

```

<?php
require("ngapuluokat/tietokanta.php");
require("ngapuluokat/kayttaja.php");
require("kehys/virheiterator.php");
require("kehys/virhe.php");

/**
 * Tämä on kaiken toiminnan ydin: itse kehysluokka. Kun pääohjelmaa
 * kutsutaan, se ensin luo tämän luokan ilmentymän ja antaa sitten sille
 * kaiken kontrollin halutun toiminnon käsittelemiseksi.
 *
 * SuomipelitKehys pääasiassa hoitaa toimintojen koordinoinnin ja välittää
 * toimintopyynnön oikealle toiminto-oliolle, sekä auttaa sitä käsittelemällä
 * ja tarkistamalla jo valmiiksi kaiken syötteenä lähetetyn datan
 * oikeellisuuden.
 *
 * TODO: kaikki tietotyyppien tarkistukset kuntoon
 */
class SuomipelitKehys
{
    /**
     * @var      Toiminto-olio, jonka kehys suorittaa
     * @access   private
     */
    var $toiminto;

    /**
     * @var      Yhteys tietokantaan
     * @access   private
     */
    var $tietokanta;

    /**
     * @var      Käyttäjän tiedot Käyttäjä-tyyppisen olion muodossa
     * @access   private
     */
    var $käyttäjä;

    /**
     * @var      Käyttäjän lähettämä syöte assosiatiivisessa taulukossa
     * @access   private
     */
    var $syöte;

    /**
     * @var      Hakemisto, josta kehys etsii siltä pyydettyä toimintoa
     * @access   private
     */
    var $toimintohakemisto;

    /**
     * @var      Tiedot mahdollisesta virheestä tai onnistumisesta,
     *          joka halutaan välittää toiminnolle
     * @access   private
     */
    var $saapuvaVirhe;

    /**
     * @var      Tiedot mahdollisesta virheestä, jonka toiminto haluaa
     *          välittää seuraavalle sivulle.
     * @access   private
     */
    var $lähteväVirhe;

    /**

```

```

* @var      Tiedot kaikista parametreja tarkistaessa eteen tulleista
*            tietotyyppivirheistä. Tällä hetkellä taulukko ei ole vielä
*            kunnolla käytössä.
* @access   private
*/
var $tietotyyppivirheet;

/**
* @var      Taulukko, joka sisältää ne komponentit, jotka
*            toiminto haluaa liittää sivuille.
* @access   private
*/
var $komponentit;

/**
* Luo kehyksen. Kehystä voidaan käyttää eri ympäristöissä:
* perussivut, ylläpito jne. ja eri alueiden toiminnot on sijoitettu
* eri hakemistoihin sivu-hakemiston alla. Hakemisto annetaan kehykselle
* parametrina.
*
* @param    string  Hakemisto, jonka alta toimintoja tulee etsiä.
*/
function SuomipelitKehys($hakemisto)
{
    // Tallennetaan toimintahakemisto myöhempää käyttöä varten
    $this->toimintahakemisto = $hakemisto;

    // Luodaan yhteys tietokantaan
    $this->tietokanta = new Tietokanta();
    sp_assert($this->tietokanta != null, TIETOKANTA_VIRHE_EI_YHTEYTTÄ);

    // Tallennetaan syötteenä annetut toiminto ja sen parametrit
    $this->syöte = $_POST;
    foreach($_GET as $avain => $arvo)
    {
        $this->syöte[$avain] = $arvo;
    }

    // Erotellaan syötteestä pois virheisiin liittyvät tiedot:
    // virheet merkitään syötteessä seuraavasti: r=<tyyppi>:<id>
    if(array_key_exists("r", $this->syöte))
    {
        $arvo = $this->syöte["r"];

        $tyyppi = intval($arvo);
        $arvo = substr($arvo, strpos($arvo, ":")+1);
        $sid = intval($arvo);

        $this->saapuvaVirhe = new Virhe($this->tietokanta, $tyyppi, $sid);

        unset($this->syöte["r"]);
    }
    else
    {
        $this->saapuvaVirhe = null;
    }

    // Luodaan käyttäjäolio, jotta toiminto voi tarkistaa käyttäjän
    // oikeudet jne...
    $this->käyttäjä = new Käyttäjä($this->tietokanta);

    // Toimintoa ei ole vielä luotu
    $this->toiminto = null;
}

