Introduction to Internet Applications

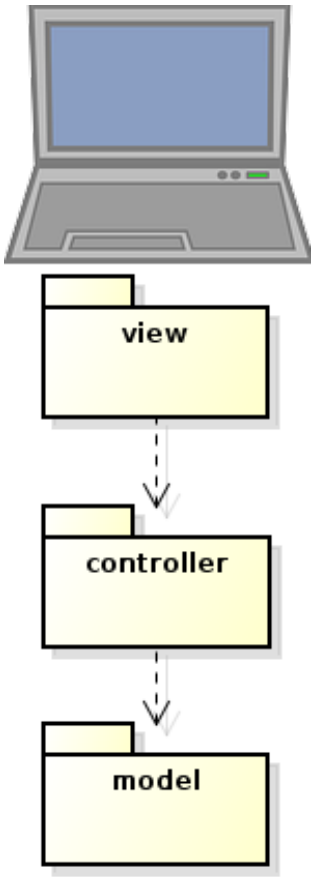
Internet Applications, ID1354

Contents

Section

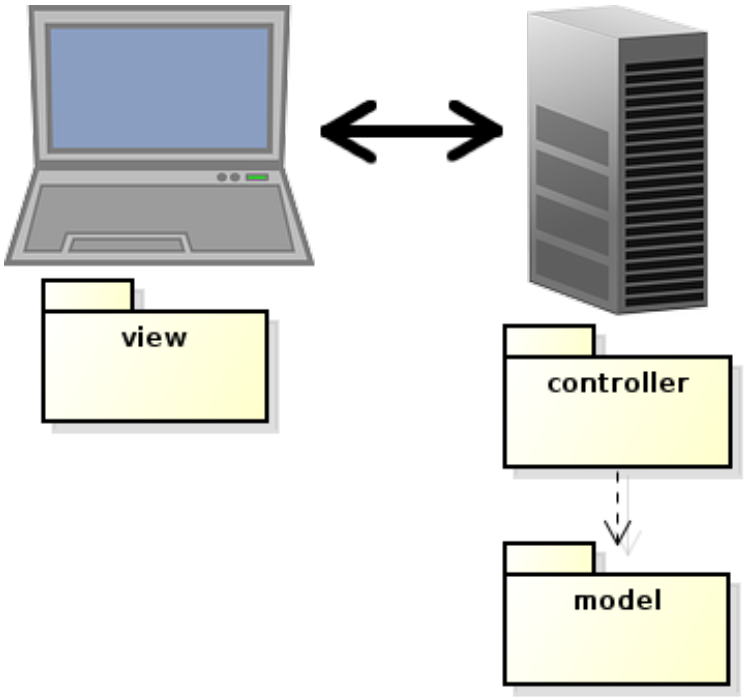
- Distributed Architectures
- User Interface Design
- Tools

Local Application



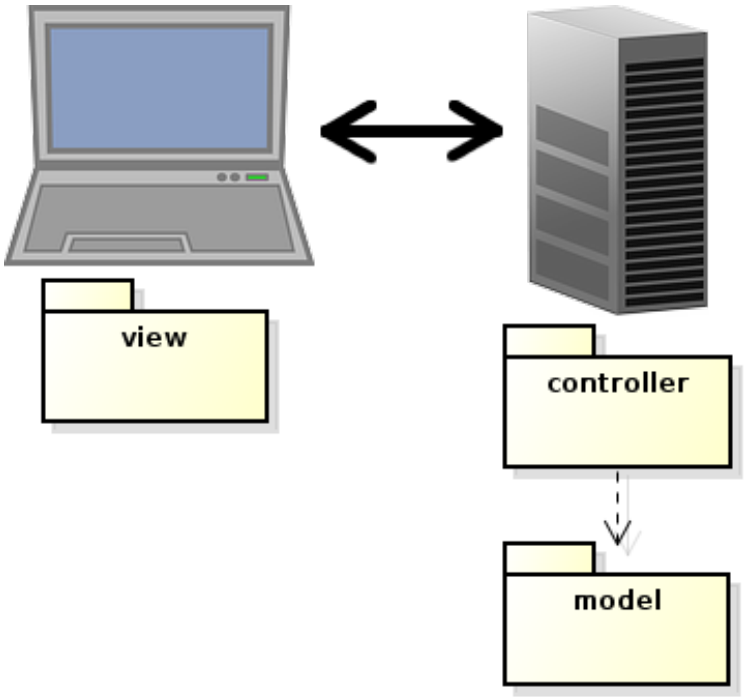
- We are familiar with an architecture where the entire application resides on the same computer.

Introducing a Server



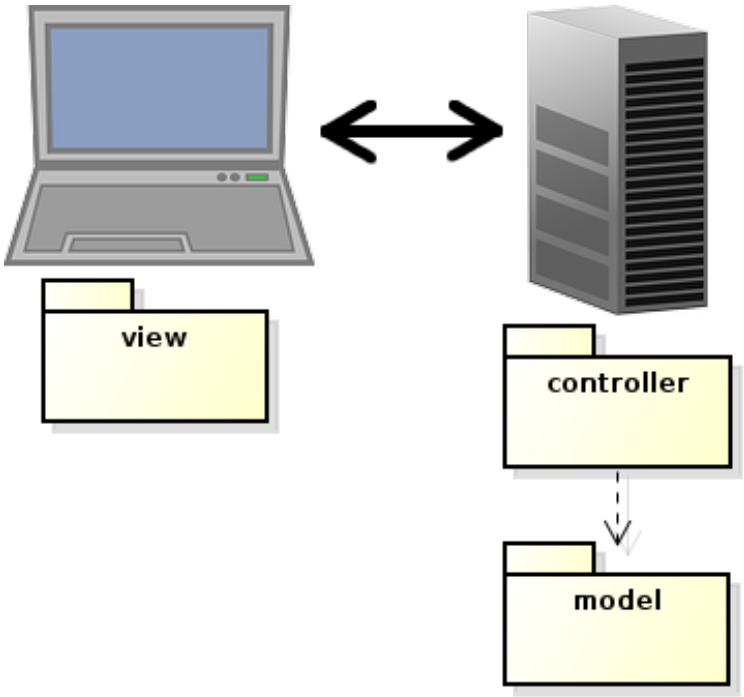
- Now, the application will be split on two tiers (computers).

Introducing a Server



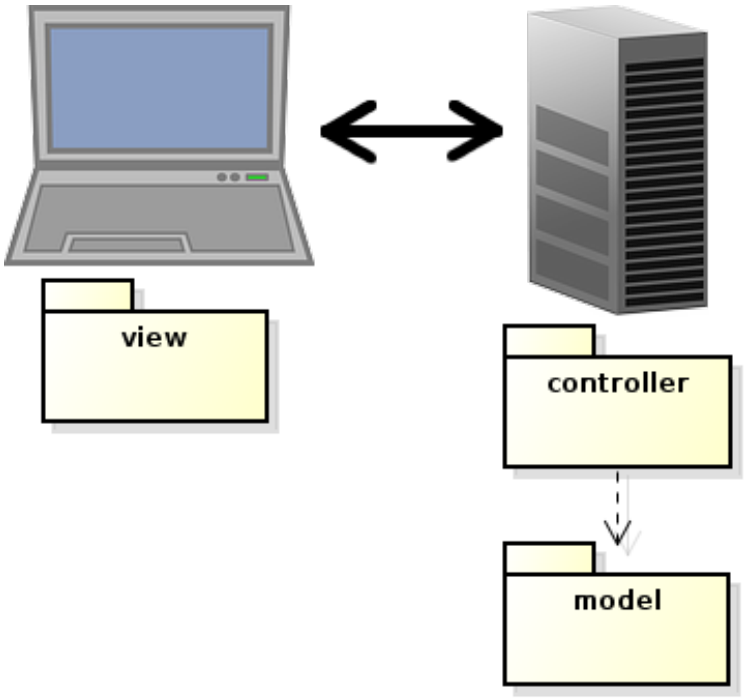
- Now, the application will be split on two tiers (computers).
- A client that has the view and a server that has controller and model.

Introducing a Server



- Now, the application will be split on two tiers (computers).
- A client that has the view and a server that has controller and model.
- The view is displayed in a web browser.

