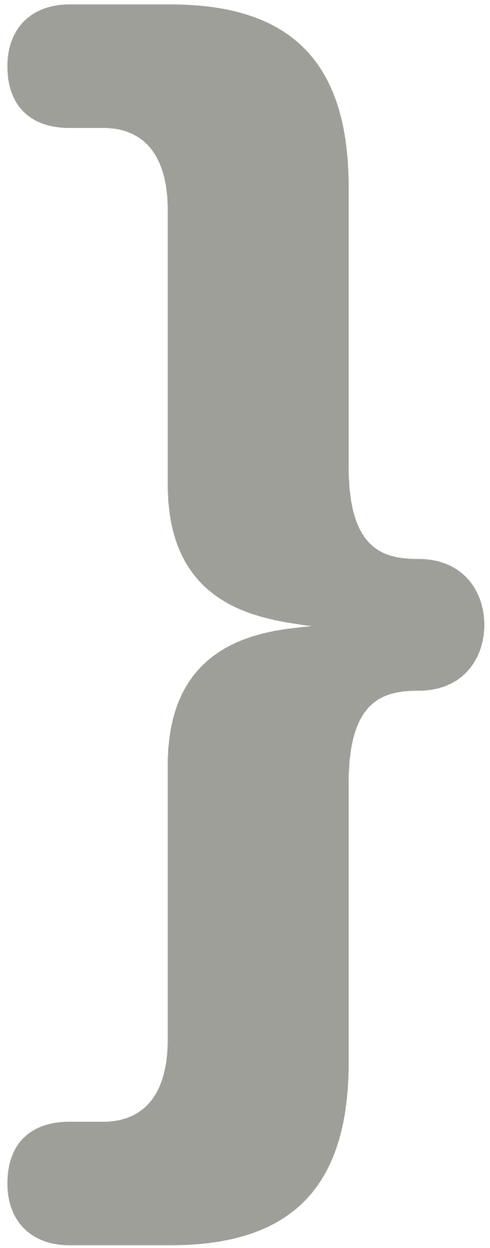
- Add notification

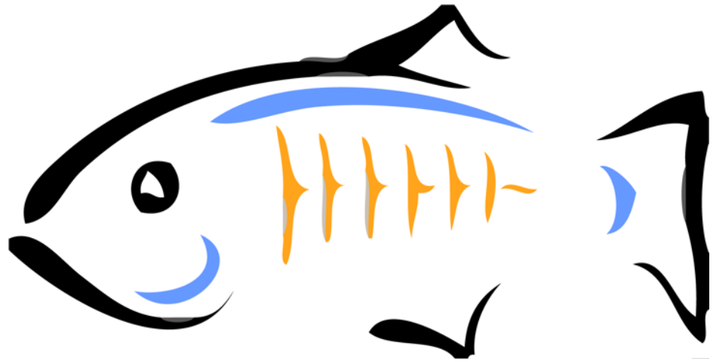
267 bytes

vaadin

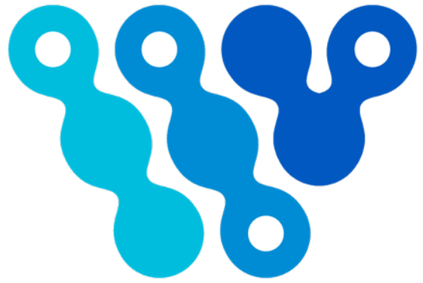
vaadin

- Collection of Java Specification Requests (JSRs)
- Implemented by app servers





vaadin



OpenWebBeans

eclipselink

Weld

Do you know some Java EE specs?

vaadin }>

Context and Dependency Injection 1.1 (CDI)
(JSR-340)

Java Transaction API 1.2 (JTA)
(JSR-907)

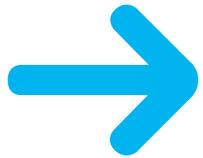
Java API for RESTful

Java API for XML based

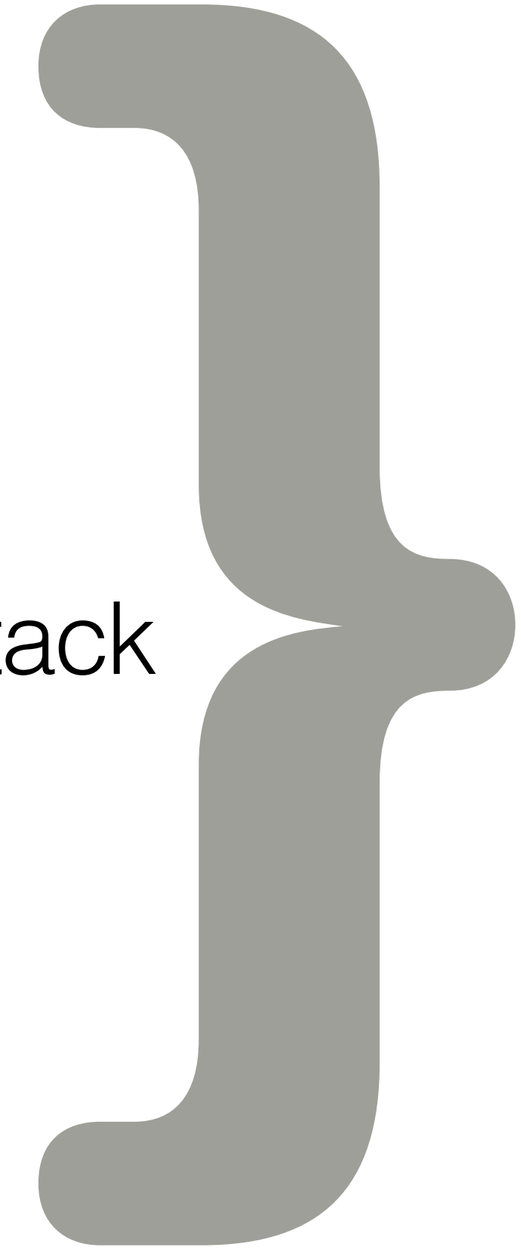
Java Architecture for XML Binding 2.2 (JAX-B)
(JSR-222)

vaadin

Which APIs should I know?



APIs that form your technology stack



Java Persistence API 2.1 (JPA)

vaadin

@Entity

Customer

```
@Id
@AutoGenerated
Long id;

@Column(nullable = false)
String name;

Date birthdate;
```

vaadin

@Entity

Customer

```
@Id
@AutoGenerated
Long id;

@Column(nullable = false)
String name;

Date birthdate;
```

Customer

Id	name	birthdate
1	Alex	07.02.1984
2	John	18.2.1992

@Entity

Customer

```
@Id
@AutoGenerated
Long id;

@Column(nullable = false)
String name;

Date birthdate;

@OneToMany(mappedBy="customer")
List<Invoice> invoices;
```

Customer

Id	name	birthdate
1	Alex	07.02.1984
2	John	18.2.1992

Invoice

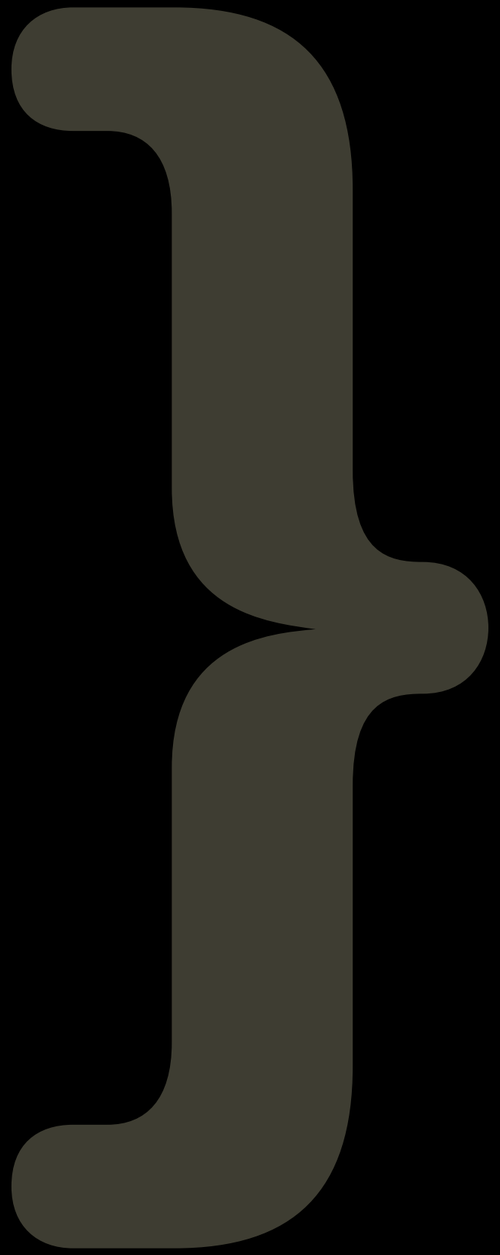
Id	customer	number
1	1	123
2	1	124

Enterprise Java Beans 3.2 (EJB)

vaadin

Enterprise Java Beans

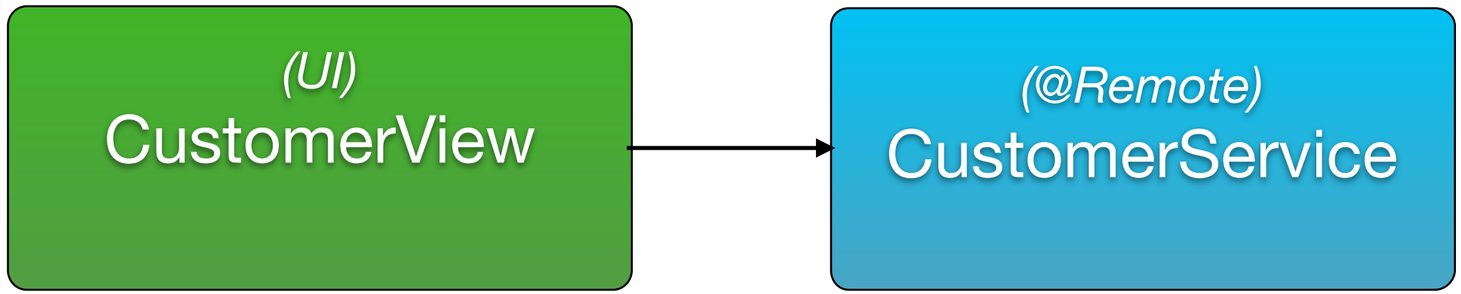
- Business layer services
- @local and @remote
- Transaction boundaries



vaadin }>

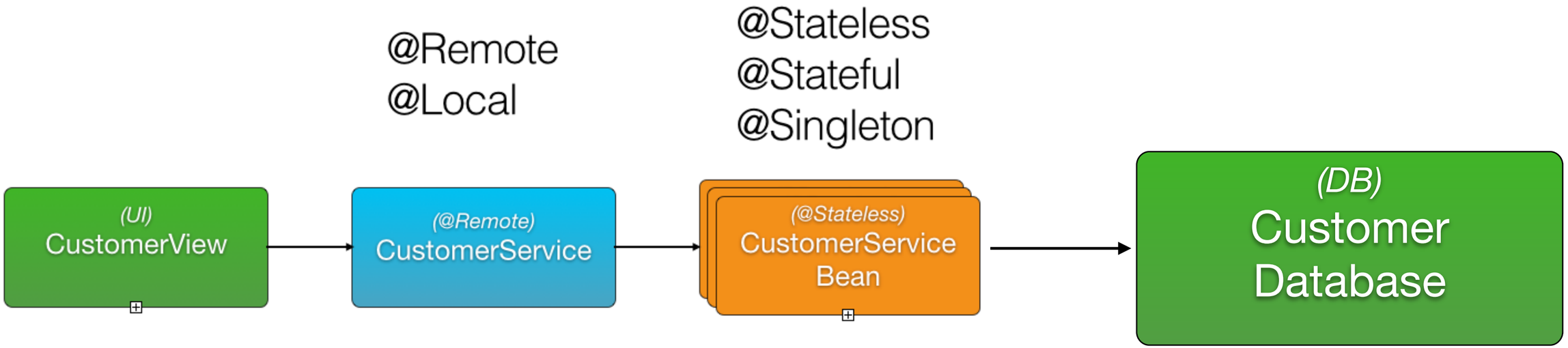

(UI)
CustomerView

@Remote
@Local



@Remote
@Local

@Stateless
@Stateful
@Singleton


@Remote
@Local
@Stateless
@Stateful
@Singleton
(UI)
CustomerView
(@Remote)
CustomerService
(@Stateless)
CustomerService
Bean
(DB)
Customer
Database

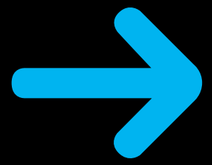
```java
@Local
public interface CustomerService {

  void storeCustomers(Collection<Customer> customers);

  void removeCustomers(Collection<Customer> customers);

  Collection<Customer> getAllCustomers()

  Optional<Customer> getCustomerByName(String name);
}
```

```java
@Stateless
public class CustomerServiceBean implements
  CustomerService {

  @PersistenceContext
  private EntityManager em;

  public void storeCustomers(Collection<Customer> cu) {
    cu.forEach(c -> storeCustomer(c));
  }

  public void storeCustomer(Customer c) {
    em.merge(c);
  }
}
```