```

```

        // Lähtevää virhettä ei ole vielä luotu
        $this->lähteväVirhe = null;
    }

    /**
     * Destruktori, joka sulkee kehyksen
     */
    function tuhoaKehys()
    {
        $this->tietokanta->suljeYhteys();
    }

    /**
     * Suorittaa kehyksen perustehtävät eli etsii pyydetyn toiminnon,
     * tarkistaa sen parametrin ja suorittaa toiminnon. Jos toimintoa ei
     * löydy, lopettaa kehyksen suorituksen.
     */
    function teeToiminto()
    {
        // Haetaan toiminto. Jos toimintoa ei ole olemassa, lopetetaan
        // ohjelman suoritus
        $toiminto = $this->haeToiminto($this->syöte["c"]);
        $this->toiminto = $toiminto;

        if($toiminto == null)
        {
            sp_kill(SYÖTE_VIRHE_EI_TOIMINTOA);
        }

        // Tarkistetaan, onko käyttäjällä toiminnon vaatimat oikeudet
        $oikeudet = $toiminto->haeVähimmäisoikeudet();
        if($this->tarkistaOikeudet($oikeudet))
        {
            // Tarkistetaan käyttäjän antaman syötteen oikeellisuus
            $parametrit = $toiminto->haeParametrilista();
            if(!$this->tarkistaParametrit($parametrit))
            {
                $this->tulostaSyötevirheilmoitus($toiminto->haeSyötevirhesivu());
            }
            else
            {
                // Jos syötteiden mukana on tullut virhe-/onnistumisilmoitus, annetaan
                // toiminnon käsitellä se ennen varsinaista toiminnon suorittamista.
                if($this->saapuvaVirhe != null)
                {
                    $toiminto->käsitteleIlmoitus($this->saapuvaVirhe);
                }

                // Jos kaikki on kunnossa, suoritetaan toiminto
                $toiminto->teeToiminto($this->syöte);

                // Tulostetaan toiminnon tulosteet
                if($toiminto->tulostetaankoSivu() == true)
                {
                    // Haetaan toiminnon haluamat komponentit
                    $this->luoSivukomponentit($toiminto->haeHalututKomponentit());

                    $this->tulostaAlku();
                    $this->tulosta($toiminto->haePohjatiedosto(), $toiminto->haeTulos());
                    $this->tulostaLoppu();
                }
                // Siirrytään toiminnon haluamalle sivulle
                else
                {
                    $osoite = $toiminto->haeKohdesivu();

```

```

        // TODO: Tässä kohti voidaan tehdä tarkistus, että sivu
        // tosiaan on Suomipelit.comin kehyksen alainen sivu ennen
        // kuin lisätään virheilmoitus perään...

        if($this->lähteväVirhe)
        {
            $osoite .= "&r=". $this->lähteväVirhe->haeVirhetyyppi()
                .": ". $this->lähteväVirhe->haeId();
        }

        header("Location: ".$osoite);
    }
}

else
{
    sp_kill("Sinulla ei ole toiminnon suorittamiseen tarvittavia oikeuksia.");
}
}

/**
 * Hakee pyydetyn toiminnon ja luo siitä instanssin, jonka sitten
 * palauttaa.
 *
 * @param string    Toimintoluokan (ja -tiedoston) nimi
 * @return          Toiminnon, jos se on olemassa. Muulloin null.
 */
function haeToiminto($toiminnonNimi)
{
    // Tarkistetaan, onko pyydettyä toimintoa olemassa
    $tiedostonimi = "sivu/toiminnot/".$this->toimintohakemisto."/". $toiminnonNimi.".
php";

    sp_debugText("Pyydetty toiminto: ".$tiedostonimi);

    if(!file_exists($tiedostonimi))
    {
        sp_debugText("Tiedostoa ei löytynyt.");
        return null;
    }
    else
    {
        // Haetaan toiminnon esittely
        require($tiedostonimi);

        // Luodaan toiminnon ilmentymä
        // TODO: Tee toiminnon hakeminen sellaiseksi, että se osaa
        // luoda juuri pyydettyntyyppisen olion.
        $toiminto = new $toiminnonNimi($this->tietokanta, $this->käyttäjä, $this);

        if($toiminto) sp_debugText("Luotiin toiminto...");

        return $toiminto;
    }
}

/**
 * Tarkistaa, että käyttäjällä on toiminnon vaatimat oikeudet.
 * Oikeudet ilmaistaan seuraavasti:
 * <ul>
 * <li><tt>KAIKKI</tt> = kaikki saavat käyttää toimintoa</li>
 * <li><tt>YLLÄPITO</tt> = kaikki ylläpitäjät</li>
 * <li><tt>PELIT</tt> = peliarvostelijat</li>
 * <li><tt>UUTISET</tt> = uutistoimittajat</li>
 * <li><tt>ARTIKKELIT</tt> = artikkelihenkilöt</li>