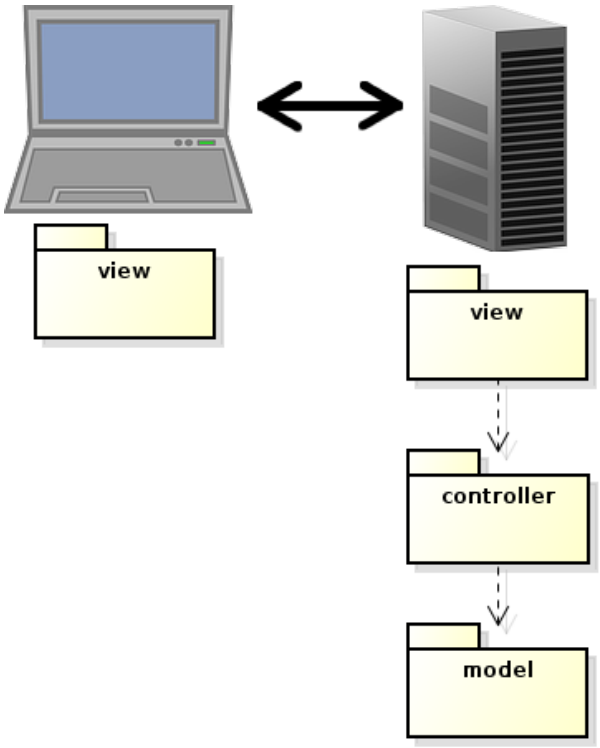
Introducing a Server



- Now, the application will be split on two tiers (computers).
- A client that has the view and a server that has controller and model.
- The view is displayed in a web browser.

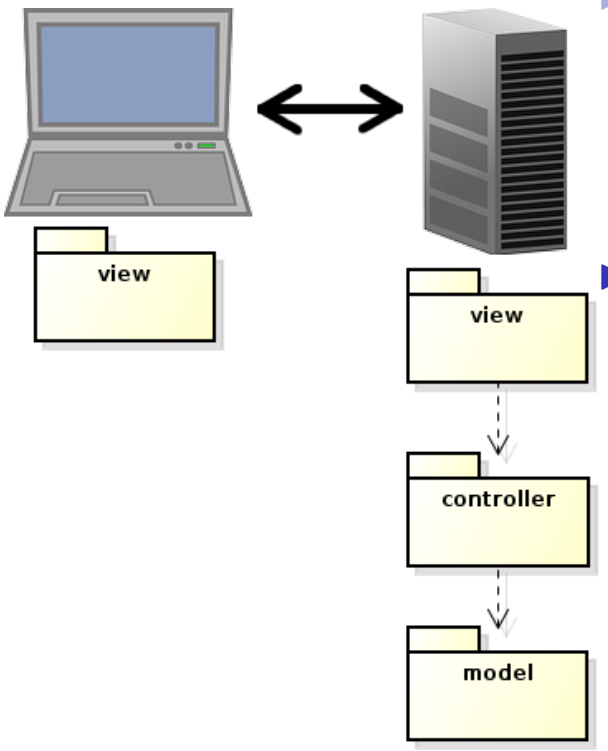
This architecture is not good, we also need layers for communication.

Server-Side Communication



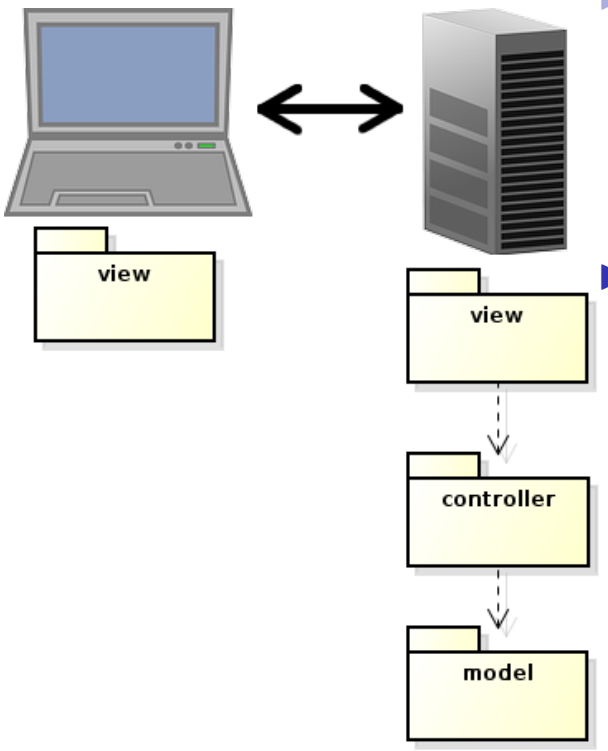
- First, we add a server layer, normally called view (a bit confusing).

Server-Side Communication



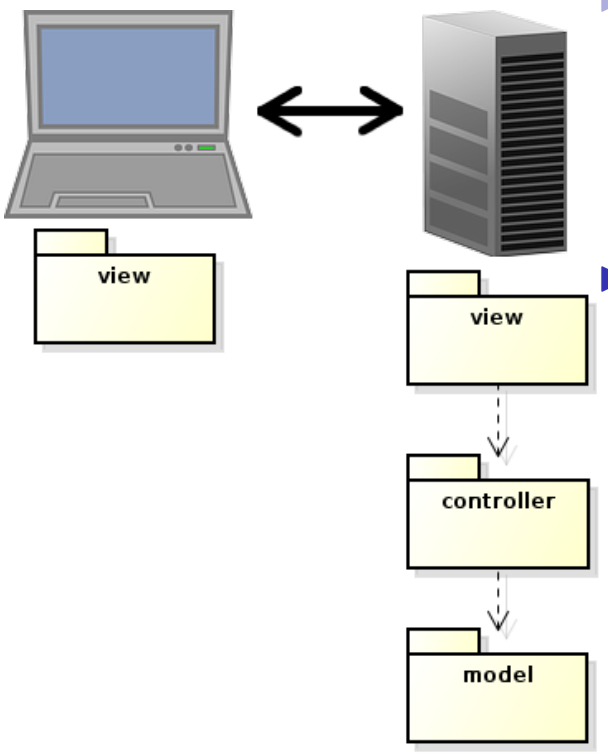
- First, we add a server layer, normally called view (a bit confusing).
- However, the server side view layer performs tasks typical of a view:

Server-Side Communication



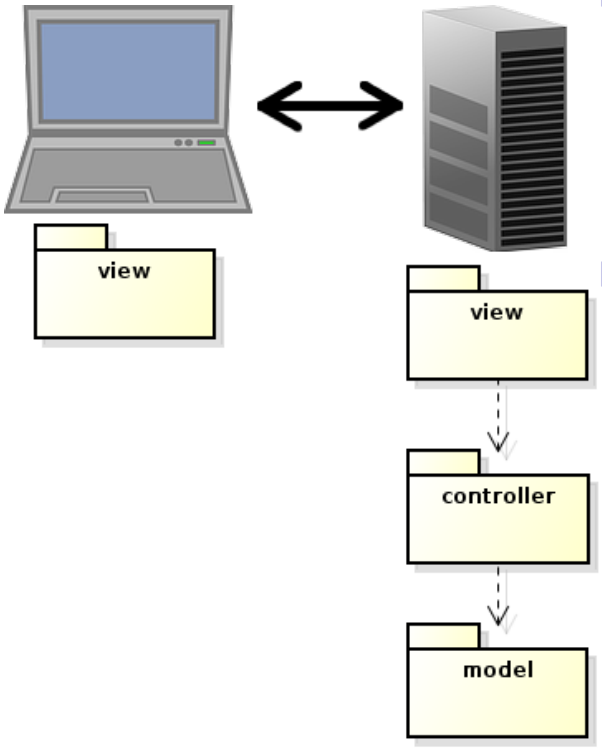
- First, we add a server layer, normally called view (a bit confusing).
- However, the server side view layer performs tasks typical of a view:
  - Creates views (HTML), which are sent to the client.

Server-Side Communication



- First, we add a server layer, normally called view (a bit confusing).
- However, the server side view layer performs tasks typical of a view:
  - Creates views (HTML), which are sent to the client.
  - Interprets user gestures, a click in a web page creates a request to the server.

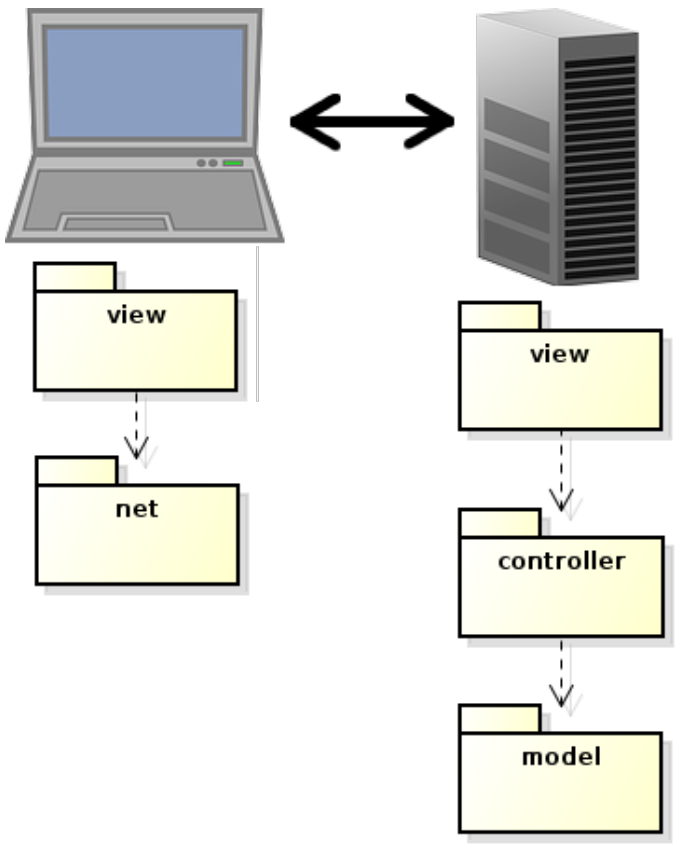
Server-Side Communication



- First, we add a server layer, normally called view (a bit confusing).
- However, the server side view layer performs tasks typical of a view:
  - Creates views (HTML), which are sent to the client.
  - Interprets user gestures, a click in a web page creates a request to the server.