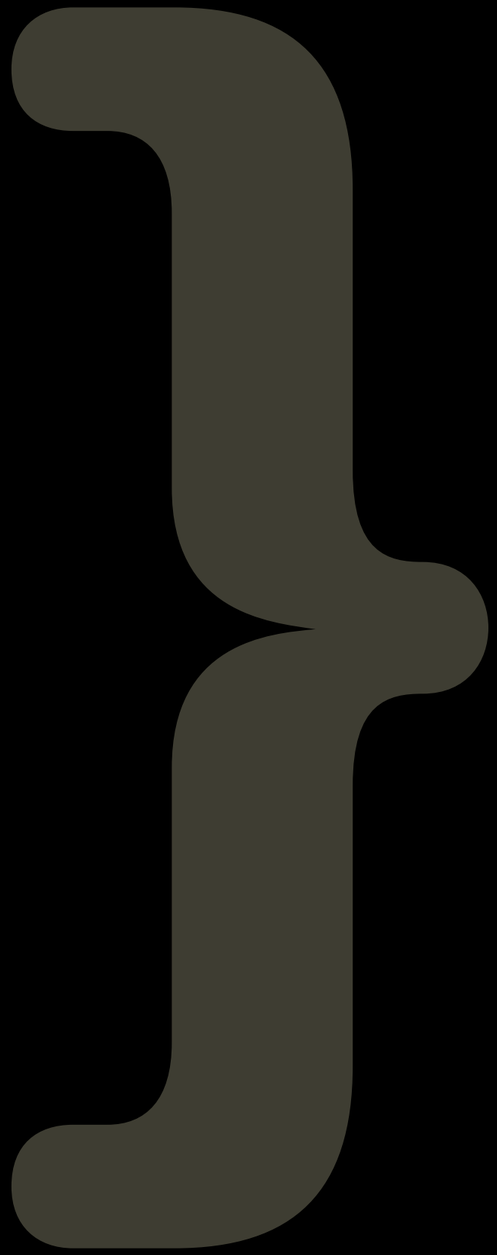
Context and Dependency Injection 1.2 (CDI)

vaadin

Context and Dependency Injection



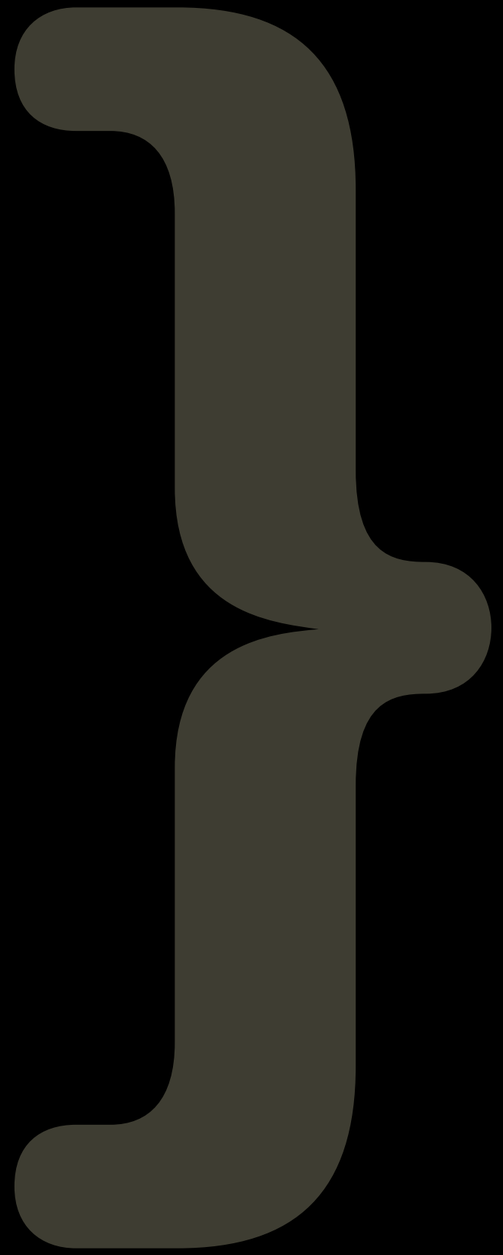
Instead of saying **new** say **@Inject**



vaadin }>

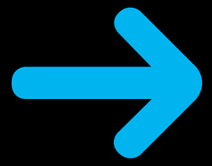
Context and Dependency Injection

- Instead of saying **new** say **@Inject**
- Decouples code and lets container manage dependencies