```

```

* <li><tt>JOHTO</tt> = eri alueiden vastuuhenkilöt</li>
* </ul>
* Lisää erilaisia oikeusryhmiä voidaan lisätä tarpeen vaatiessa.
*
* @param String Oikeusryhmän nimi. Joku edellä listatuista
*           vaihtoehtoista. Jos annettu merkkijono ei ole mikään tunnistettu
*           oikeusryhmä, käsitellään sitä tunnuksena, ja tarkistetaan,
*           onko käyttäjän tunnus annettu merkkijono.
*/
function tarkistaOikeudet($oikeudet)
{
    switch($oikeudet)
    {
        // Kaikki saavat käyttää toimintoa
        case "KAIKKI":
            return true;

        // Vain ylläpitäjille
        case "YLLÄPITO":
            if($this->käyttäjä->onkoYlläpitäjä()) {
                return true;
            }
            break;

        // Peliarvostelijat
        case "PELIT":
            if($this->käyttäjä->onkoPeliarvostelija()) {
                return true;
            }
            break;

        // Uutistoimittajat
        case "UUTISET":
            if($this->käyttäjä->onkoUutistoimittaja()) {
                return true;
            }
            break;

        // Artikkelinkirjoittajat
        case "ARTIKKELIT":
            if($this->käyttäjä->onkoArtikkelinkirjoittaja()) {
                return true;
            }
            break;

        // Eri alueiden vastuuhenkilöt
        case "JOHTO":
            if($this->käyttäjä->onkoSuperkäyttäjä()) {
                return true;
            }
            break;

        // Ei tunnistettu ryhmäkoodia, oletetaan se käyttäjän
        // tunnukseksi.
        default:
            if($this->käyttäjä->haeTunnus() == $oikeudet) {
                return true;
            }
            break;
    }

    // Jos metodista ei poistuttu switch-lauseessa, ei
    // käyttäjällä ole vaadittavia oikeuksia.
    return false;
}

```

```

/**
 * Tarkistaa, että luokalle annetut parametrit täyttävät
 * parametrina annetut säännöt.
 *
 * <ul>
 * <li>numeric: lukuarvo (int, float jne...)</li>
 * <li>string: merkkijono</li>
 * <li>file: tiedosto</li>
 * <li>date: päiväys</li>
 * <li>email: sähköpostiosoite</li>
 * <li>url: internet-osoite</li>
 * </ul>
 *
 * @param array Taulukko, joka sisältää kaikkien parametrien nimet
 *           ja tietotyyppit.
 */
function tarkistaParametrit($parametrisäännöt)
{
    $ok = true;

    // Jos parametrisääntöjä ei ole asetettu, kaikki parametrit kelpaavat
    if(!is_array($parametrisäännöt))
    {
        return true;
    }

    // Muuten tarkistetaan kaikki vaaditut parametrit. Ylimääräisistä
    // parametreista ei välitetä.
    foreach($parametrisäännöt as $nimi => $tyyppi)
    {
        if($this->tarkistaParametri($nimi, $tyyppi) == false)
        {
            $this->tietotyyppivirheet[$nimi] = $tyyppi;
            $ok = false;
        }
    }

    return $ok;
}

/**
 * Tarkistaa, onko syötteenä annettu parametri, jonka nimi = $nimi ja
 * tyyppi = $tyyppi.
 *
 * @param String Tarkistettavan syöteparametrin nimi
 * @param String Tarkistettavan syöteparametrin tyyppi
 */
function tarkistaParametri($nimi, $tyyppi)
{
    $onkoKunnossa = true;

    // Tarkistetaan, että parametri on annettu
    if(!array_key_exists($nimi, $this->syöte))
    {
        return false;
    }

    // Tarkistetaan, että sen tyyppi on oikea
    switch($tyyppi)
    {
        case "numeric":
            $onkoKunnossa = is_numeric($this->syöte[$nimi]);
    }
}