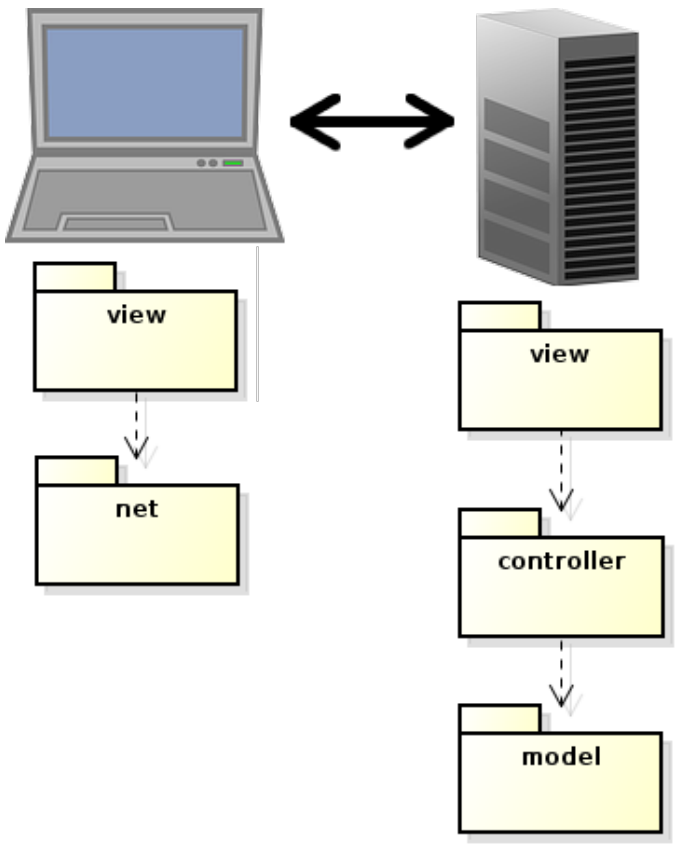
It might seem that we need yet a layer, for network handling. There is such a layer, but it is in the web server. We don't write it ourselves.

Client-Side Communication



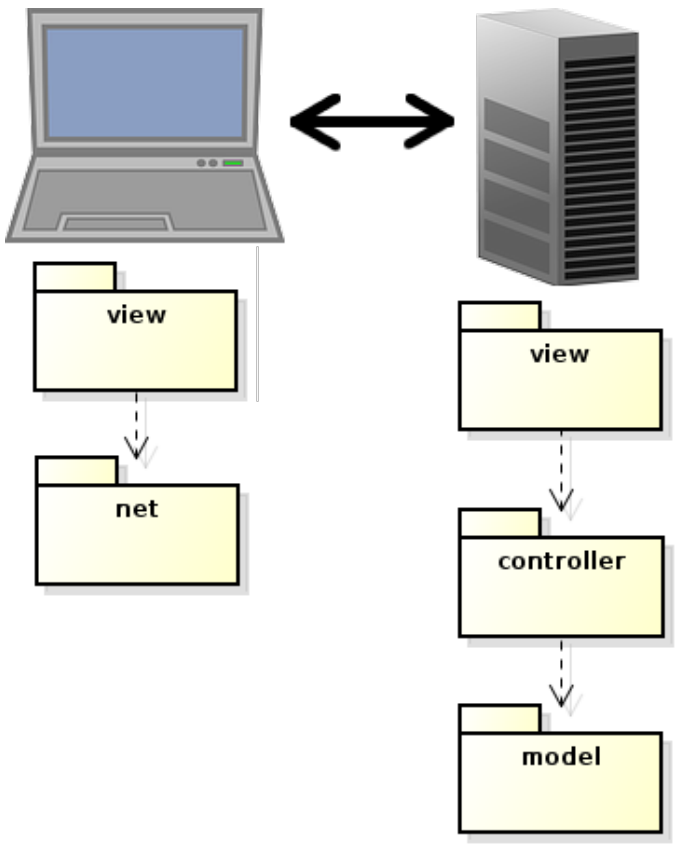
- Next, we add a client layer for communication, the net layer.

Client-Side Communication



- Next, we add a client layer for communication, the net layer.
- Actually, the browser handles most of the communication.
  - The small network code written by us is normally considered part of the client-side view, the net layer is omitted.

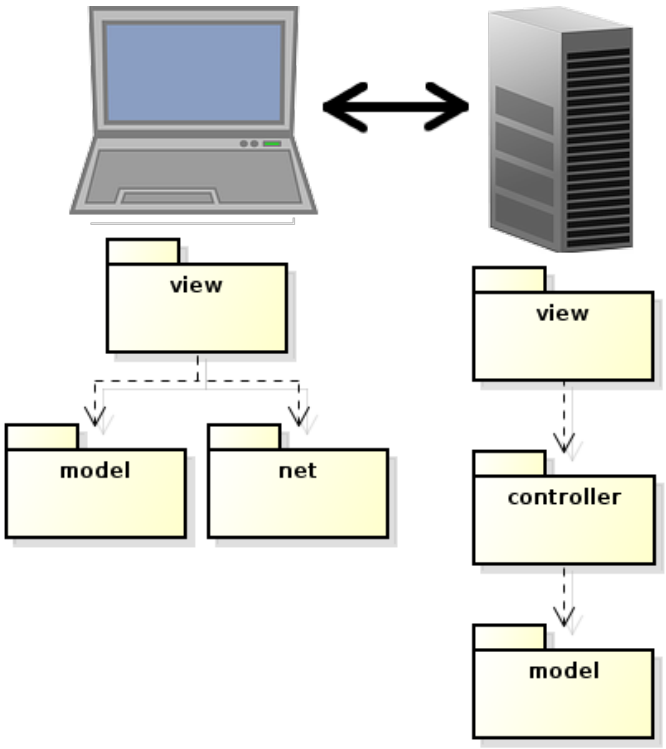
Client-Side Communication



- Next, we add a client layer for communication, the net layer.
- Actually, the browser handles most of the communication.
  - The small network code written by us is normally considered part of the client-side view, the net layer is omitted.

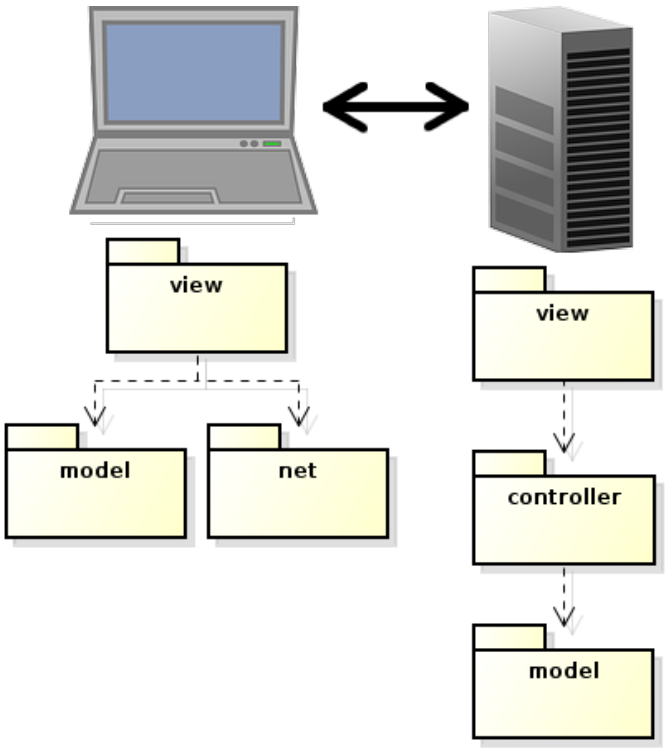
- This is a traditional web application.