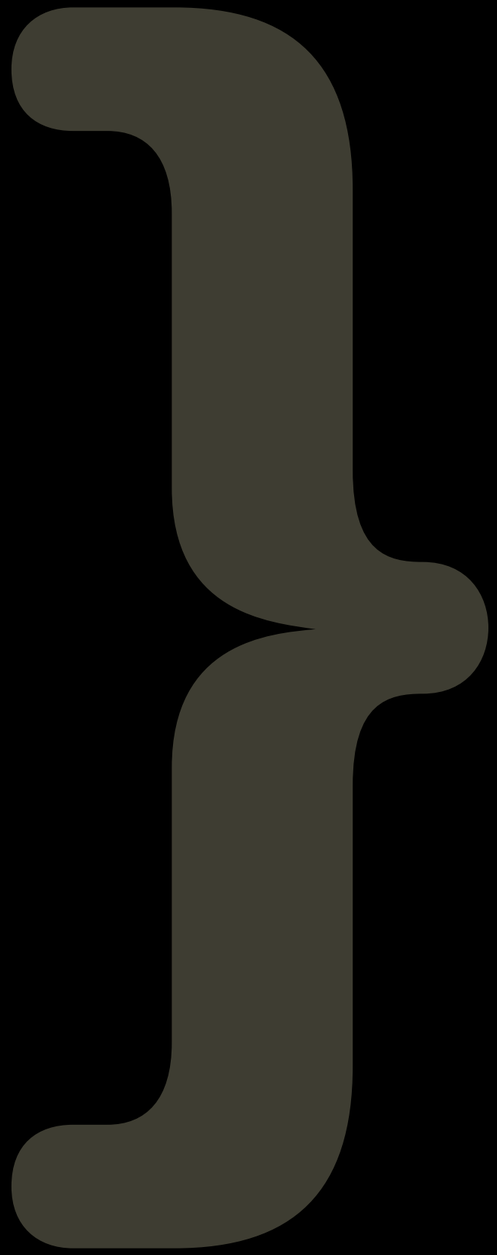


vaadin }>

Context and Dependency Injection



Object references by **scopes**

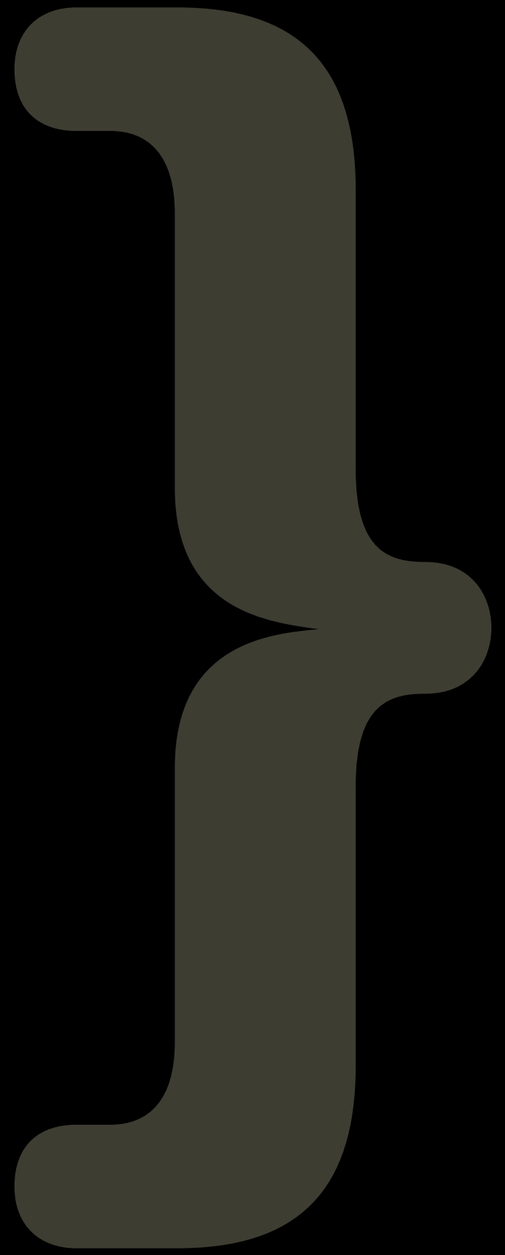


vaadin }>

Context and Dependency Injection

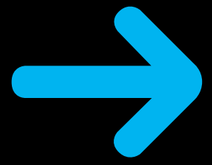
- Object references by **scopes**

- @ApplicationScoped
- @SessionScoped
- @RequestScoped

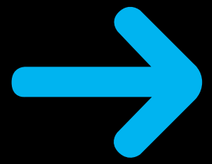


vaadin }>

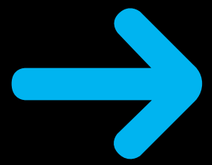
Context and Dependency Injection



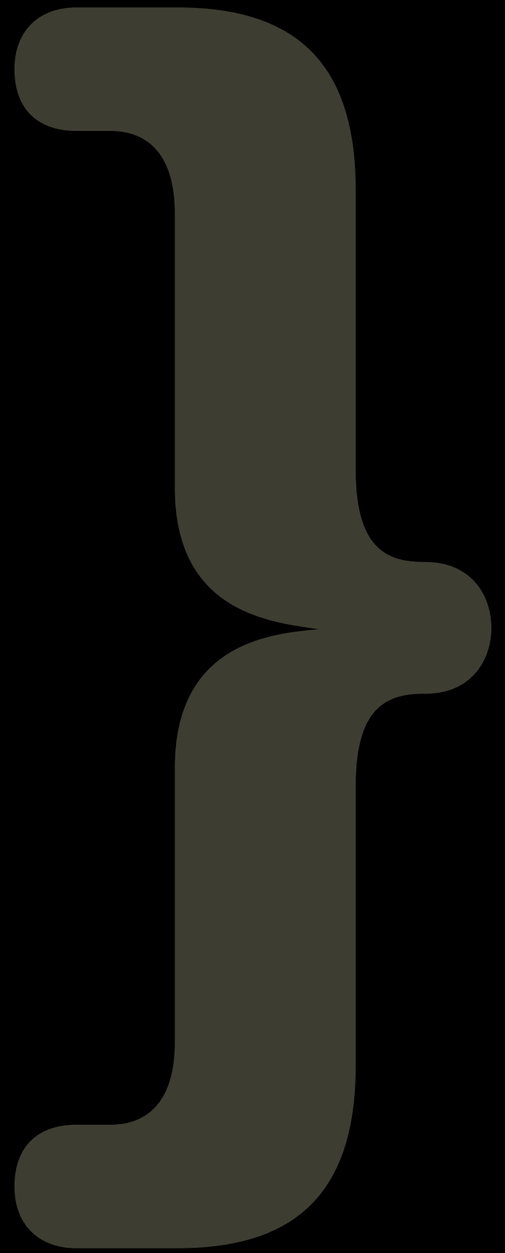
Object references by **scopes**



@ApplicationScoped
@SessionScoped
@RequestScoped



@UIScoped
@ViewScoped

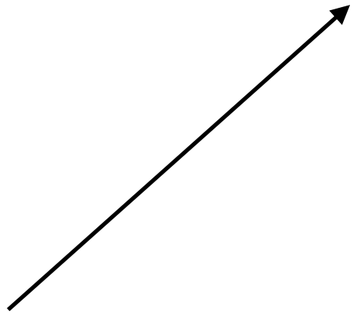


vaadin }>

@UIScoped

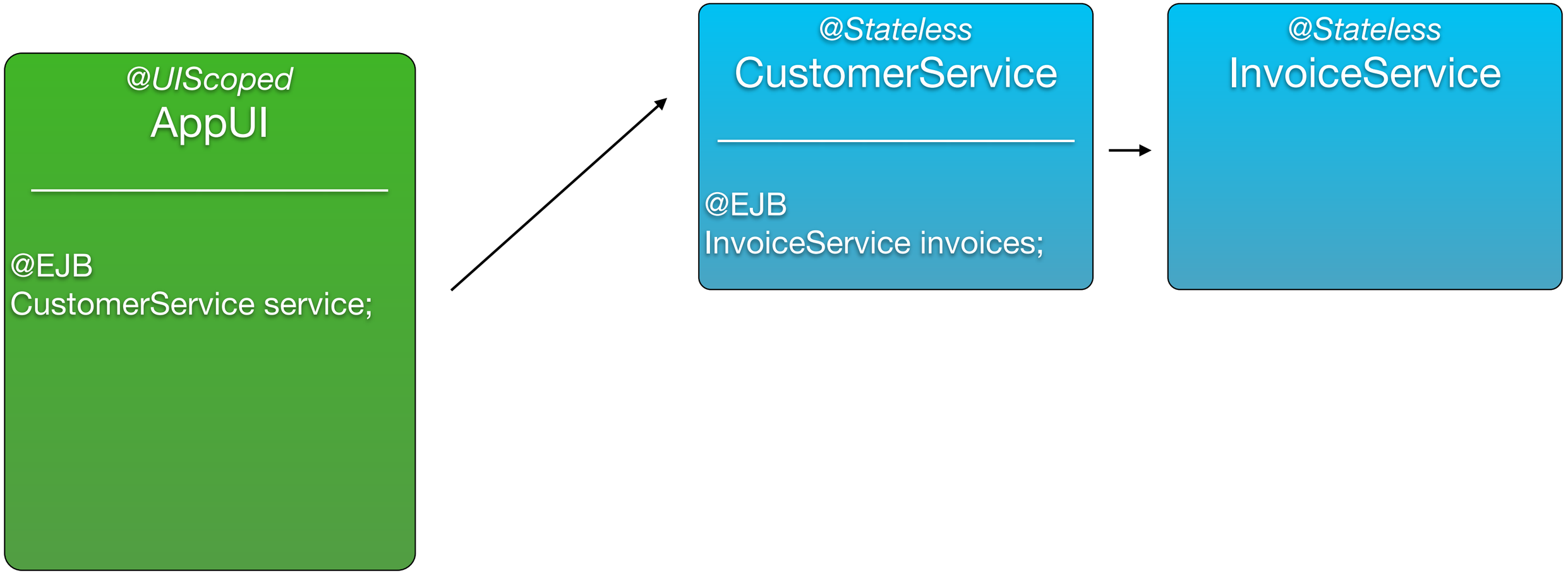
AppUI

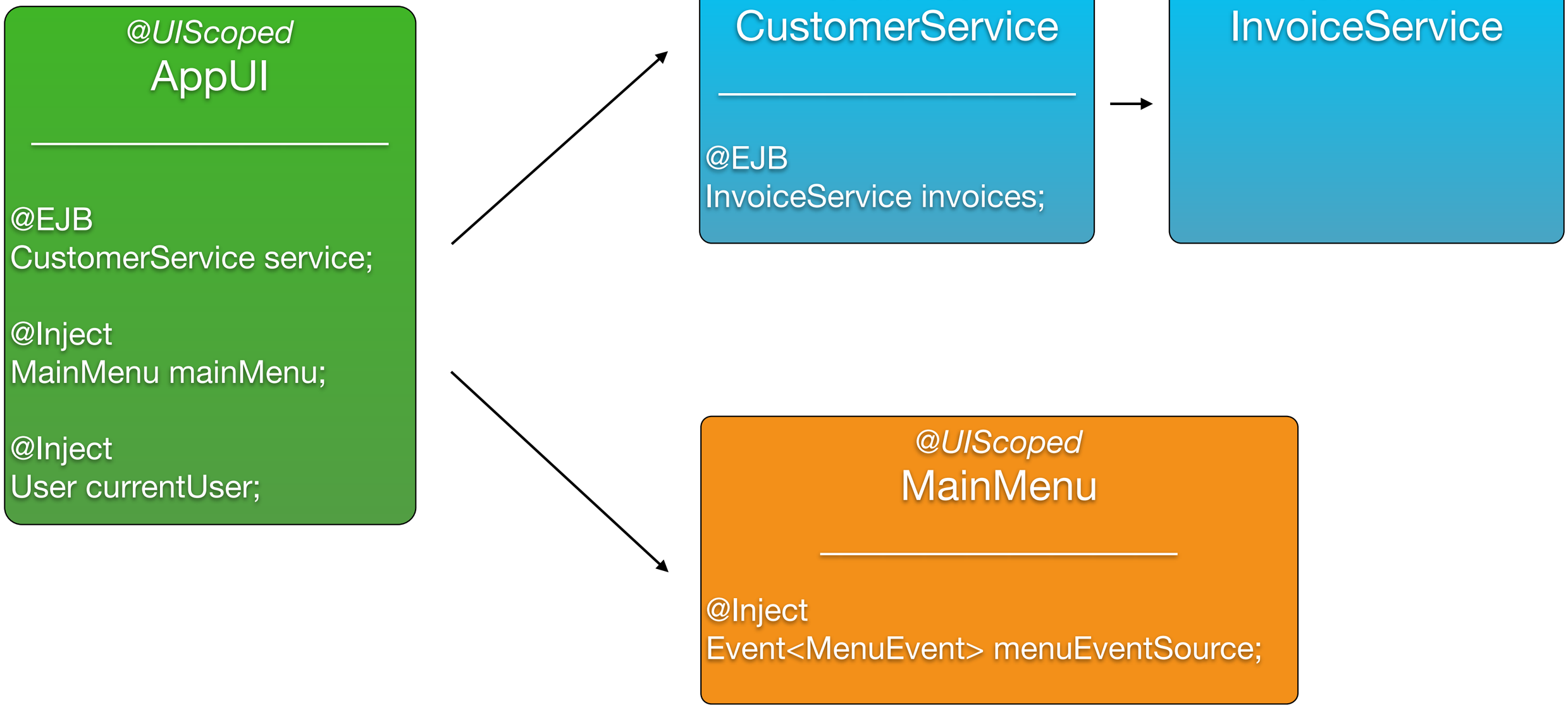
```
@EJB
CustomerService service;
```



@Stateless

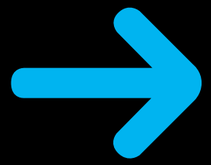
CustomerService


@UIScoped
AppUI
@EJB
CustomerService service;
@Stateless
CustomerService
@EJB
InvoiceService invoices;
@Stateless
InvoiceService

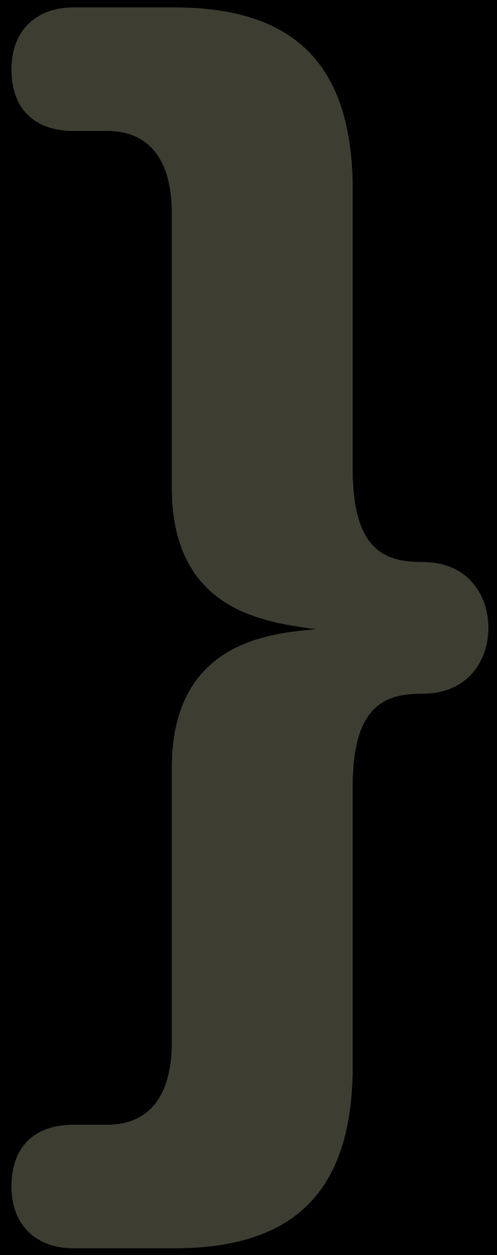

@UIScoped
AppUI
@EJB
CustomerService service;
@Inject
MainMenu mainMenu;
@Inject
User currentUser;
@Stateless
CustomerService
@EJB
InvoiceService invoices;
@Stateless
InvoiceService
@UIScoped
MainMenu
@Inject
Event<MenuEvent> menuEventSource;

Integration to EE through Vaadin CDI

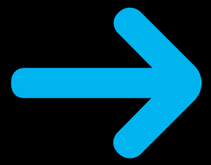
vaadin



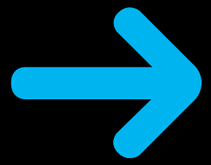
Managed UI with @CDIUI



vaadin }>

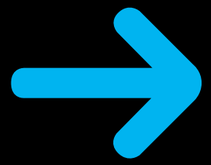


Managed UI with @CDIUI

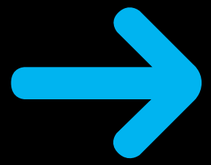


Allows injection with @Inject and @EJB

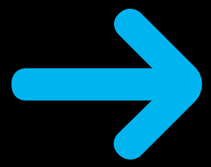
vaadin }>



Managed UI with @CDIUI



Allows injection with @Inject and @EJB



Easily reference EE objects

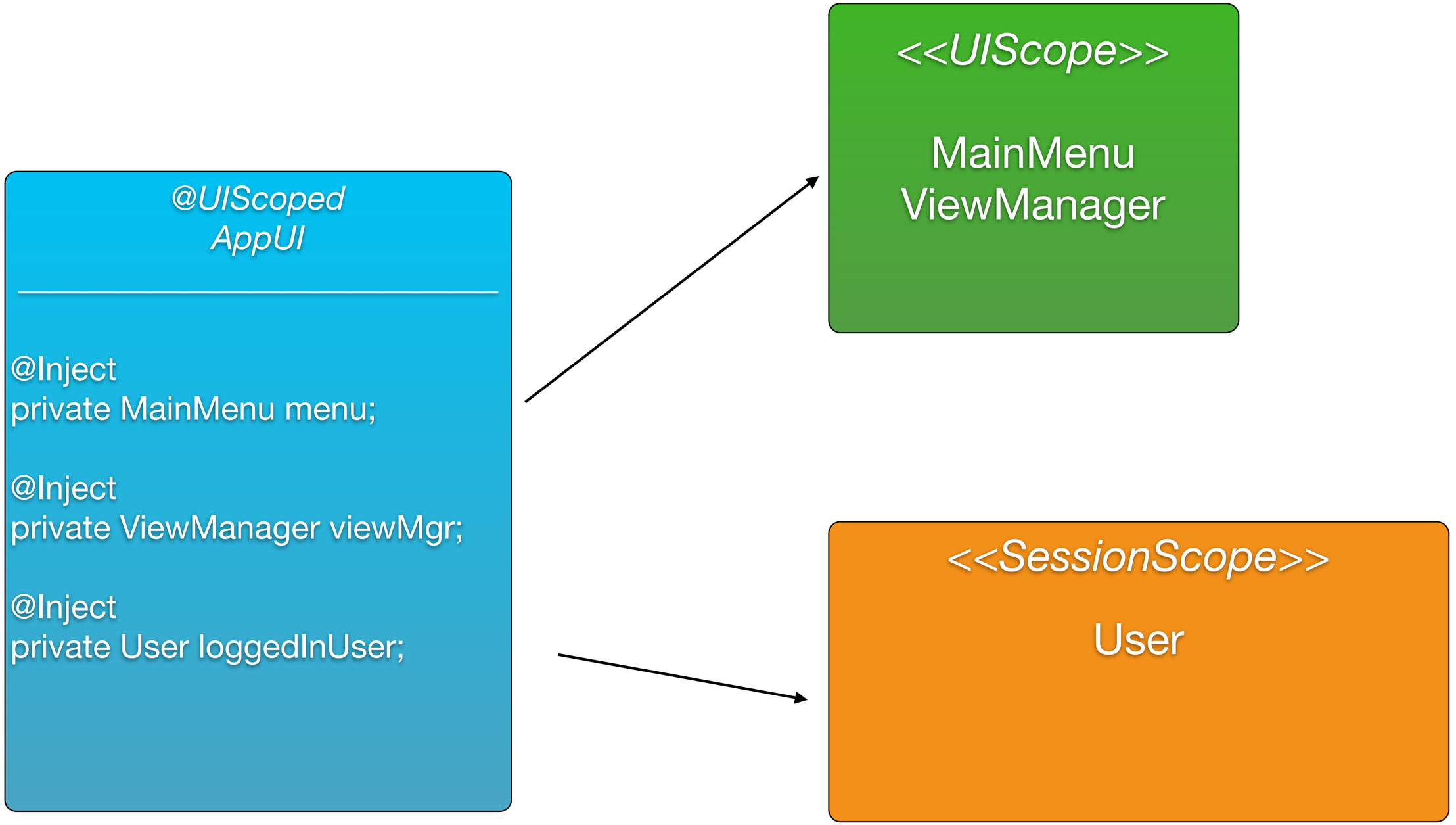
vaadin }>

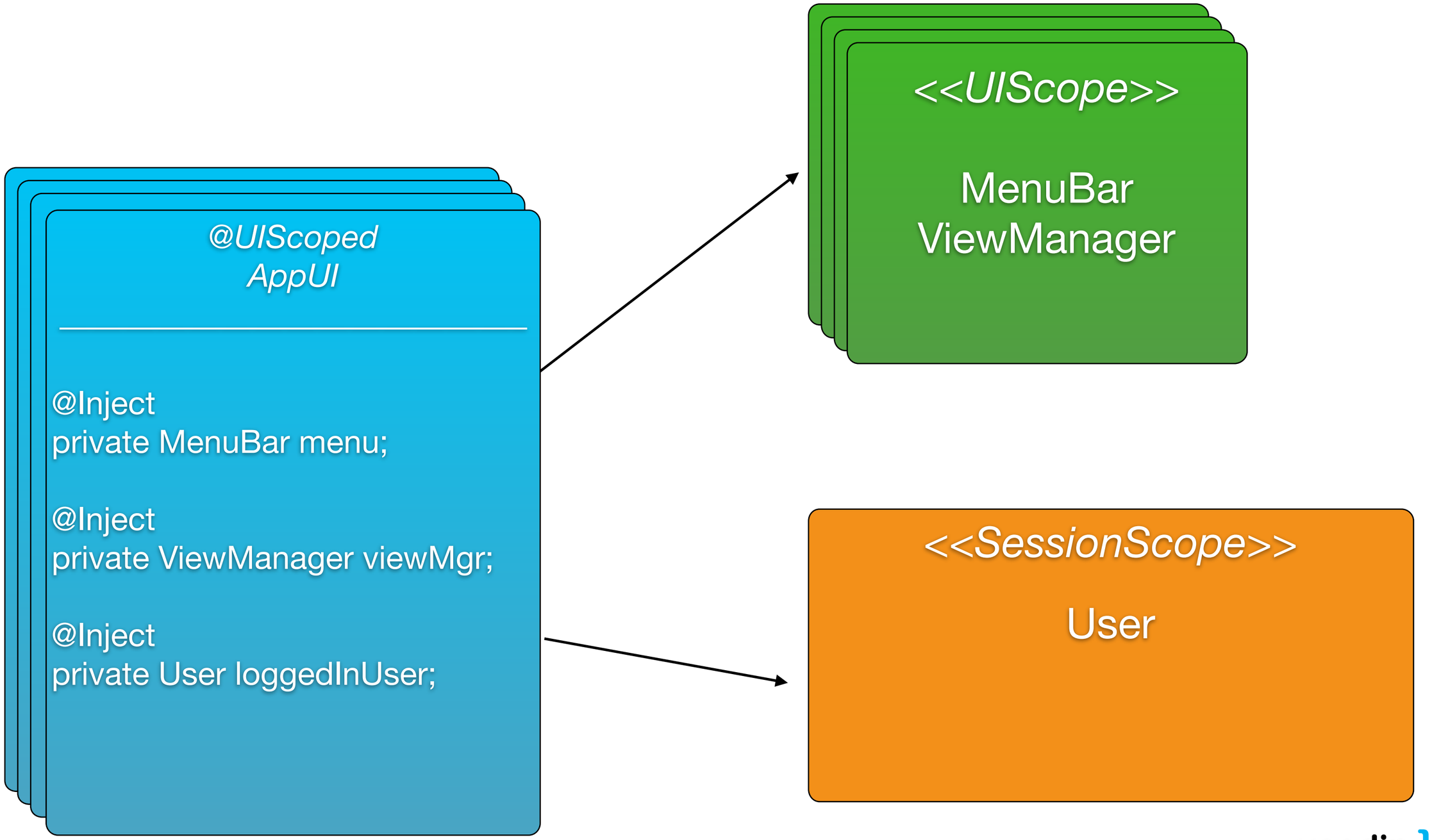
```
@CDIUI("")
public class AppUI extends UI {

}
```