```

```

        break;

    case "int":
        $onkoKunnossa = is_numeric($this->syöte[$nimi]);
        break;

    case "string":
        $onkoKunnossa = is_string($this->syöte[$nimi]);
        break;

    case "date":
        $onkoKunnossa = tarkistaPäiväys($this->syöte[$nimi]);
        break;

    case "email":
        $onkoKunnossa = tarkistaEmail($this->syöte[$nimi]);
        break;

    case "url":
        $onkoKunnossa = tarkistaURL($this->syöte[$nimi]);
        break;

    case "id":
        $onkoKunnossa = is_int($this->syöte[$nimi]);
        break;

    default:
        $onkoKunnossa = false;
        sp_debugText("Tuntematon parametrityyppi");
        break;
    }

    return $onkoKunnossa;
}

/**
 * Tarkistaa, onko annettu syöte oikein muotoiltu päiväys
 */
function tarkistaPäiväys($päiväys)
{

}

/**
 * Tarkistaa, onko annettu syöte oikein muotoiltu sähköpostiosoite
 */
function tarkistaSähköposti($osoite)
{

}

/**
 * Tarkistaa, onko annettu syöte oikein muotoiltu sähköpostiosoite
 */
function tarkistaURL($osoite)
{

}

/**
 * Luo kaikki toiminnon haluamat sivukomponentit ja tallentaa

```

```

* ne taulukkoon. Tämä taulukko sitten välitetään sitten alku-
* ja loppu-templateille, jotka tulostavat komponentit oikeisiin
* paikkoihin.
*
* @param Array Taulukko, johon on tallennettu haluttujen
* komponenttien nimet ja sijoituspaikat:
* Esim. "K_UusimmatAiheet" => "oikea_1"
* Sijoituspaikkojen koodit päätetään sivukehikkoa
* suunniteltaessa.
*/
function luoSivukomponentit($komponenttienNimet)
{
    // Luodaan ensin sivuston oletuskomponentit
    $this->luoOletuskomponentit();

    // Ja sitten toiminnon haluamat
    if(is_array($komponenttienNimet))
    {
        foreach($komponenttienNimet as $komponentinNimi => $sijoituspaikka)
        {
            // Tarkistetaan, onko pyydetty komponentti olemassa
            $tiedostonimi = "sivu/komponentit/".$this->toimintohakemisto."/".
            $komponentinNimi.".php";
            sp_debugText("Pyydetty komponentti: ".$tiedostonimi);

            if(!file_exists($tiedostonimi))
            {
                sp_debugText("Tiedostoa ei löytynyt.");
                $this->komponentit[$sijoituspaikka] = null;
            }
            else
            {
                require($tiedostonimi);

                $komponentti = new $komponentinNimi($this->tietokanta, $this->käyttäjä);

                if($komponentti)
                {
                    sp_debugText("Luotiin komponentti...");
                    $this->komponentit[$sijoituspaikka] = $komponentti;
                }
            }
        }
    }
}

/**
* Luo sivuston oletuskomponentit hakemalla komponenttien nimet ja paikat
* asetustiedostosta.
*
* TODO: tällä hetkellä tässä kohti huijataan, ja komponentit on
* hard-koodattu tähän funktioon. Korjaa tämä asia...
*/
function luoOletuskomponentit()
{
    require("sivu/komponentit/yleiset/K_OikeatLinkit.php");
    $this->komponentit["oikea_1"] =
        new K_OikeatLinkit($this->tietokanta, $this->käyttäjä);

    require("sivu/komponentit/yleiset/K_Kirjautumislomake.php");
    $this->komponentit["kirjautumisalue"] =
        new K_Kirjautumislomake($this->tietokanta, $this->käyttäjä);

    require("sivu/komponentit/yleiset/K_KäyttäjäTervehdys.php");