The MVVM Pattern



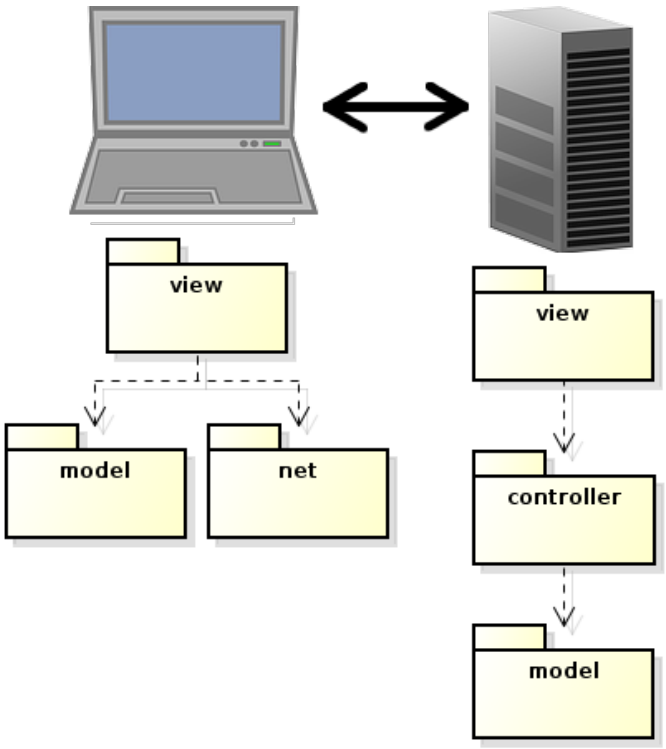
- The trend is that data is stored also on the client, therefore we get a client-side model.

The MVVM Pattern



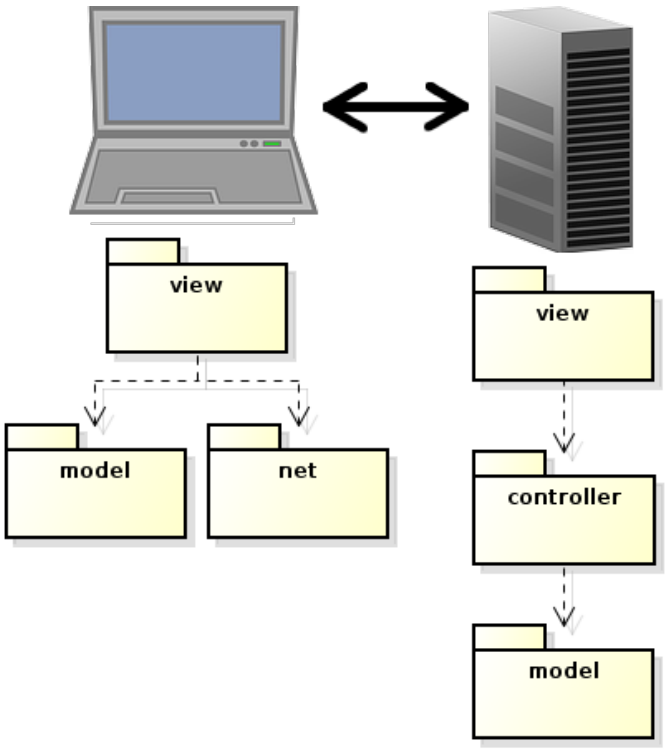
- The trend is that data is stored also on the client, therefore we get a client-side model.
- This reduces the network communication, since we do not need to resend the entire view each time the user does something.

The MVVM Pattern



- The trend is that data is stored also on the client, therefore we get a client-side model.
- This reduces the network communication, since we do not need to resend the entire view each time the user does something.
- Thereby, the application becomes faster.

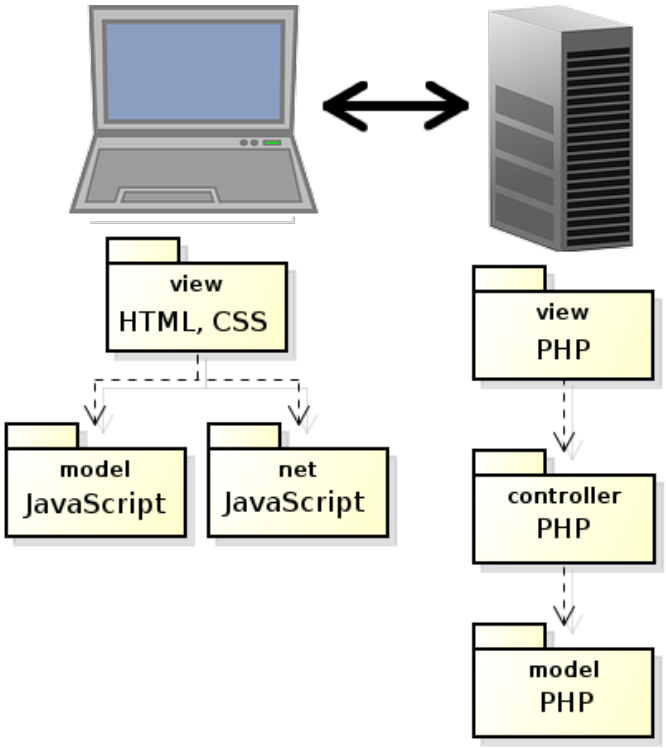
The MVVM Pattern



- The trend is that data is stored also on the client, therefore we get a client-side model.
- This reduces the network communication, since we do not need to resend the entire view each time the user does something.
- Thereby, the application becomes faster.

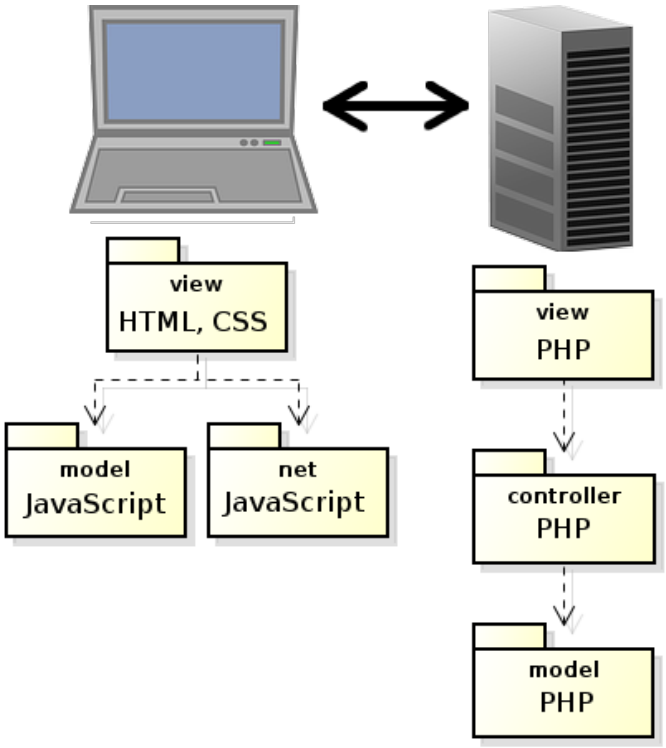
- This is referred to as the MVVM, model-view-viewmodel pattern.

Programming Languages



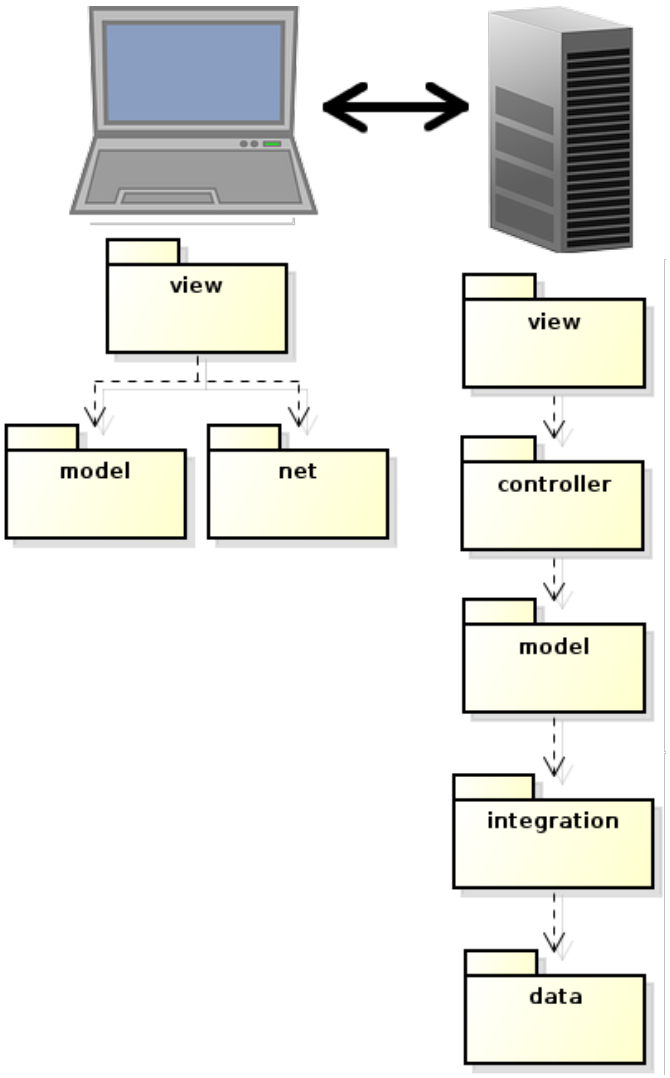
- This is the architecture we will normally use during the course.