vaadin }>

```
@CDIUI("")
public class AppUI extends UI {

   @Inject
   private MainMenu mainMenu;

   @Inject
   private User currentUser;

   @Inject
   private ViewManager viewManager;

   public void init(VaadinRequest request) {
      VerticalLayout layout = new VerticalLayout();
      layout.addComponent(mainMenu);

      setContent(layout);
   }
}
```

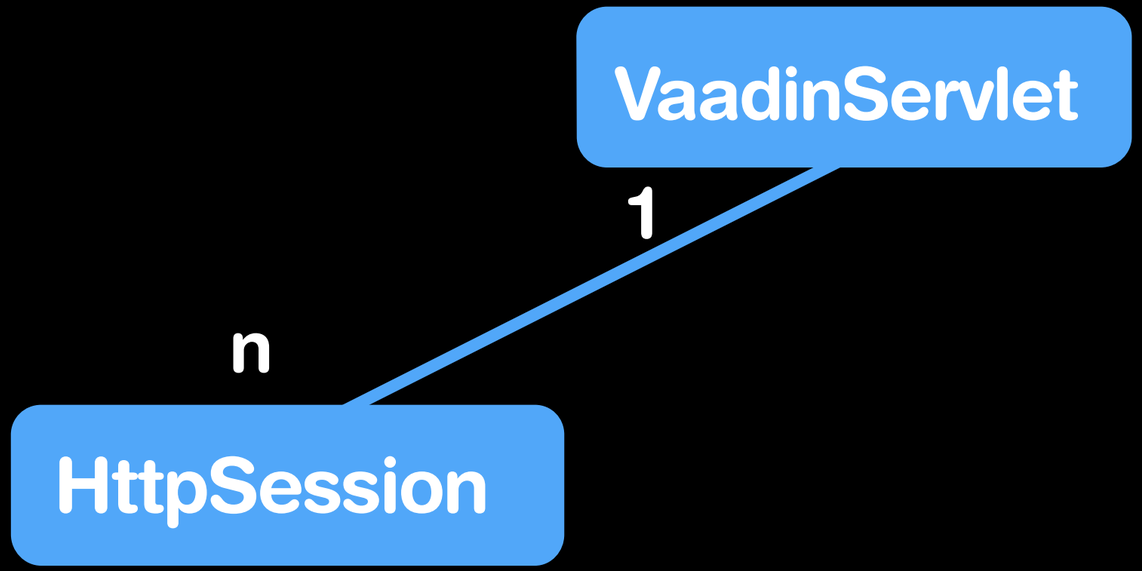
vaadin }>

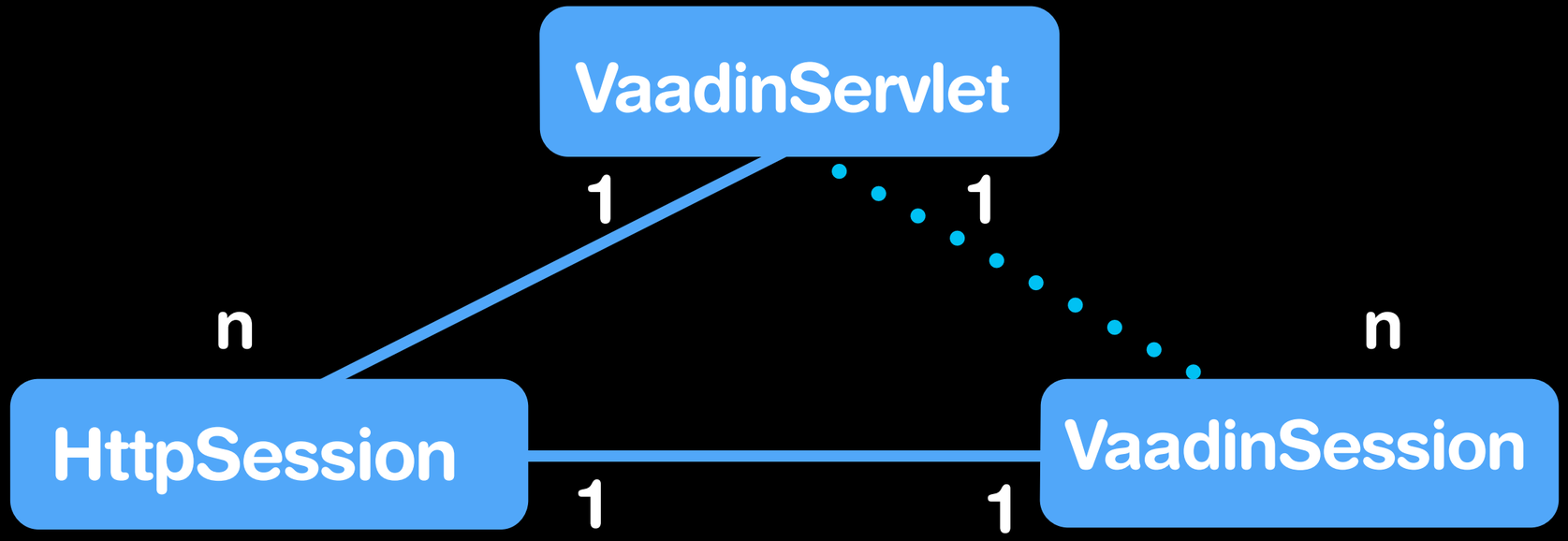

@UIScoped
AppUI
@Inject
private MainMenu menu;
@Inject
private ViewManager viewMgr;
@Inject
private User loggedInUser;
<<UIScope>>
MainMenu
ViewManager
<<SessionScope>>
User

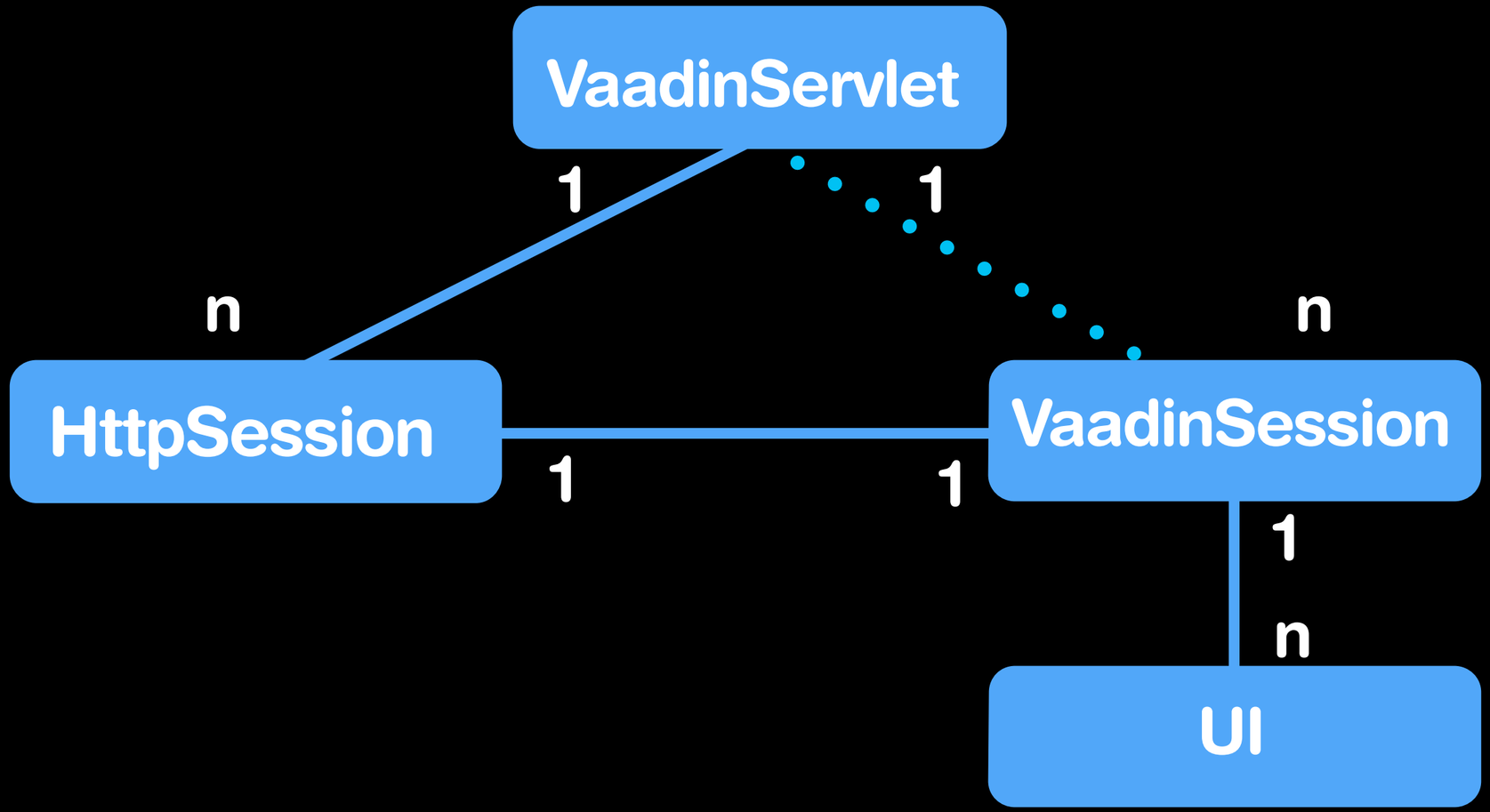

@UIScoped
AppUI
@Inject
private MenuBar menu;
@Inject
private ViewManager viewMgr;
@Inject
private User loggedInUser;
<<UIScope>>
MenuBar
ViewManager
<<SessionScope>>
User

VaadinServlet

vaadin


VaadinServlet
1
n
HttpSession


VaadinServlet
1
1
n
n
HttpSession
VaadinSession
1
1


VaadinServlet
1
1
n
n
HttpSession
VaadinSession
1
1
1
n
UI

@UIScoped

- UI specific bean references
- CDI context for mapping beans per UI
- @UIScoped

vaadin }>

```
@UIScoped
public class MainMenu extends CustomComponent {

}
```

vaadin }>

```java
@UIScoped
public class MainMenu extends CustomComponent {

   @Inject
   private Event<NavigationEvent> eventSource;


   protected void onMenuItemClicked(MenuItem item) {
      eventSource.fireEvent(new NavigationEvent(item));
   }
}
```

```
@CDIUI("")
public class AppUI extends UI {

   …

   protected void onNavigationEvent(@Observes
                                    NavigationEvent event) {

      viewMgr.navigateTo(event.getView());
   }
}
```

Structuring Vaadin App with **Model View Presenter**

vaadin

Do you like **spaghetti?**

vaadin

Do you like
spaghetti?

Let’s clean it!

