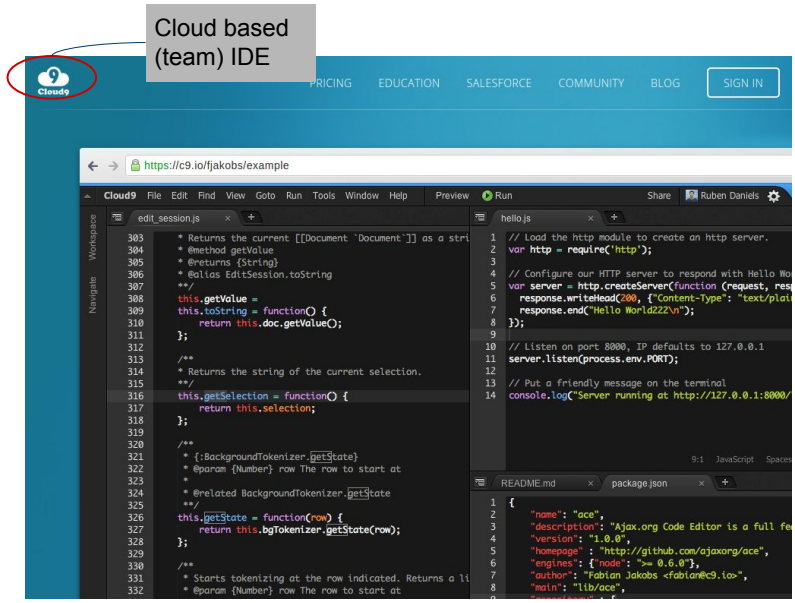
Web Applications 2017

Week 4, Slides 2

Content

- Realtime web
- Polling
- WebSockets
- Language Localization
- Conversion
- Validation
- Exception handling

Real time Web



Readings

- [10 uses for real time web]
- [Status of real time web 2016]

Polling

Client side script

```
timerId: 0,
poll: function() {

            var delay = 1000 * parseInt($("#delay").val());
            console.log(delay);
            var self = this;
            this.timerId = setTimeout(function() {
                $.get("meter.php", {action: "poll"}).fail(function() {
                    console.log("server down");
                    self.stop();
                }).done(function(result) {
                    console.log(result);
                    var json = JSON.parse(result);
                    if (json.started === true) {
                        self.display(json.requests / json.nstudents);
                        self.dec();
                    }
                    self.timerId = setTimeout(self.poll.bind(self), delay);
                })
            }, delay);
        },
```

Readings

- Polling

Web Sockets Client API

```
var chatClient = new WebSocket("ws://localhost:8080/websocket/chat");

chatClient.onmessage = function (evt) {
    var p = document.createElement("p");
    p.setAttribute("class", "server");
    p.innerHTML = "Server: " + evt.data;
    var container = document.getElementById("container");
    container.appendChild(p);
}

function send() {
    var input = document.getElementById("message");
    var p = document.createElement("p");
    p.setAttribute("class", "client");
    p.innerHTML = "Me: " + input.value;
    var container = document.getElementById("container");
    container.appendChild(p);
    chatClient.send(input.value);
    input.value = "";
}
```

In browser

JEE WebSockets

```
@ServerEndpoint(value = "/chat")
public class ChatServer {

    private static final Logger LOG
            = Logger.getLogger(ChatServer.class.getName());

    @OnOpen
    public void onOpen(Session session) {
        LOG.log(Level.INFO, "New connection with client: {0}",
                session.getId());
    }

    @OnMessage
    public String onMessage(String message, Session session) {
        LOG.log(Level.INFO, "New message from Client [{0}]: {1}",
                new Object[]{session.getId(), message});
        return "Server received [" + message + "]";

    ...
}
```

Run on Tomcat or other

Readings

- Node WebSockets
- JEE WebSockets

Node WebSockets

```
const WebSocket = require('ws');
const wss = new WebSocket.Server({port: 8080});

// Broadcast to all.
wss.broadcast = function broadcast(data) {
    wss.clients.forEach(function each(client) {
        if (client.readyState === WebSocket.OPEN) {
            client.send(data);
        }
    });
};

wss.on('connection', function connection(ws) {
    ws.on('message', function incoming(data) {
        // Broadcast to everyone else.
        wss.clients.forEach(function each(client) {
            if (client !== ws && client.readyState === WebSocket.OPEN) {
                client.send(data);
            }
        });
    });
});
```

Using ws module

Readings

- Node WebSockets
- JEE WebSockets

Language Localization etc.