Programming Languages



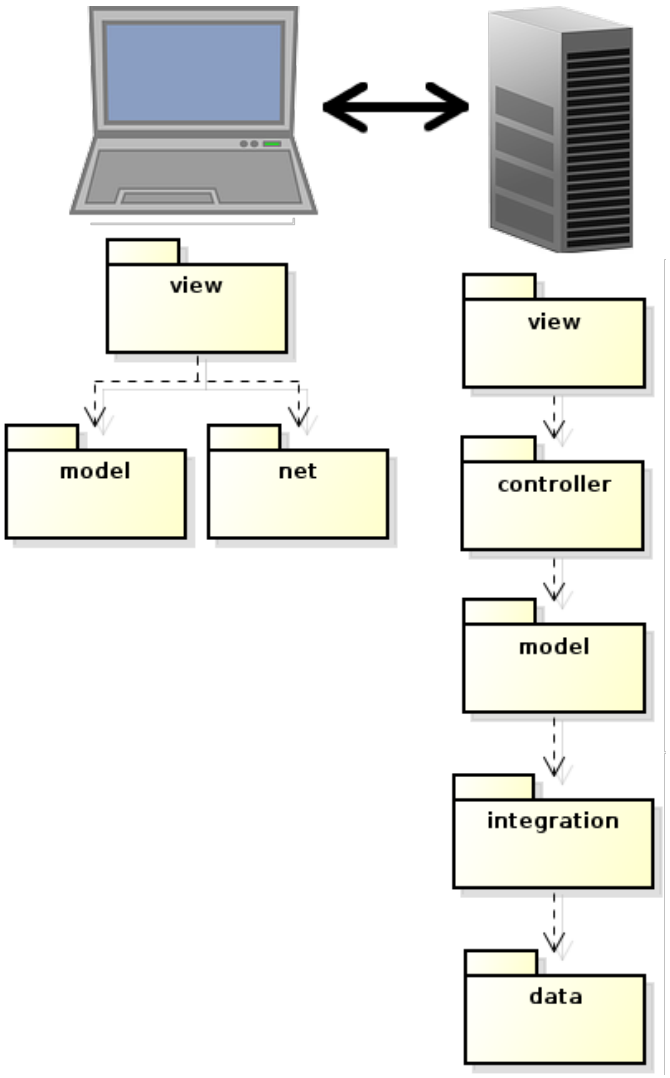
- This is the architecture we will normally use during the course.
- The view is programmed in HTML and CSS, client side behavior is programmed in JavaScript and the entire server side code is written in PHP.

Three-Tier Architecture



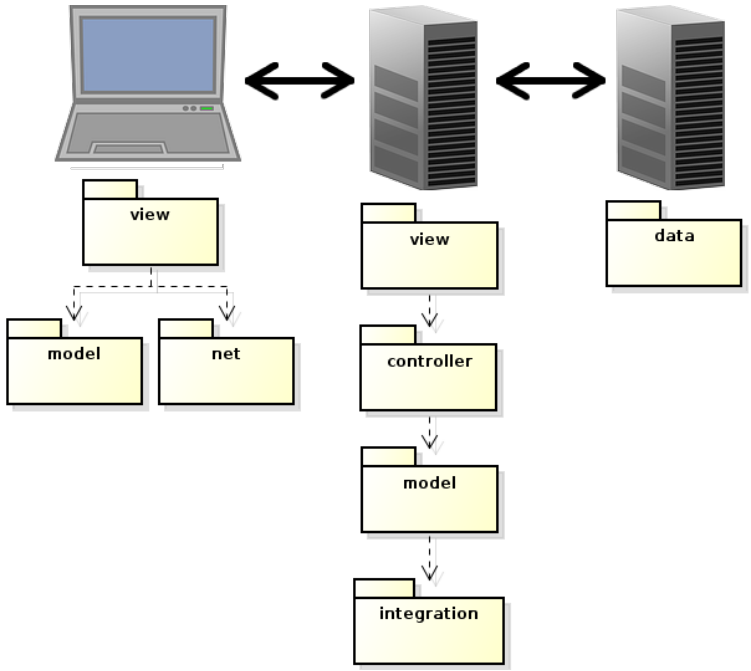
- Of course, we also need to store data. That is done in the data layer, which is often a database.

Three-Tier Architecture



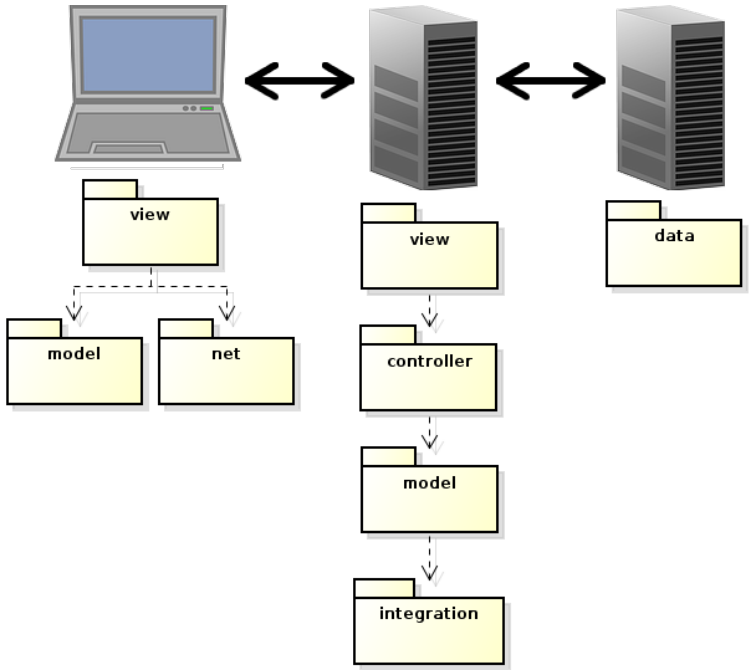
- Of course, we also need to store data. That is done in the data layer, which is often a database.
- We also introduce the integration layer, to handle the database calls.

Three-Tier Architecture (Cont'd)



- In a bigger application, we would most likely place the database in a separate node.

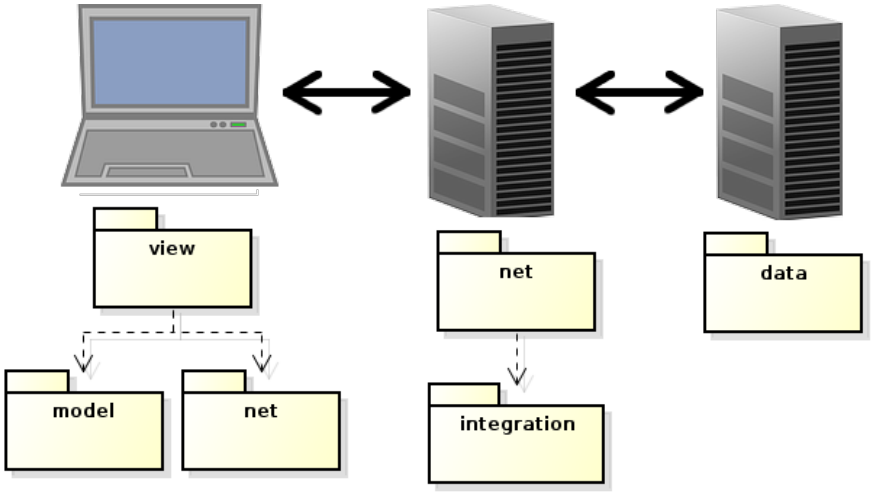
Three-Tier Architecture (Cont'd)



- In a bigger application, we would most likely place the database in a separate node.
- This is called three-tier architecture and is, since long time, the dominating architecture for web applications.