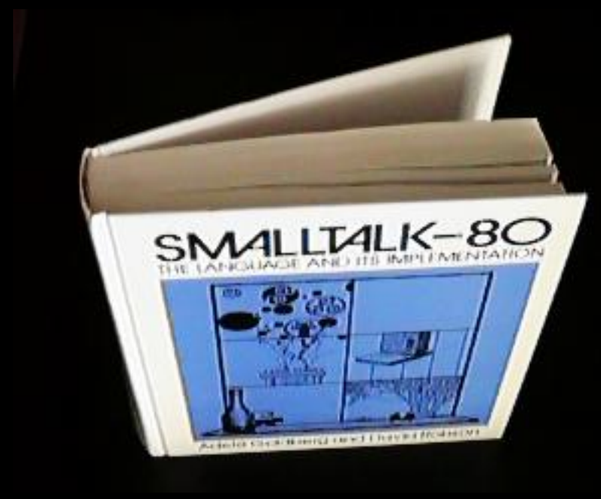
vaadin }>

History behind MVP

vaadin

Late 1970s

- Originally Model-View-Controller
- SmallTalk-80
- Controller is mediator between end user and application



vaadin

1980s

- Taligent Model-View-Presenter
- Influenced by SmallTalk-80
- Model, View, Presenter,
  Interactors, Commands, Selections
- Presenter orchestrates the
  structure, not the input

late 1980s

- Dolphin-Smalltalk MVP
- Simplified Taligent MVP
- View handles events by notifying presenter
- Presenter handles the logic, not the user input control

vaadin }>

UI layer MVP targets

Clean code best practices

like…

Single responsibility principle

Class should have only one reason to change

vaadin }>

SINGLE RESPONSIBILITY PRINCIPLE

Just Because You Can, Doesn't Mean You Should

Image credit: http://lostechies.com/derickbailey/2009/02/11/solid-development-principles-in-motivational-pictures/

vaadin

Open closed principle

Open for extension, closed for change

vaadin }>

OPEN CLOSED PRINCIPLE

Open Chest Surgery Is Not Needed When Putting On A Coat

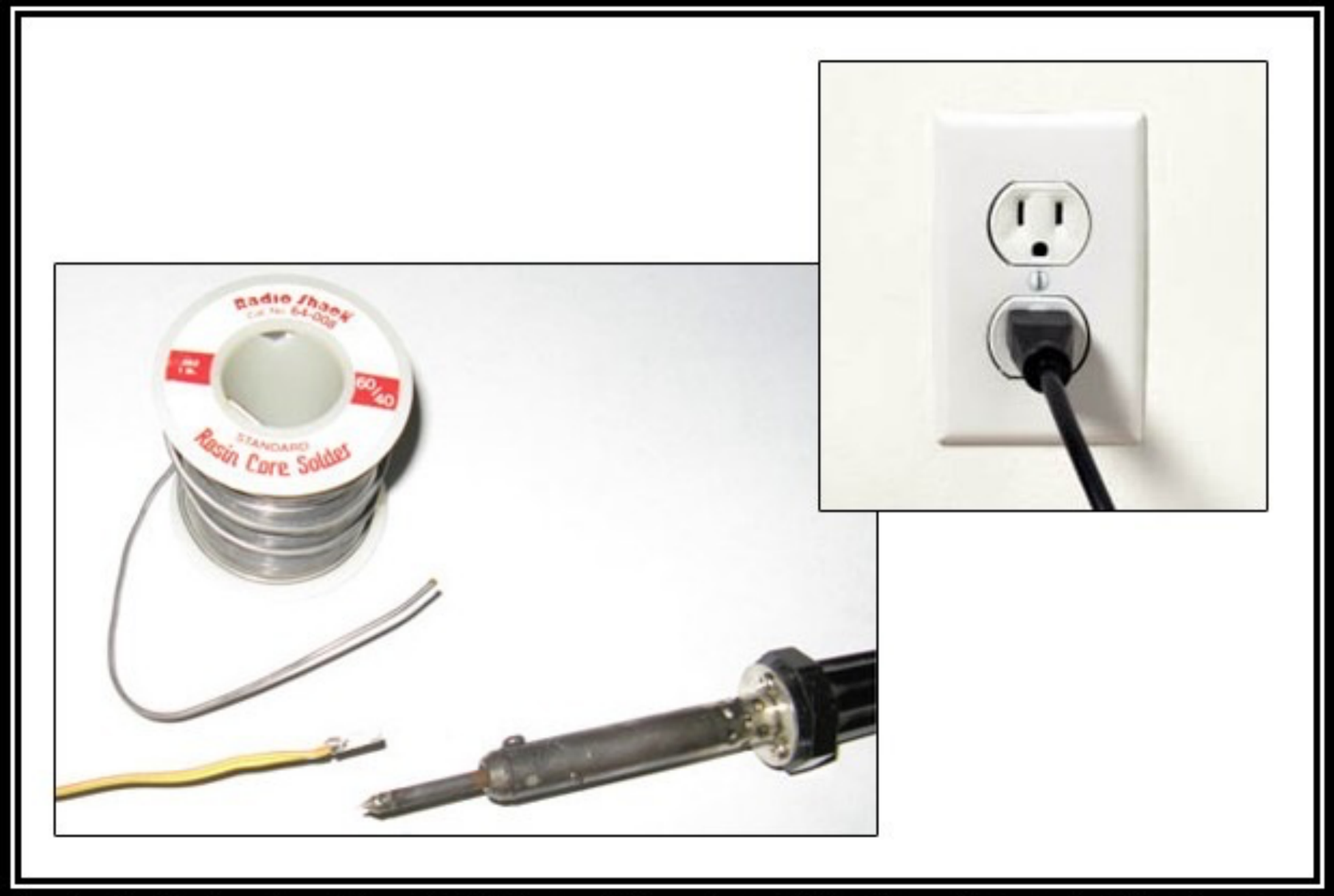
Image credit: http://lostechies.com/derickbailey/2009/02/11/solid-development-principles-in-motivational-pictures/

vaadin

Dependency Inversion principle

Depend on abstraction, not concretions

vaadin }>



DEPENDENCY INVERSION PRINCIPLE

Would You Solder A Lamp Directly To The Electrical Wiring In A Wall?

Image credit: http://lostechies.com/derickbailey/2009/02/11/solid-development-principles-in-motivational-pictures/