JEE
- Predefined, using message bundles (*.properties files)
- Also other resources handled

Node/Express
- Modules

Readings
- JEE resource handling
- GitHub - jeresig/i18n-node-2: Lightweight simple translation module ...

Conversion

Conversion GUI <-> Application and Application <-> Database

JEE

- GUI
  - Special converter classes (object <-> String)
  - Standard built (JSF) in or create own
- Database: JPA

Node

- GUI: Everything “string based”, … (but sometimes not …)
- Database: Sequelize

JEE GUI Conversion (JSF)

```
// Standard
<h:inputText id="date" value="#{order.date}" required="true">
     <f:convertDateTime type="time" dateStyle="long"/>
</h:inputText>

// Own
@FacesConverter(value = "pnumbConverter")
public class PhoneNumberConverter implements Converter {
  public Object getAsObject(FacesContext context, UIComponent
                            component, String value) {...}
 public String getAsString(FacesContext context,UIComponent
                            component, Object value) {...}
}

// Use in page
<h:inputText value="..." ....>
    <f:converter converterId="pNumbConverter"/>
</h:inputText>
```

If specific needs needs

- Standard converters for: DateTime, Number, Boolean, Byte, Character, Double, Float, Integer, Long, Short, BigDecimal, BigInteger or …
- … Custom converters (classes)

Standard converters

- Using tags in pages

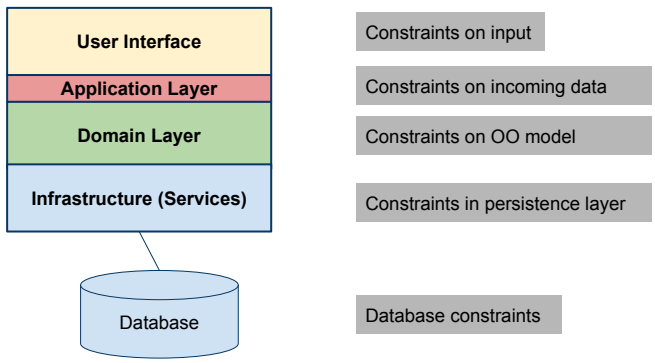
Custom converters

- Classes implementing Converter

Validation

Principle

- All layers should validate incoming data!
- Client side and server side



JEE Validation

```
// JSF page
<h:inputText validatorMessage="Must be 1-10 (incl)">
     <f:validateLongRange minimum="1" maximum="10"/>
</h:inputText>
```

GUI

```
// CDI Bean
Bean Validation (in beans)
@Digits(integer=2, fraction=0)
@Min(value = 1, message = "To small")
private int value;
@NotNull
@Size(min = 1, max = 8, message = "Must use 1-8 chars")
private String data;
@Pattern(regexp = "[A_Z][a-z]*")
private String pattern;
```

Application layer

```
// Entity Class
@Column(unique=true, nullable=false)
@Column(precision=8, scale=2)
@Column(length=40)
```

Persistence layer

Node/Express validation

```
// express-validation module
var validate = require('express-validation')
app.post('/login', validate(validation.login), function(req, res){
    res.json(200);
});

module.exports = { body: {
    email: Joi.string().email().required(),
    password: Joi.string().regex(/[a-zA-Z0-9]{3,30}/).required()
}};

// Sequelize
var ValidateMe = sequelize.define('foo', {
  foo: {
    type: Sequelize.STRING,
    validate: {
        is: ["^[a-z]+$",'i'],
        notNull: true,
        ...
    }
```

Exceptionhandling

JEE

- Predefined in some places (JSF, CDI), possibly using message bundles
- JEE 8 add global exception handling
- Best practices (?)

Node/Express

- Production practices

Readings

- Node exceptionhandling

Summary and Project Kick-Off

Summarize techniques, designs and prepare for the seminars!