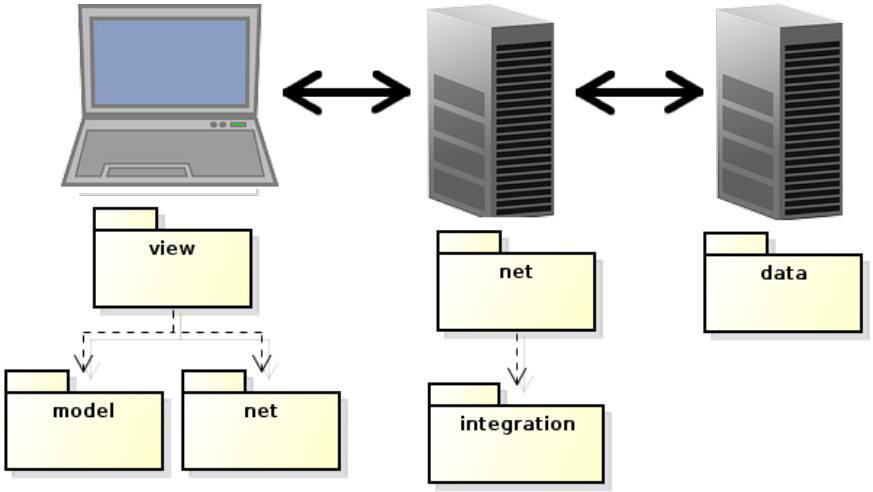
Question 1

Event-Driven Architecture



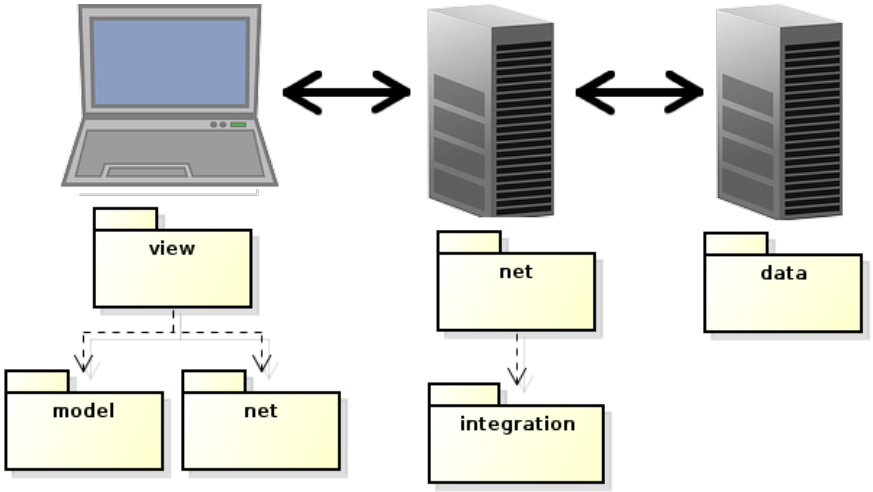
- In the latest year, there is a growing tendency to move business logic to the client, perhaps completely remove the server-side model.

Event-Driven Architecture



- In the latest year, there is a growing tendency to move business logic to the client, perhaps completely remove the server-side model.
- This is made possible with web sockets, which enable full duplex browser-server communication.

Event-Driven Architecture



- In the latest year, there is a growing tendency to move business logic to the client, perhaps completely remove the server-side model.
- This is made possible with web sockets, which enable full duplex browser-server communication.
- The motive is to reduce communication latency. The browser informs the server about user actions, but does not wait for response before updating the view.

Section

- Distributed Architectures
- User Interface Design
- Tools

Use UI Guidelines!

- This is not a course in human-computer interaction. Still, it is mandatory to consider basic heuristics for user interface design.

Use UI Guidelines!

- This is not a course in human-computer interaction. Still, it is mandatory to consider basic heuristics for user interface design.
- There are some short introductory texts on user interface design available at Nielsen Norman Group, such as:

Use UI Guidelines!

- This is not a course in human-computer interaction. Still, it is mandatory to consider basic heuristics for user interface design.
- There are some short introductory texts on user interface design available at Nielsen Norman Group, such as:
  - 10 Usability Heuristics for UI Design, **http://www.nngroup.com/articles/ten-usability-heuristics/**

Use UI Guidelines!

- This is not a course in human-computer interaction. Still, it is mandatory to consider basic heuristics for user interface design.
- There are some short introductory texts on user interface design available at Nielsen Norman Group, such as:
  - 10 Usability Heuristics for UI Design, **http://www.nngroup.com/articles/ten-usability-heuristics/**
  - Top 10 Guidelines for Homepage Usability, **http://www.nngroup.com/articles/top-ten-guidelines-for-homepage-usability/**

Use UI Guidelines!

- This is not a course in human-computer interaction. Still, it is mandatory to consider basic heuristics for user interface design.
- There are some short introductory texts on user interface design available at Nielsen Norman Group, such as:
  - 10 Usability Heuristics for UI Design, **http://www.nngroup.com/articles/ten-usability-heuristics/**
  - Top 10 Guidelines for Homepage Usability, **http://www.nngroup.com/articles/top-ten-guidelines-for-homepage-usability/**
  - Top 10 Mistakes in Web Design, **http://www.nngroup.com/articles/top-10-mistakes-web-design/**

1. Visibility of system status

J. Nielsen's 10 Usability Heuristics

- The system should always keep users informed about what is going on, through appropriate feedback.

1. Visibility of system status

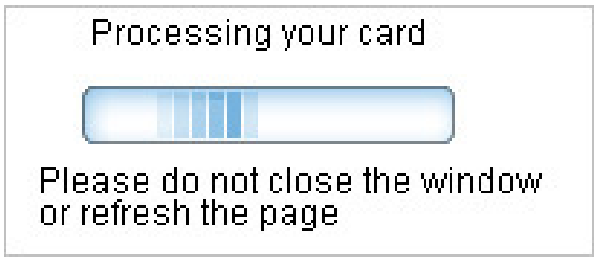
J. Nielsen's 10 Usability Heuristics

- The system should always keep users informed about what is going on, through appropriate feedback.
- The UI must change within one second after a user action, or the user might think nothing happened.

1. Visibility of system status

J. Nielsen's 10 Usability Heuristics

- The system should always keep users informed about what is going on, through appropriate feedback.
- The UI must change within one second after a user action, or the user might think nothing happened.
- Good examples:



2. Match between system and the real world

J. Nielsen's 10 Usability Heuristics

- Use words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

2. Match between system and the real world

J. Nielsen's 10 Usability Heuristics

- Use words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- Good example ("How can we help you?" better than "FAQ"):

2. Match between system and the real world

J. Nielsen's 10 Usability Heuristics

- Use words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- Good example ("How can we help you?" better than "FAQ"):
- Bad example ("Continue if enabled" is system oriented language):

3. User control and freedom

J. Nielsen's 10 Usability Heuristics

- We often do things by mistake, and therefore need a clearly marked "emergency exit" to leave an unwanted state without having to go through an extended dialogue. Support undo and redo.

3. User control and freedom

J. Nielsen's 10 Usability Heuristics

- We often do things by mistake, and therefore need a clearly marked "emergency exit" to leave an unwanted state without having to go through an extended dialogue. Support undo and redo.
- Good examples:

4. Consistency and standards

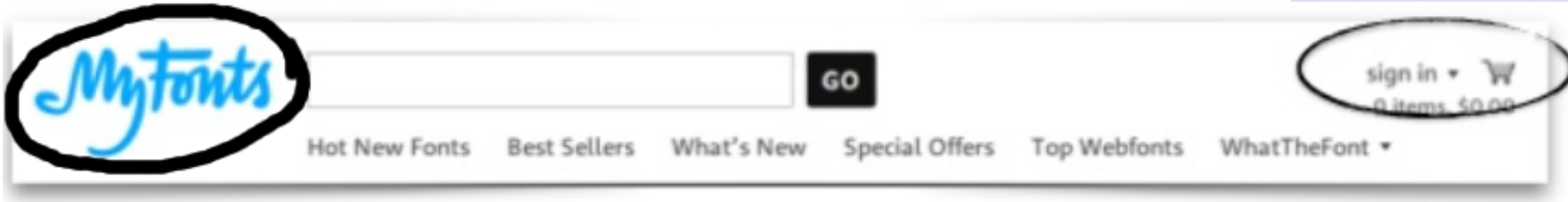
J. Nielsen's 10 Usability Heuristics

- Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

4. Consistency and standards

J. Nielsen's 10 Usability Heuristics

- Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- Good example (Sign in at top right, logo with link to index page at top left):



5. Error prevention

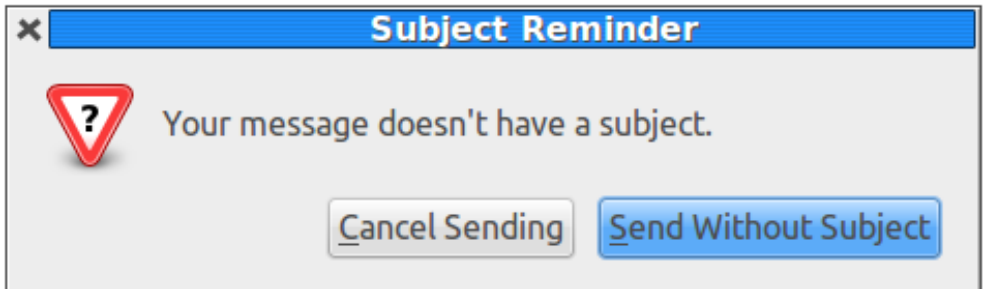
J. Nielsen's 10 Usability Heuristics

- Create a careful design which prevents problems from occurring. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