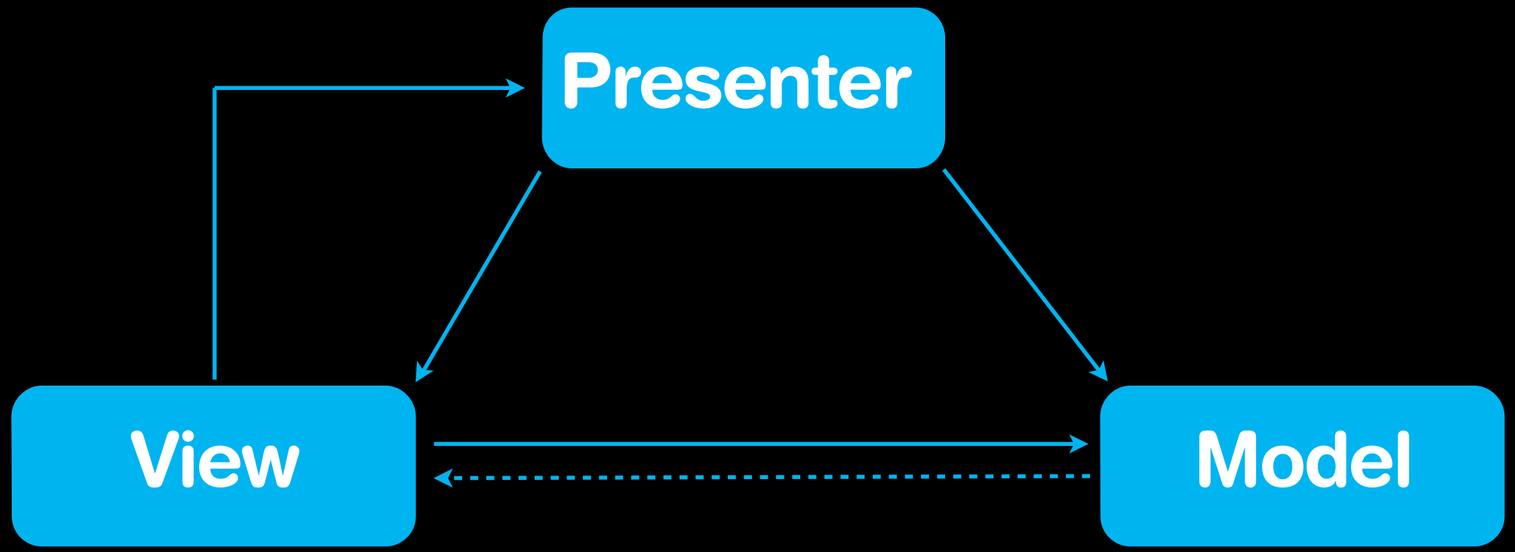
vaadin

Today
many MVP variations exists

Here's one…

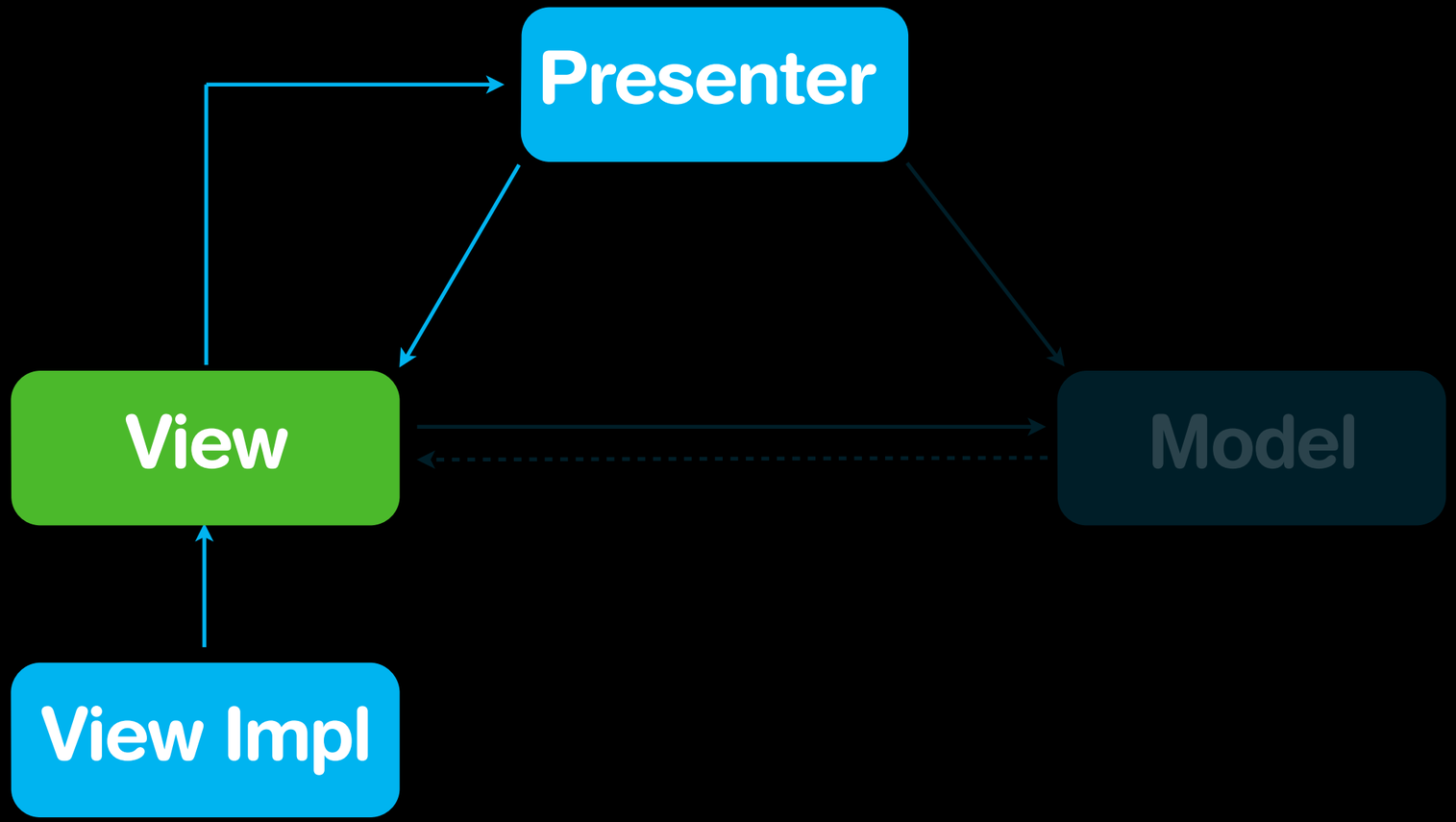
vaadin

Model-View-Presenter



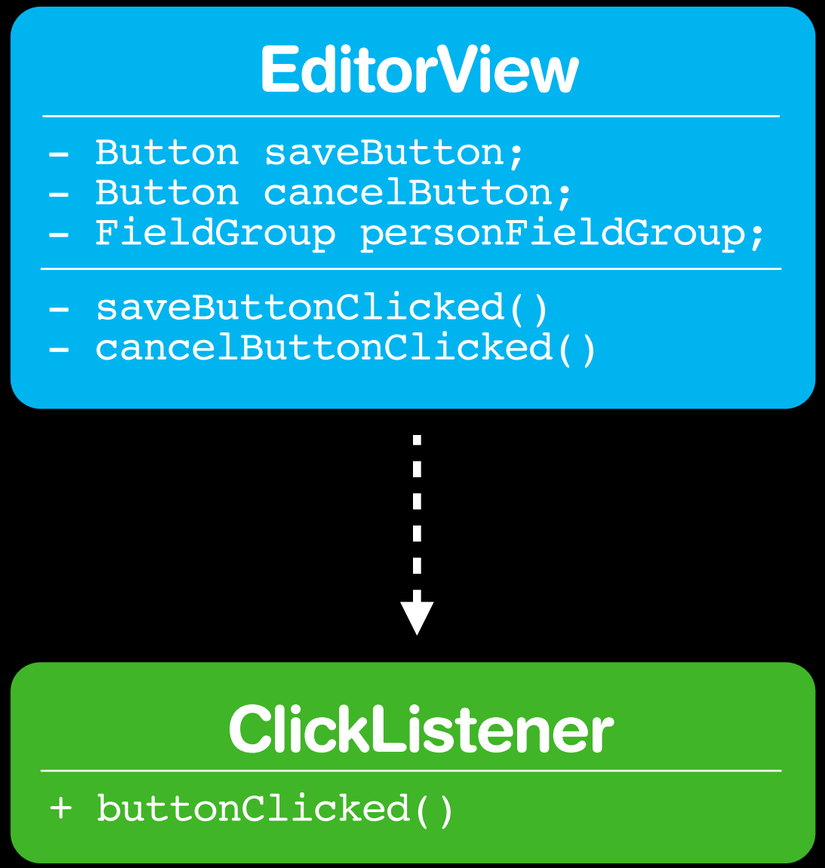
vaadin

Model-View-Presenter

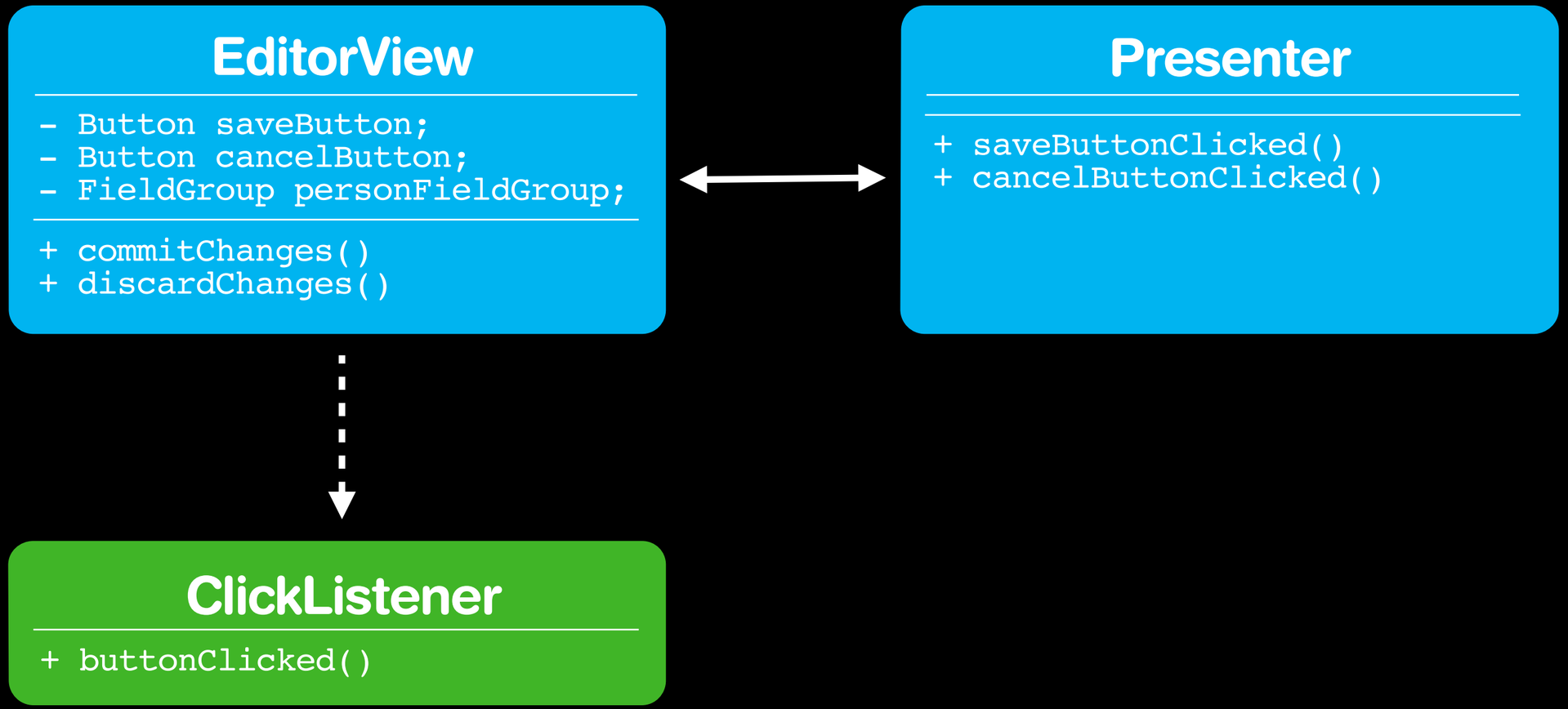


vaadin

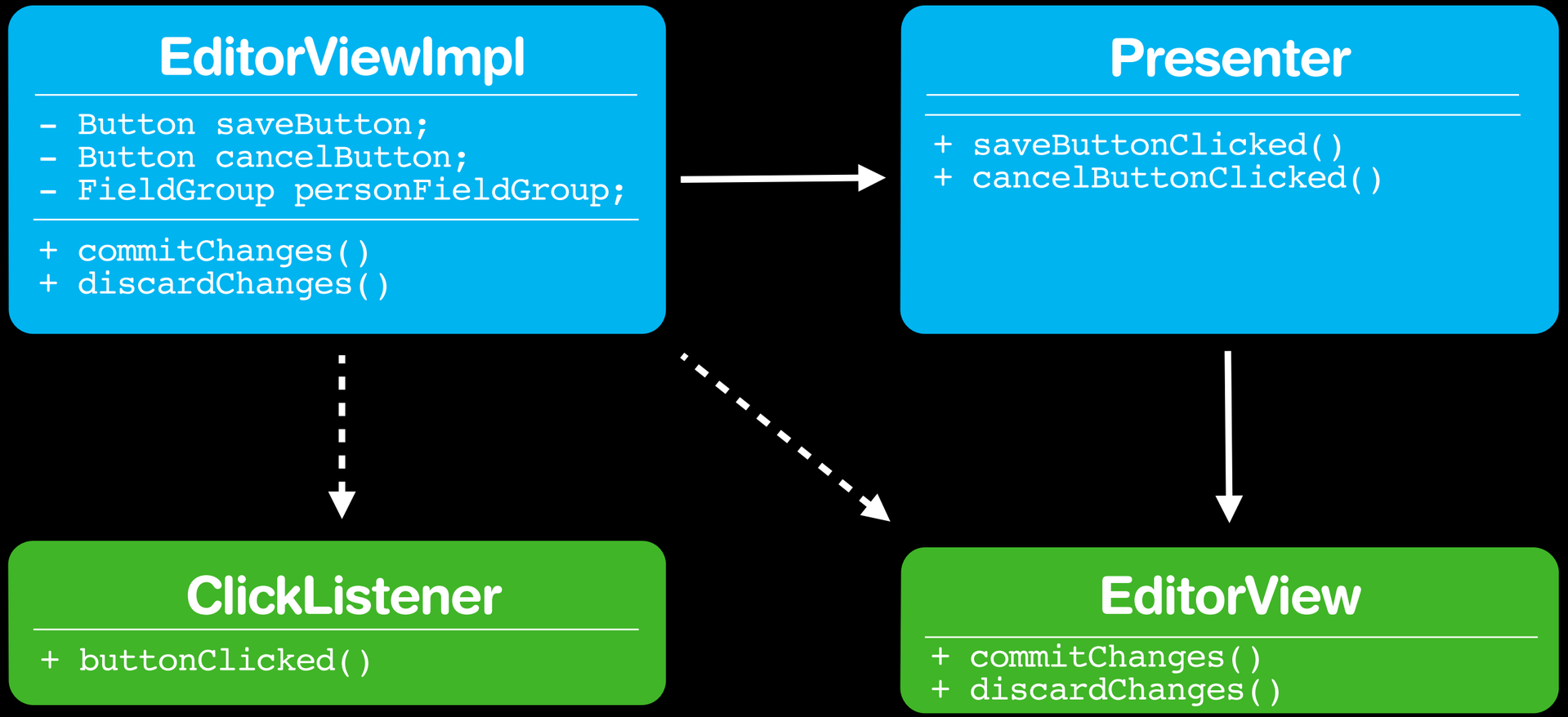
Example



Example

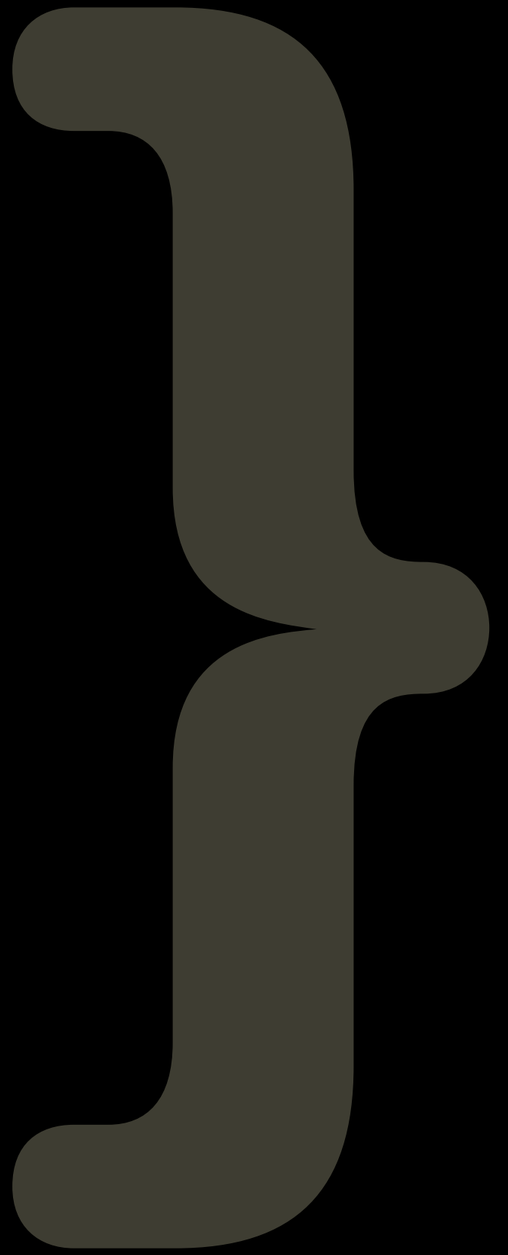

EditorView
- Button saveButton;
- Button cancelButton;
- FieldGroup personFieldGroup;
+ commitChanges()
+ discardChanges()
Presenter
+ saveButtonClicked()
+ cancelButtonClicked()
ClickListener
+ buttonClicked()