```



```

        $this->komponentit["infokenttä"] =
            new K_KäyttäjäTervehdys($this->tietokanta, $this->käyttäjä);
    }

/**
 * Tulostaa toiminnon tulokset käyttäen tiedoissa välitettävää
 * pohjatiedostoa. Pohjatiedoston nimi välitetään taulukossa avaimella
 * <code>pohjaTiedosto</code>. Toiminnon tulosten pohjatiedostoon ei
 * kuitenkaan laiteta yleistä sivukehystä, joka sen sijaan tulostetaan
 * aina tietyistä omista pohjatiedostoista - joille myös välitetään
 * toiminnon palauttama tulostaulukko.
 *
 * @param array Assosiatiivinen taulukko, jonka arvoina ovat kaikki
 *             tulostettavat asiat.
 */
function tulosta($pohjatiedostonimi, $tiedot)
{
    // Haetaan käytettävä pohja
    $pohjatiedosto = "sivu/ulkoasu/html/".$this->toimintohakemisto."/".
    $pohjatiedostonimi;
    $pohja = new Template($pohjatiedosto, $tiedot);

    sp_debugText($pohja->pohjatiedosto);

    $pohja->tulostaSivu();
}

/**
 * Tulostaa sivukehikon alun, eli kaiken sen, mitä halutaan näyttää ennen
 * itse toiminnon tulostusta.
 */
function tulostaAlku()
{
    $tiedot = array();

    // Haetaan otsikko
    $tiedot["otsikko"] = $this->toiminto->haeOtsikko();

    // Tulostettavat sivukomponentit mukaan kuvioon
    if(is_array($this->komponentit))
    {
        foreach($this->komponentit as $sijoituspaikka => $komponentti)
        {
            $tiedot[$sijoituspaikka] = $komponentti;
            sp_debugText("Sijoituspaikkaan ".$sijoituspaikka." menee komponentti");
        }
    }

    sp_debugText("Tulostetaan sivukehikon alku");
    $pohja = new Template("sivu/ulkoasu/html/yleiset/alku.html", $tiedot);
    $pohja->tulostaSivu();
}

/**
 * Tulostaa sivukehikon lopun, eli kaiken sen, mitä halutaan näyttää
 * itse toiminnon jälkeen.
 */
function tulostaLoppu()
{
    $tiedot = array();

    // Tulostettavat sivukomponentit mukaan kuvioon

```

```

        if(is_array($this->komponentit))
        {
            foreach($this->komponentit as $sijoituspaikka => $komponentti)
            {
                $tiedot[$sijoituspaikka] = $komponentti;
            }
        }

        sp_debugText("Tulostetaan sivukehikon loppu");
        $pohja = new Template("sivu/ulkoasu/html/yleiset/loppu.html", $tiedot);
        $pohja->tulostaSivu();
    }

    /**
     * Tulostaa virheilmoituksen, joka näytetään silloin, kun syöte
     * on väärä. Mikäli toiminnolle ei ole määriteltä omaa syötevirheilmoitus-
     * templatea, käytetään oletusversiota.
     */
    function tulostaSyötevirheilmoitus($sivu)
    {
        if($sivu == null)
        {
            $sivu = "sivu/ulkoasu/html/yleiset/syotevirhe.html";
        }

        $pohja = new Template($sivu, $this->tietotyyppivirheet);
        $pohja->tulostaSivu();
    }

    /**
     * Lisää toiminnon lähettämän virheilmoituksen kehyksen muistissa
     * olevaan lähtevään virheeseen.
     *
     * @param boolean Tieto siitä, onko kyseessä virhe (true) vai
     *               onnistuminen (false)
     * @param String Välitettävä ilmoitus.
     */
    function lisääIlmoitus($onkoVirhe, $virheilmoitus)
    {
        // Jos tämä on ensimmäinen ilmoitus, luodaan virhe-olio
        if($this->lähteväVirhe == null)
        {
            if($onkoVirhe == false)
            {
                // TODO: Muuta käyttämään vakioita heti kun tiedetään,
                // miten se parhaiten onnistuu...
                $this->lähteväVirhe = new Virhe($this->tietokanta, 1);
            }
            else
            {
                $this->lähteväVirhe = new Virhe($this->tietokanta, 2);
            }
        }

        // Sitten lisätään ilmoitukseen annettu rivi
        $this->lähteväVirhe->lisääIlmoitus($virheilmoitus);
    }
}
?>

```