5. Error prevention

J. Nielsen's 10 Usability Heuristics

- Create a careful design which prevents problems from occurring. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
- Good example:



6. Recognition rather than recall

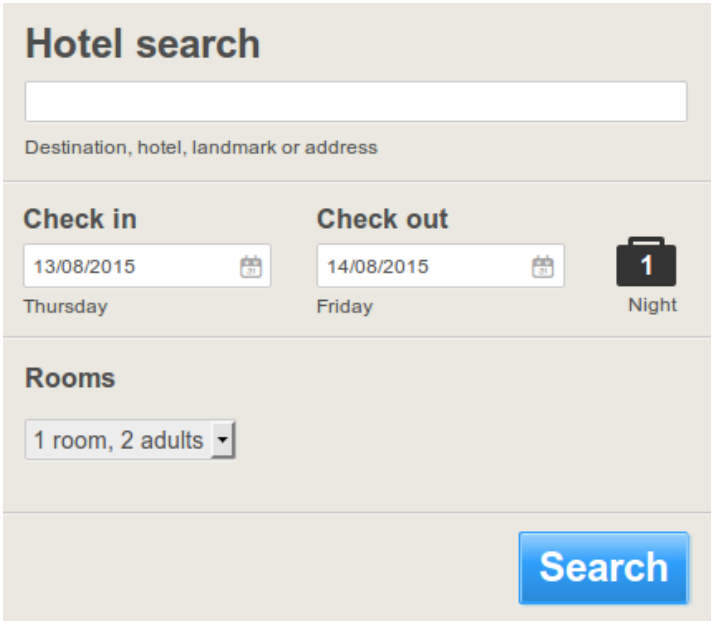
J. Nielsen's 10 Usability Heuristics

- Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

6. Recognition rather than recall

J. Nielsen's 10 Usability Heuristics

- Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- Good example:



- Clear headline.
- No doubt where to click to start the search.

7. Flexibility and efficiency of use

J. Nielsen's 10 Usability Heuristics

- Accelerators, unseen by the novice user, may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

7. Flexibility and efficiency of use

J. Nielsen's 10 Usability Heuristics

- Accelerators, unseen by the novice user, may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- Examples are
  - *Saved searches*
  - *Items you recently looked at*
  - *Save query for later*

8. Aesthetic and minimalist design

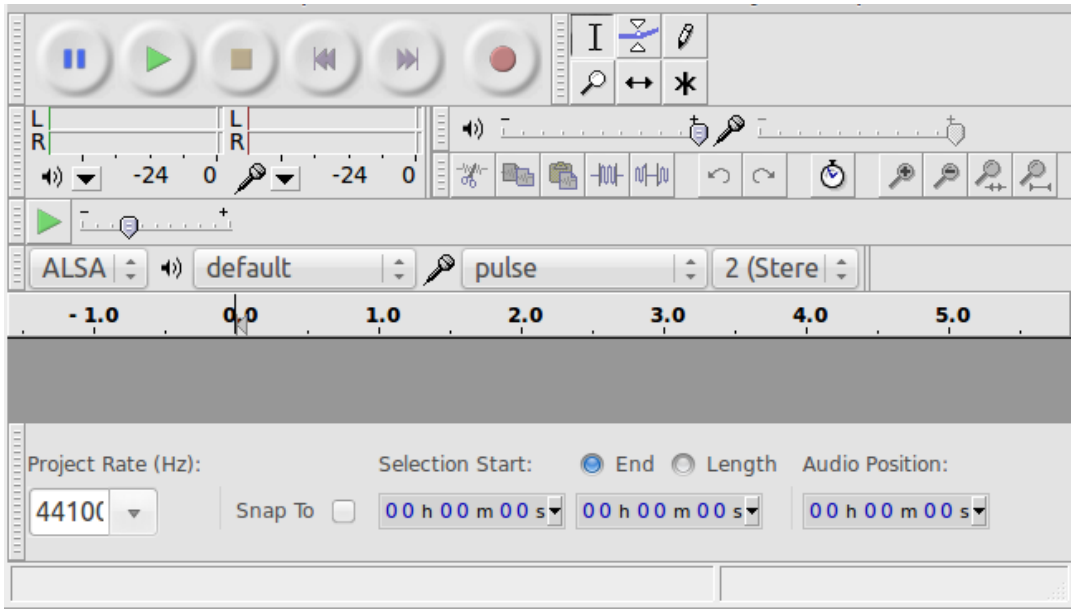
J. Nielsen's 10 Usability Heuristics

- Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their visibility.

8. Aesthetic and minimalist design

J. Nielsen's 10 Usability Heuristics

- Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their visibility.
- Bad example:



9. Help users recognize, diagnose, and recover from errors

J. Nielsen's 10 Usability Heuristics

- Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

9. Help users recognize, diagnose, and recover from errors

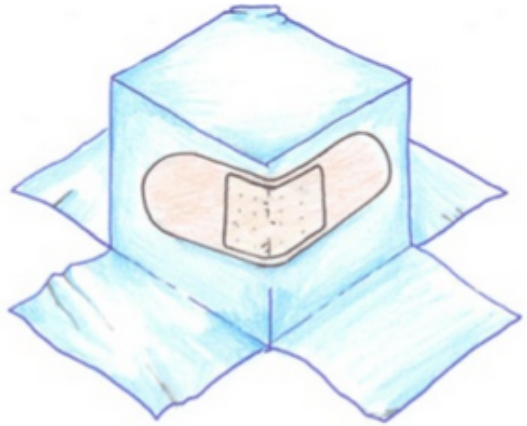
J. Nielsen's 10 Usability Heuristics

- Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- Do not tell the user *unexpected exception* or anything similar.

9. Help users recognize, diagnose, and recover from errors

J. Nielsen's 10 Usability Heuristics

- Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- Do not tell the user *unexpected exception* or anything similar.
- Good examples:



Error

Something went wrong. Don't worry, your files are still safe and the Dropboxers have been notified. Check out our Help Center and forums for help, or head back to home.

10. Help and documentation

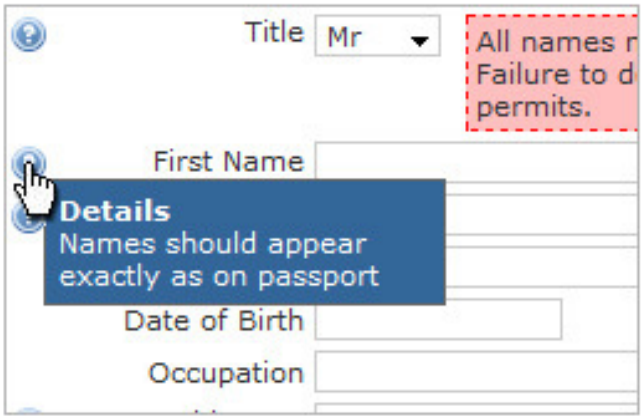
J. Nielsen's 10 Usability Heuristics

- Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

10. Help and documentation

J. Nielsen's 10 Usability Heuristics

- Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.
- Good example:



Question 2

Section

- Distributed Architectures
- User Interface Design
- Tools

Web Development Tools

- There are many tools that facilitates developing web applications.

Web Development Tools

- There are many tools that facilitates developing web applications.
- Browser support varies between tools, most examples will be using Firefox.

Web Development Tools

- There are many tools that facilitates developing web applications.
- Browser support varies between tools, most examples will be using Firefox.
- You are strongly advised to start using some of the following tools, they will help you a lot.

Browser Web Console

- Most browsers have a built-in console.

Browser Web Console

- Most browsers have a built-in console.
- The console logs information associated with the web page, for example errors and warnings related to JavaScript, CSS and network requests.

Browser Web Console

- Most browsers have a built-in console.
- The console logs information associated with the web page, for example errors and warnings related to JavaScript, CSS and network requests.
- It enables you to run JavaScript expressions in the web page.

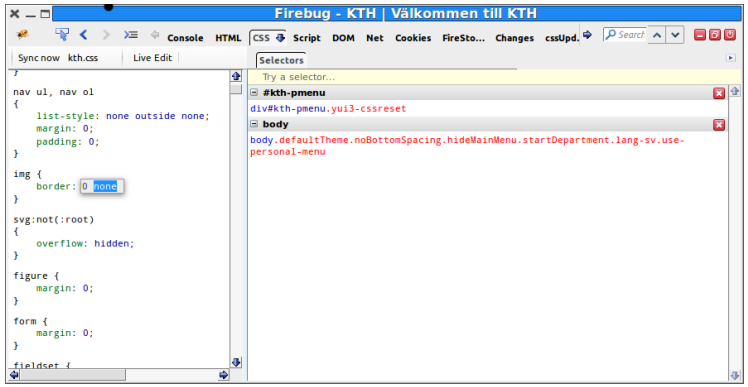
Browser Web Console

- Most browsers have a built-in console.
- The console logs information associated with the web page, for example errors and warnings related to JavaScript, CSS and network requests.
- It enables you to run JavaScript expressions in the web page.
- It also lets you choose elements from the web page and have their HTML and CSS displayed.