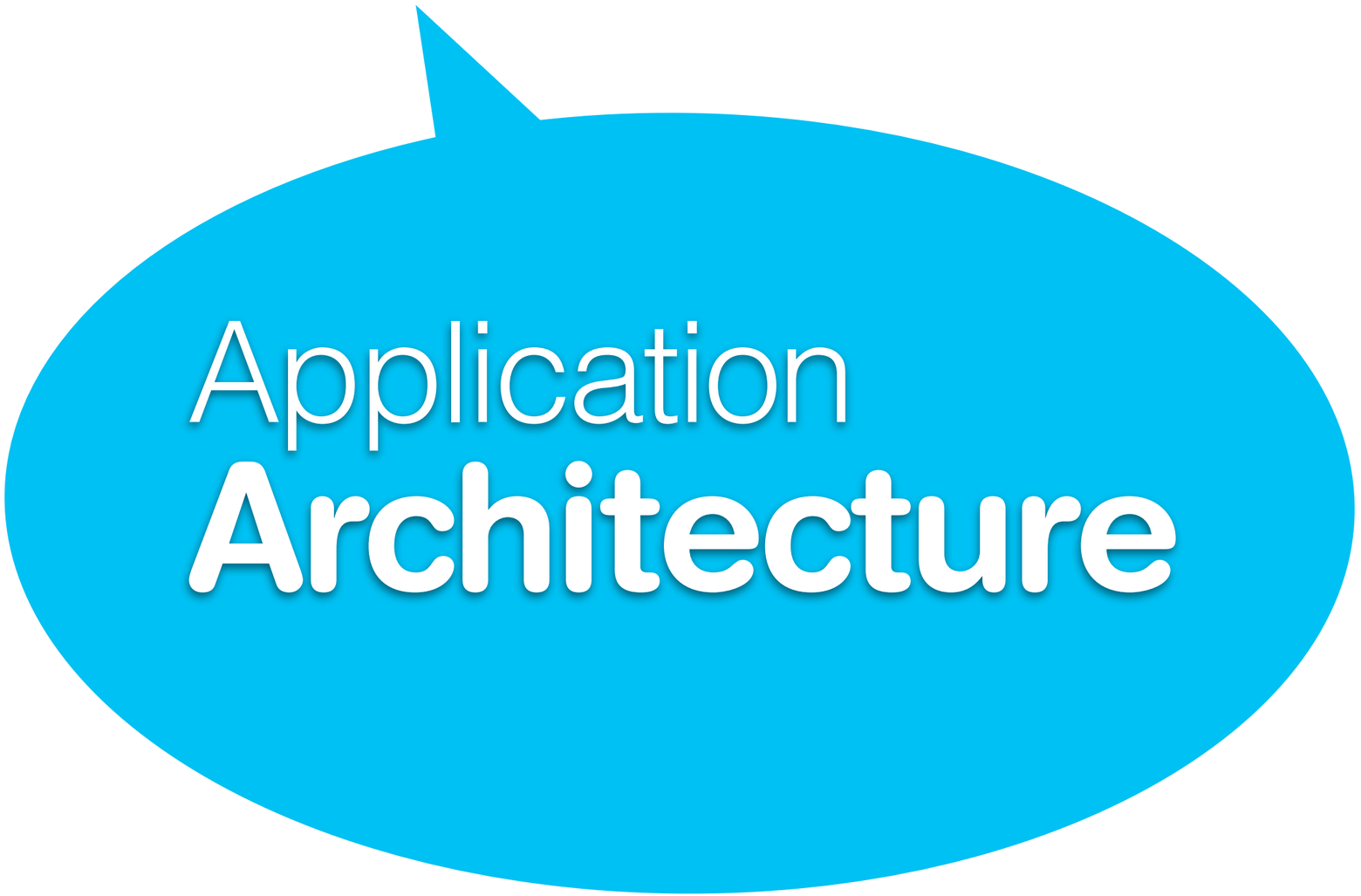
Example



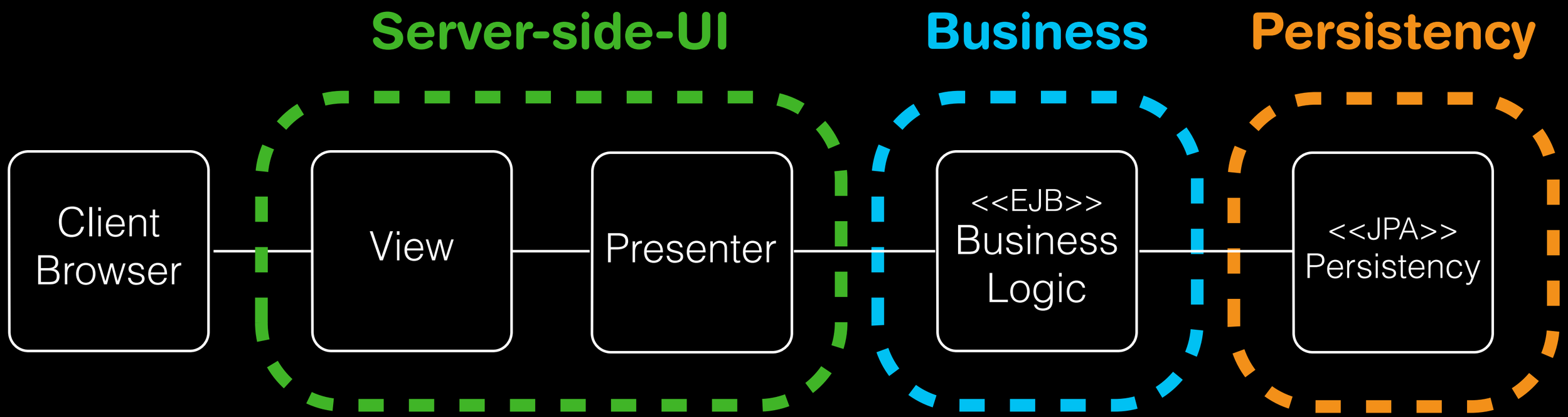
Benefits of MVP

- Simpler classes by SRP
- Complex UI logic separated from rendering
- Logic becomes easier to test by DIP



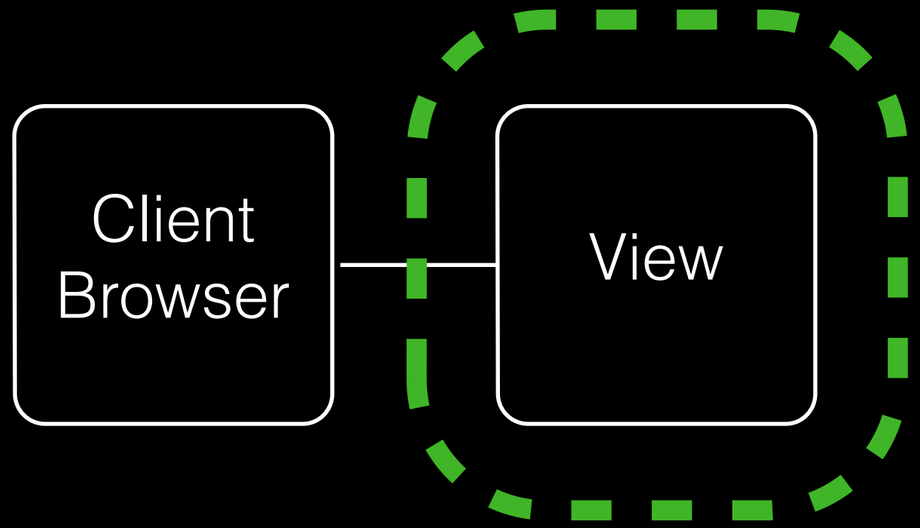


vaadin


Server-side-UI
Business
Persistency
Client Browser
View
Presenter
<<EJB>> Business Logic
<<JPA>> Persistency

vaadin

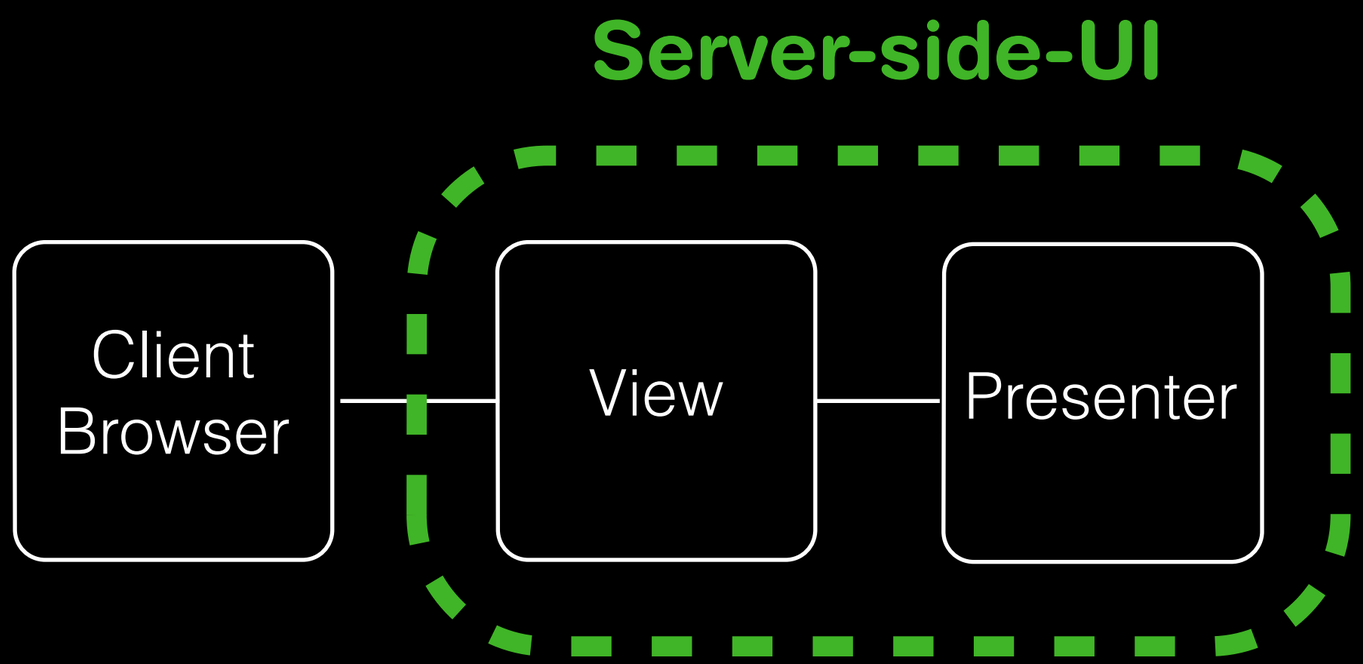
Server-side-UI



vaadin

```
public interface CustomerView extends
                        ApplicationView<CustomerViewPresenter> {

}
```

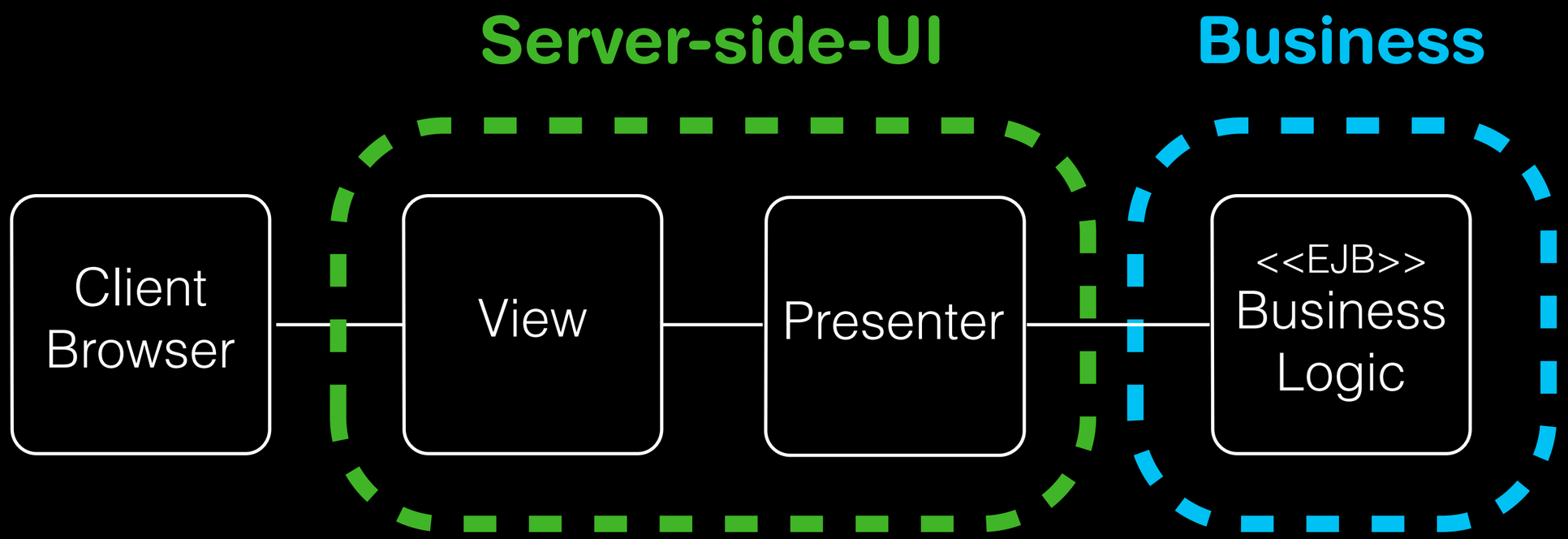
```
public interface CustomerView extends
                        ApplicationView<CustomerViewPresenter> {

    void populateCustomers(Collection<Customer> customers);

    void openEditorFor(Customer customer);

    void closeEditor();

    void removeTableSelection();
}
```


Server-side-UI
Client Browser
View
Presenter

```
@ViewScoped
public class CustomerViewPresenter extends AbstractPresenter<CustomerView> {

}
```

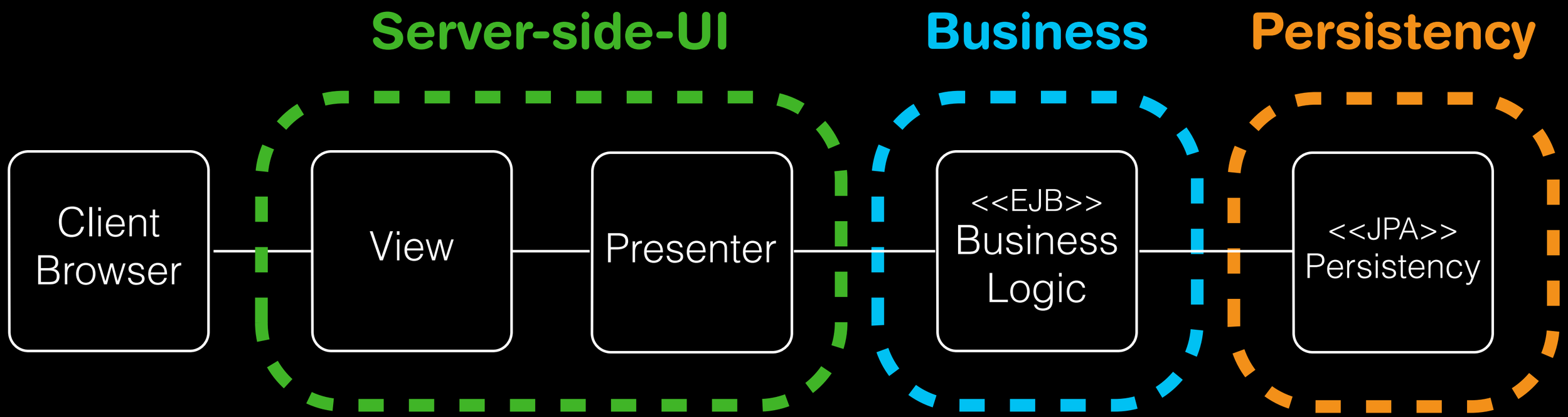
```java
@ViewScoped
public class CustomerViewPresenter extends AbstractPresenter<CustomerView> {

    @EJB
    private CustomerService customerService;

    @Override
    protected void onViewEnter() {
        getView().populateCustomers(customerService.getAllCustomers());
    }
}
```

```java
@ViewScoped
public class CustomerViewPresenter extends AbstractPresenter<CustomerView> {

    @EJB
    private CustomerService customerService;

    @Override
    protected void onViewEnter() {
        getView().populateCustomers(customerService.getAllCustomers());
    }

    public void onCustomerSaved(@Observes CustomerSavedEvent event) { … }

    public void onCustomerRemoved(@Observes CustomerRemovedEvent event) { … }

    public void onCustomerSelected(@Observes CustomerSelectedEvent event) { … }
}
```