Browser Web Console (Cont'd)

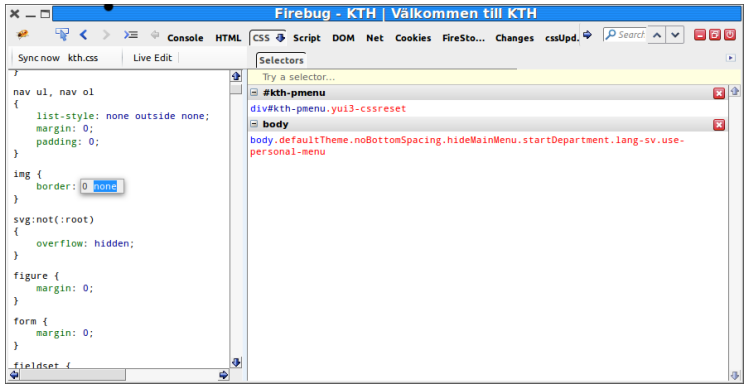
- The console is opened with **`Ctrl-Shift-K`** in Firefox and **`Ctrl-Shift-J`** in Chrome.

Firebug



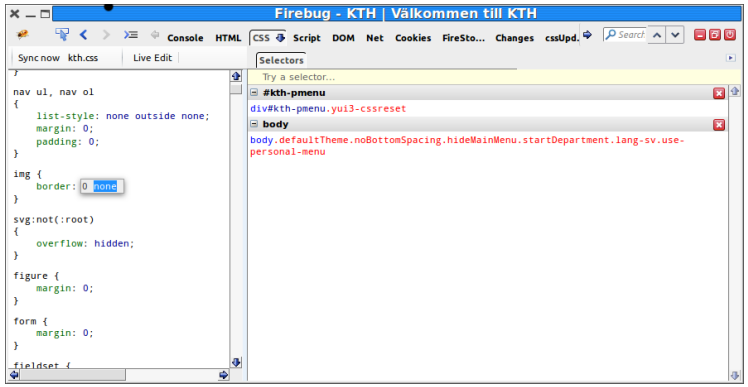
- Firebug is a powerful plug-in to Firefox.

Firebug



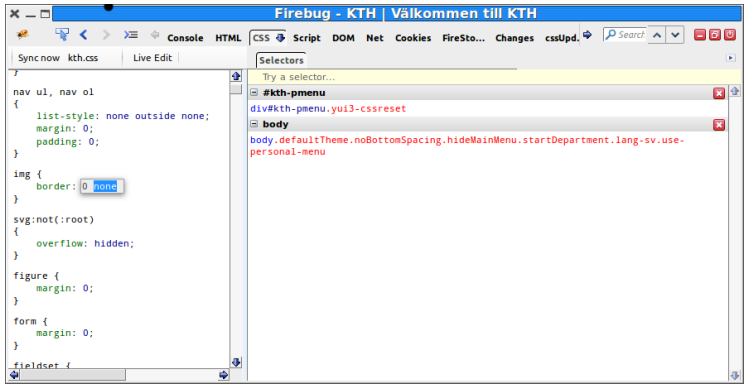
- Firebug is a powerful plug-in to Firefox.
- In addition to console features, you can for example debug JavaScript, mark HTML elements, edit CSS and log network traffic.

Firebug



- Firebug is a powerful plug-in to Firefox.
- In addition to console features, you can for example debug JavaScript, mark HTML elements, edit CSS and log network traffic.
- There are also many plug-ins to Firebug.

Firebug



- Firebug is a powerful plug-in to Firefox.
- In addition to console features, you can for example debug JavaScript, mark HTML elements, edit CSS and log network traffic.
- There are also many plug-ins to Firebug.
- There is a cross-browser version of Firebug, written in JavaScript, that offers a subset of the functionality for most other browsers.

Web Developer

- Web Developer is a powerful plug-in to Firefox, which allows you to:

Web Developer

- Web Developer is a powerful plug-in to Firefox, which allows you to:
  - edit HTML and CSS.

Web Developer

- Web Developer is a powerful plug-in to Firefox, which allows you to:
  - edit HTML and CSS.
  - See the area covered by a chosen element.

Web Developer

- Web Developer is a powerful plug-in to Firefox, which allows you to:
  - edit HTML and CSS.
  - See the area covered by a chosen element.
  - See the page in different screen resolutions.

Web Developer

- Web Developer is a powerful plug-in to Firefox, which allows you to:
  - edit HTML and CSS.
  - See the area covered by a chosen element.
  - See the page in different screen resolutions.
  - Edit cookies.

Web Developer

- Web Developer is a powerful plug-in to Firefox, which allows you to:
  - edit HTML and CSS.
  - See the area covered by a chosen element.
  - See the page in different screen resolutions.
  - Edit cookies.
  - Validate HTML and CSS.

Web Developer

- Web Developer is a powerful plug-in to Firefox, which allows you to:
  - edit HTML and CSS.
  - See the area covered by a chosen element.
  - See the page in different screen resolutions.
  - Edit cookies.
  - Validate HTML and CSS.
- Web Developer has been ported to Chrome.

Validators

- There are online validators for both HTML and CSS. Links can be found on the course web site.

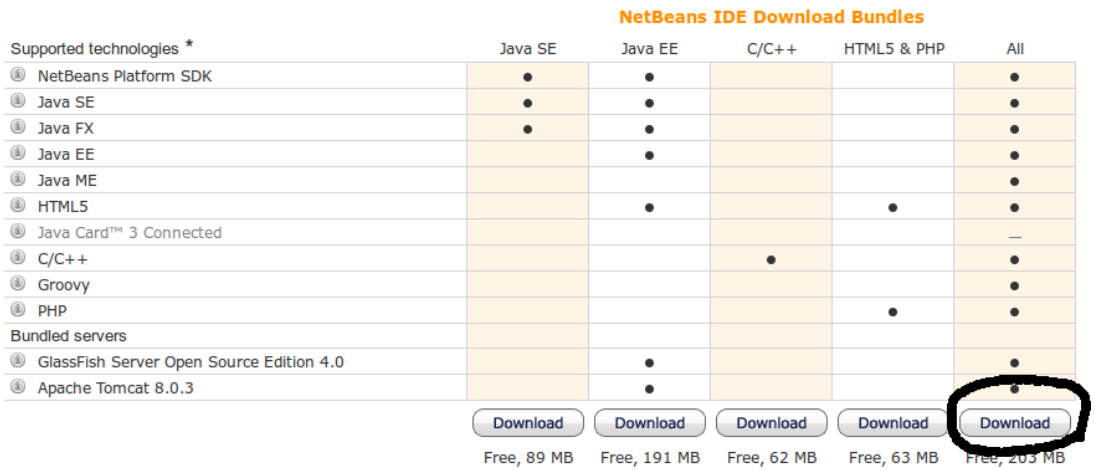
Validators

- There are online validators for both HTML and CSS. Links can be found on the course web site.
- Remember to always validate your HTML and CSS code.

NetBeans

- There are many different IDEs for web development, all have their pros and cons.

NetBeans



NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	C/C++	HTML5 & PHP	All
NetBeans Platform SDK	•	•			•
Java SE	•	•			•
Java FX	•	•			•
Java EE		•			•
Java ME					•
HTML5		•		•	•
Java Card™ 3 Connected					–
C/C++			•		•
Groovy					•
PHP				•	•
Bundled servers					
GlassFish Server Open Source Edition 4.0		•			•
Apache Tomcat 8.0.3		•			•
	Download	Download	Download	Download	Download
	Free, 89 MB	Free, 191 MB	Free, 62 MB	Free, 63 MB	Free, 203 MB

- There are many different IDEs for web development, all have their pros and cons.
- NetBeans will be used for examples during the course. Make sure to download the All version, see image above.

NetBeans

- There are many different IDEs for web development, all have their pros and cons.
- NetBeans will be used for examples during the course. Make sure to download the All version, see image above.
- Most important is that you actually use an IDE, do not program in a text editor unless you are really sure it is what you prefer.

JSFiddle and JSLint

- JSFiddle is an online editor where you can test HTML, CSS and JavaScript.

JSFiddle and JSLint

- JSFiddle is an online editor where you can test HTML, CSS and JavaScript.
- JSLint is an online tool for testing JavaScript code quality.

W3Schools Try It Yourself

- **`w3schools.com`** has excellent tutorials for all languages covered in the course.

W3Schools Try It Yourself

- `w3schools.com` has excellent tutorials for all languages covered in the course.
- All examples are presented with an online editor where you can experiment with your code.