Server-side-UI
Business
Client Browser
View
Presenter
<<EJB>>
Business Logic

```java
@Local
public interface CustomerService {

   void storeCustomers(Collection<Customer> customers);

   void removeCustomers(Collection<Customer> customers);

   Collection<Customer> getAllCustomers();

   Optional<Customer> getCustomerByUsername(String username);
}
```

```java
@Stateless
@TransactionAttribute(TransactionAttributeType.REQUIRED)
public class CustomerServiceBean implements CustomerService {

   @PersistenceContext(unitName = "appUnit")
   private EntityManager entityManager;

   @Override
   public void storeCustomers(Collection<Customer> customers) {
      customers.forEach(cu -> entityManager.merge(cu));
   }

   @Override
   public Collection<Customer> getAllCustomers() {
      return entityManager.createQuery(query).getResultList();
   }

   …
}
```


Server-side-UI
Business
Persistency
Client Browser
View
Presenter
<<EJB>> Business Logic
<<JPA>> Persistency

vaadin

```java
@Entity
public class Customer {

   @Id
   @AutoGenerated
   private Long id;

   private String name;

   @Temporal(DATE)
   private Date birthDate;

   public boolean isPersisted() {
      return id != null;
   }


    …
}
```

```
<persistence-unit name="appUnit" transaction-type="JTA">
   <jta-data-source>jdbc/app-backend</jta-data-source>
   <class>org.vaadin.example.backend.entity.Customer</class>

   <properties>
      <property name="…" … />
   </properties>
</persistence-unit>
```

How to

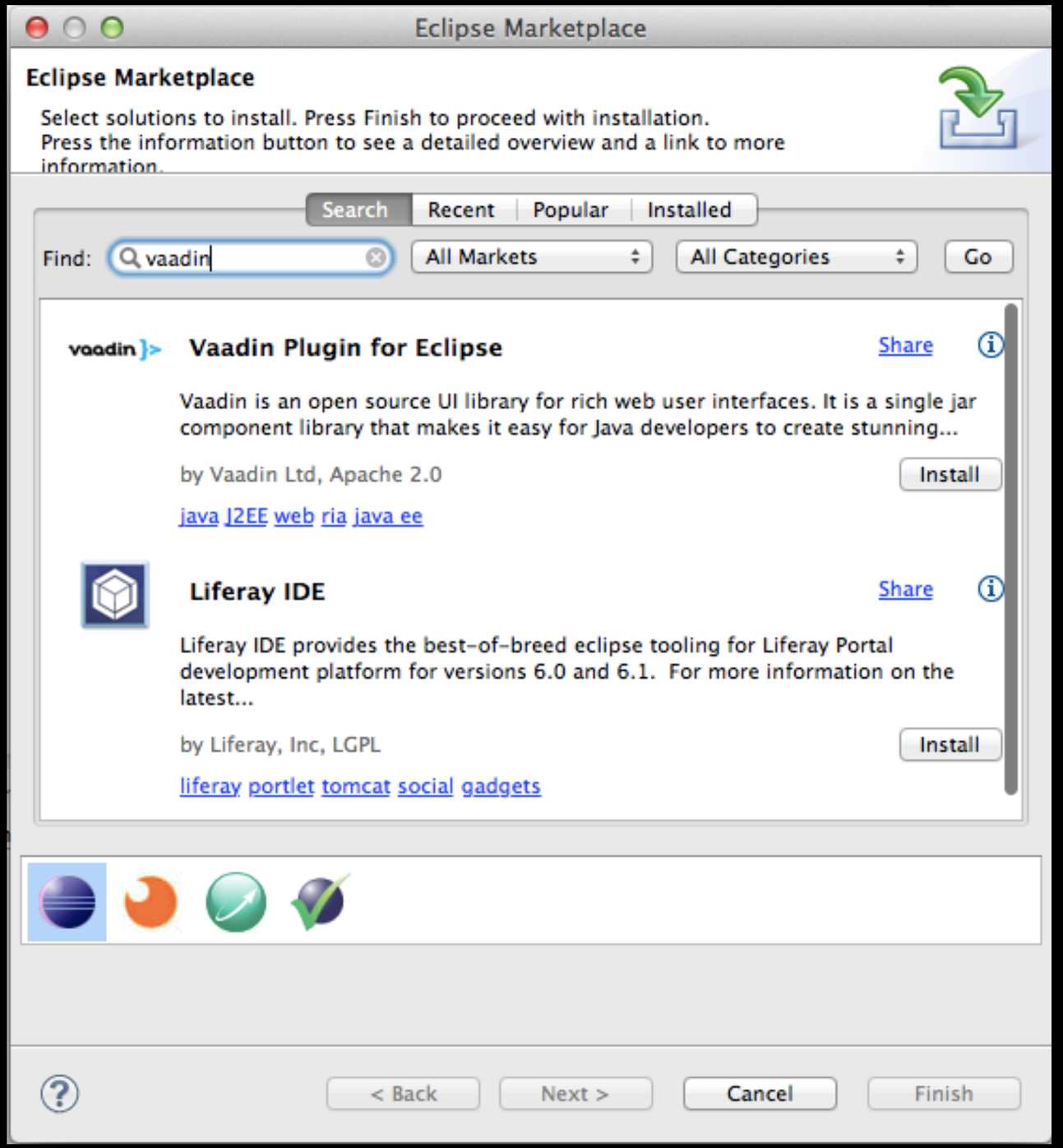
Get Started?

vaadin

```
<dependency>
  <groupId>com.vaadin</groupId>
  <artifactId>vaadin-cdi</artifactId>
  <version>1.0.2</version>
</dependency>

<repository>
   <id>vaadin-addons</id>
   <url>http://maven.vaadin.com/vaadin-addons</url>
</repository>
```

vaadin }>

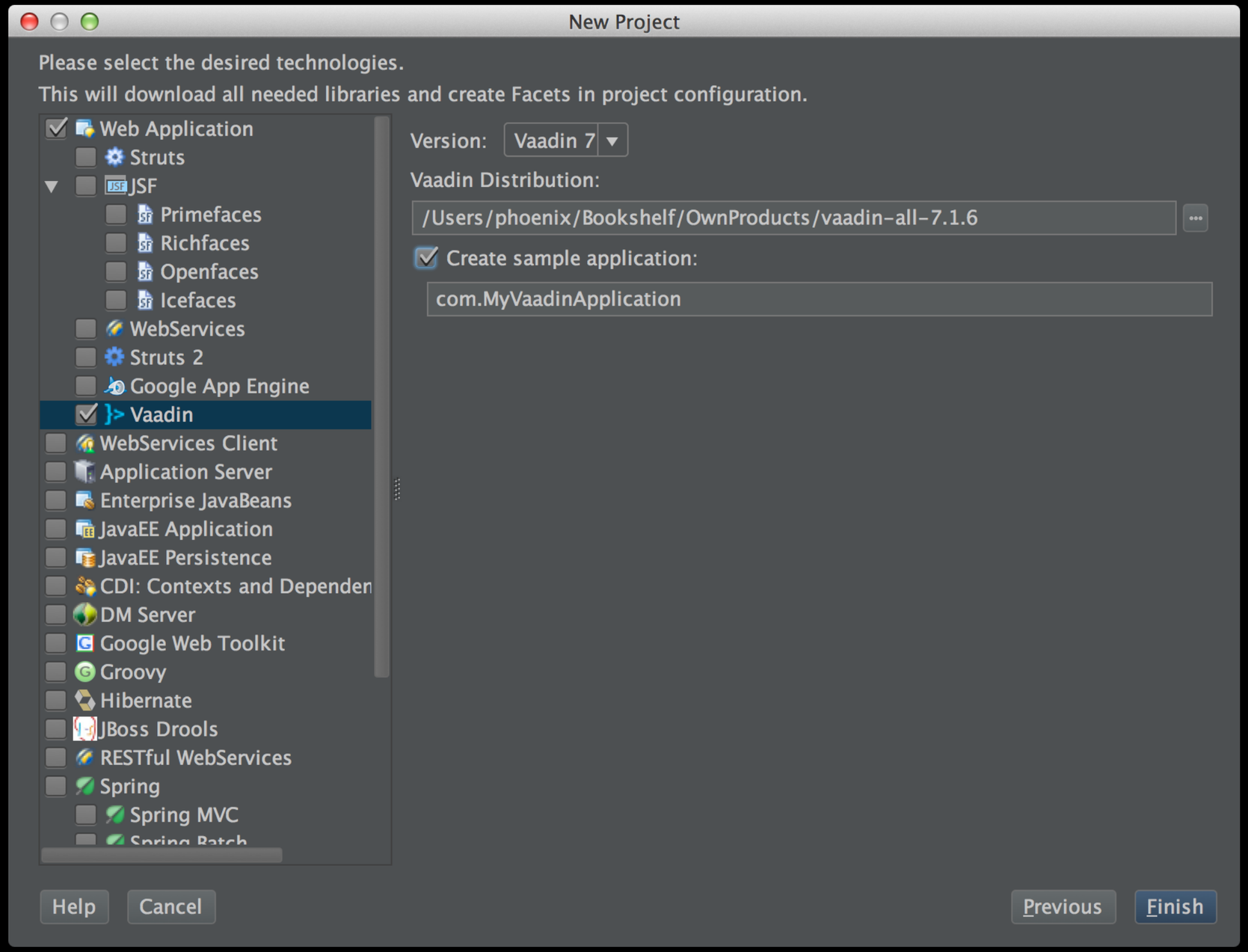
Eclipse



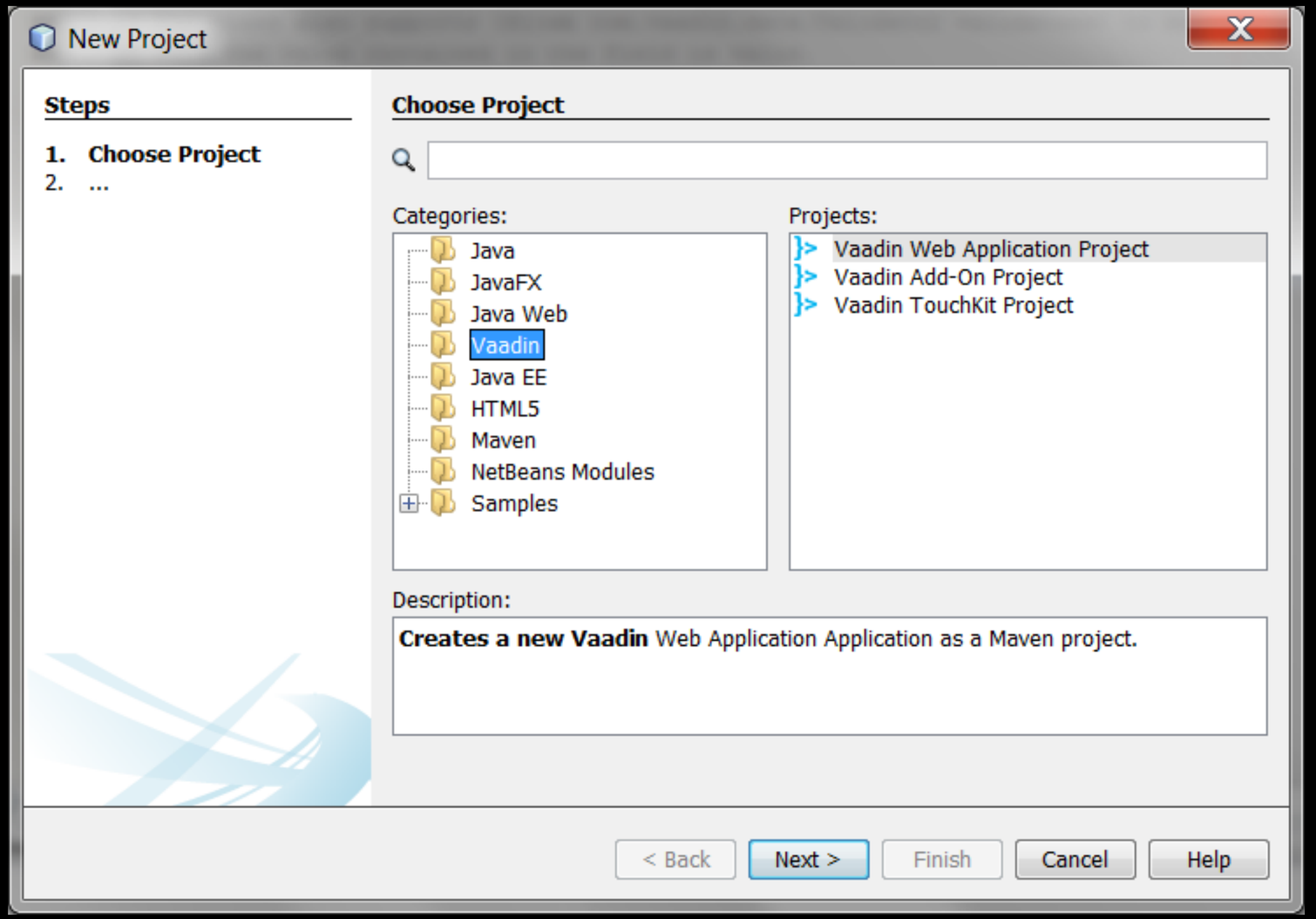
Download plugin
from Martketplace

IntelliJ IDEA

Built-in support



Netbeans



Download plugin
Netbeans Plugin Portal

JavaCro

Example App

github.com/peterl1084/cdiexample

Peter Lehto
@peter_lehto
expert & trainer
vaadin

What did we learn today?

1. Java EE contains tons of specifications for enterprise apps
2. Vaadin is a great way to assemble UIs with components
3. Combining Vaadin with Java EE works best through CDI
4. Write maintainable code by following best practices

JavaCro

Thank you!

Peter Lehto
@peter_lehto
expert & trainer
vaadin